Prosodic Event Detection for Speech Understanding using Neural Networks

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Philosophie (Dr. phil.) genehmigte Abhandlung.

Vorgelegt von

Sabrina Stehwien

aus Toronto, Kanada

Hauptberichter Prof. Dr. Ngoc Thang Vu
Mitberichter Prof. Dr. Bernd Möbius

Tag der mündlichen Prüfung: 16. Dezember 2019

Institut für Maschinelle Sprachverarbeitung
der Universität Stuttgart

2020

Erklärung (Statement of Authorship)
Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst habe und dabei keine andere als die angegebene Literatur verwendet habe. Alle Zitate und sinngemäßen Entlehnungen sind als solche unter genauer Angabe der Quelle gekennzeichnet.

I hereby declare that this text is the result of my own work and that I have not used sources without declaration in the text. Any thoughts from others or literal quotations are clearly marked.

_____

(Sabrina Stehwien)

Contents

List of Abbreviations

ASR	Automatic speech recognition
ATIS	Airline Travel Information Systems corpus
BDC	Boston Directions Corpus
BURNC	Boston University Radio News Corpus
CNN	Convolutional neural network
CoNLL	Conference on Computational Natural Language Learning
CPU	Central processing unit
DIRNDL	Diskurs-Informations-Radio-Nachrichten-Datenbank für Linguistische Analysen
FFNN	Feed-forward neural network
GPU	Graphical processing unit
GRAIN	German Radio Interviews corpus
GToBI(S)	German Tones and Break Indices (Stuttgart system)
HtNR	Harmonics-to-noise ratio
LeaP	"Learning prosody in a foreign language" project corpus
LSTM	Long short-term memory network
NLP	Natural language processing
NN	Neural network
NP	Noun phrase
OOV	Out-of-vocabulary
PaIntE	Parametrized Intonation Events
POS	Part of speech
PP	Prepositional phrase
RNN	Recurrent neural network
RMS	Root mean square
SGD	Stochastic Gradient Descent
SLU	Spoken language understanding
ToBI	Tones and Break Indices
TTS	Text-to-speech synthesis
WER	Word error rate
ZCR	Zero-crossing rate

List of Figures

List of Tables

Acknowledgements

I would like to thank my supervisor Ngoc Thang Vu for making this thesis possible and for his help and guidance during my research. Apart from providing instrumental advice on neural networks, he allowed me to balance *computation* and *linguistics* in this thesis while keeping the "big picture" in mind. I appreciate that he dedicated time for thorough but flexible supervision, and placed great importance on creating a productive and positive atmosphere. I would also like to extend my gratitude to Bernd Möbius and Jonas Kuhn for participating in the committee and enabling a pleasant defense, as well as for providing mentoring and feedback.

Antje Schweitzer adopted the role of "unofficial co-advisor" and contributed greatly with her expertise on prosody modeling and phonetics. I am deeply grateful for her support, mentoring and kind words over the last years since my time as a student assistant at IMS. Her encouragement played a key role in my decision to pursue my own research at this institute. I would also like to thank Caroline Féry for her wise perspective and helpful advice as a mentor. I am grateful that she took the time to host visits in Frankfurt, which were always a welcome change of scene for me, and provided me the opportunity to deepen my knowledge on prosody. Furthermore I would like to thank Andrew Rosenberg for sharing his expertise on prosodic event detection.

Working at IMS has been an enjoyable experience, and my sincere thanks goes out to my colleagues for this memorable time, including inspiring collaborations, conversations and coffee breaks. I would like to thank my co-authors of various conference papers whose contributions found their way into this thesis, especially Ina Rösiger, Arndt Riester, Katrin Schweitzer and Kerstin Eckart. I would also like to acknowledge the Digital Phonetics group that, not long after it was established at IMS, quickly grew into a fun and supportive network of talented young researchers.

Finally, special thanks to my parents for their invaluable support and guidance without which this thesis would not have been written.

Abstract

The prosody of West-Germanic languages such as English and German contributes to the interpretation and disambiguation of speech. The notion of exploiting this connection for speech technology, although regarded as challenging, has motivated a notable amount of work on spoken dialogue systems and speech recognition. Since standard automatic speech recognition (ASR) systems do not recognize prosody, this information is lost in a pipeline setup for speech understanding that involves several components that aim to extract meaning from speech.

One of the main contributions of this thesis is therefore the development and investigation of an efficient method of prosodic event detection. As most common methods of automatically recognizing prosody, it consists of a supervised machine learning task that predicts pitch accents and phrase boundaries from time-aligned speech data. We use a convolutional neural network (CNN) for prosodic event detection because of its ability to learn local patterns in the input. We present several experiments showing that the model performs well on English and German data using a small set of acoustic descriptors as input features. The experimental results show that the performance is comparable to methods that use more linguistically informed features and that this model can be readily applied to new datasets.

We tested the use of automatically predicted and word-based binary absence or presence labels of prosodic events for three different applications: We first show that pitch accent features, obtained using a prosodic event detector developed by Schweitzer (2010), can assist the task of semantic slot filling on a recognized version of the English benchmark dataset ATIS (Hemphill et al., 1990). This experiment demonstrates that no manual annotations of the speech data are required during test time. Motivated by a pilot study by Rösiger and Riester (2015), we present a collaborative experiment showing that even automatically predicted prosodic information significantly improves the performance of coreference resolution on the German DIRNDL radio new corpus (Eckart et al., 2012). The prosodic labels were predicted using the CNN-based method, pretrained on the Boston University Radio News Corpus (Ostendorf et al., 1995). Our

contribution to the silver-standard corpus resource GRAIN (Schweitzer et al., 2018) was the provide prosodic event labels using the CNN-based approach. We show that these more coarse-grained labels can be combined with a more linguistically informed method of fine-grained labeling to improve the confidence of the annotations.

The final focus of this thesis is the analysis of the feature representation learned by the neural network for prosodic event detection. We use a linear regression analysis to investigate latent acoustic, temporal and context word information in this representation. The findings confirm our assumption that the CNN is able to learn word duration and relevant voiced regions on its own from the frame-based acoustic input data.

This work shows that neural networks are an efficient method of prosodic event detection. The resulting models simplify research on integrating prosodic information in various downstream tasks and they can be shown to encode important prosodic correlates in the learned feature representations. We conclude that prosodic modeling based on state-of-the-art neural networks can help advance the field of automatic speech understanding and bridge the gap in research on the processing of spoken and written language.

Deutsche Zusammenfassung

Die Prosodie der westgermanischen Sprachfamilie, zu der die englische und die deutsche Sprache gehören, trägt einen bedeutenden Teil zum Verständnis und zur Disambiguierung von gesprochenen Äußerungen bei. Der Wunsch, diesen Zusammenhang für die Sprachtechnologie auszunutzen, hat in der Vergangenheit mehrere Forschungsarbeiten unter Anderem im Bereich der Sprachdialogsysteme und der Spracherkennung motiviert, obwohl die prosodische Modellierung und ihre Verwendung in technischen Ansätzen allgemein als Herausforderung gilt. Gängige Systeme zur automatischen Spracherkennung (*automatic speech recognition*, ASR) geben keine Information über die Prosodie der erkannten Äußerungen aus. Dies dazu führt dazu, dass die prosodische Information in einem Versuchsaufbau fehlt, in der ein ASR-System die erste Komponente einer Pipeline darstellt und in der die Prosodie für spätere Module zum automatischen Sprachverstehen nutzbar gemacht werden soll.

Auf dieser Problematik basiert ein wesentlicher Beitrag dieser Arbeit, nämlich die Entwicklung und Erforschung einer effizienten Methode zur automatischen Erkennung von Prosodie. Der häufigste Ansatz hierzu ist die Erkennung von prosodischen Ereignissen (*prosodic event detection*), nämlich Pitchakzente und Phrasengrenztöne, aus segmentierten Sprachdaten mithilfe von überwachten maschinellen Lernverfahren. In dieser Arbeit wird eine spezielle Netzwerkarchitektur, bezeichnet als *Convolutional Neural Network* (CNN), für die Erkennung prosodischer Ereignisse eingesetzt, da diese Netze lokale Muster in den Daten modellieren. In mehrere Experimenten wird gezeigt, dass dieses Modell mit nur einer geringen Menge an akustischen Features gute Ergebnisse auf englischen und deutschen Sprachdaten erzielt. Diese sind vergleichbar mit den Ergebnissen von ähnlichen Methoden, die zudem auf Features basieren, die mit mehr Vorverarbeitungsaufwand hergestellt sind.

In einer Reihe von Experimenten wird die automatisch erstellte Information über das Vorhandensein von prosodischen Ereignissen auf Wörtern für verschiedene Anwendungen getestet. Für das Semantic Slot Filling auf einer automatisch transkribierten Version des englischen Bezugskorpus ATIS (Hemphill et al., 1990) kann die Information über

das Vorhandensein eines Pitchakzentes, vorhergesagt mit einer Methode nach Schweitzer (2010), hilfreich sein. Dieses Experiment zeigt außerdem, dass während der Testphase des Lernverfahrens keine manuellen Annotationen nötig sind. Ein kollaborativ durchgeführtes Experiment, basierend auf der Pilotstudie von Rösiger und Riester (2015), zeigt, dass automatisch ermittelte prosodische Features die Ergebnisse eines Koreferenzauflösungssystems signifikant verbessern. Für dieses System wurde die CNN-basierte Methode zur prosodischen Erkennung auf dem Boston University Radio News Corpus (Ostendorf et al., 1995) trainiert, und auf dem deutschen Korpus DIRNDL (Eckart et al., 2012) für die Koreferenzauflösung angewendet. Diese Methode der eher groben Erkennung des Vorhandenseins prosodischer Ereignisse kann mit aufwendigeren Methoden einer genaueren Erkennung kombiniert und ergänzt werden, was in einem Beitrag zu den automatischen Annotationen der *Silver-Standard*-Korpusressource GRAIN (Schweitzer et al., 2018) veranschaulicht wird.

Der abschließende Schwerpunkt dieser Arbeit ist die Analyse der Features, die das neuronale Netz für die prosodische Erkennung erstellt. Die verwendete Methode besteht aus einer linearen Regressionsanalyse zur Untersuchung der latenten akustischen und temporalen Repräsentationen, sowie der aus den umliegenden Kontextwörtern. Die Ergebnisse der Analysen bestätigen die Annahme, dass das CNN die Wortdauer sowie relevante stimmhafte Regionen aus den akustischen, Frame-basierten Eingabedaten lernen kann.

Zusammenfassend zeigt diese Arbeit, dass neuronale Netze eine effiziente Methode für die Erkennung prosodischer Ereignisse darstellen. Die daraus entstehenden Modelle können die Erforschung der Verwendung von Prosodie für das maschinelle Sprachverstehen vereinfachen. Zudem enthalten die gelernten Repräsentationen nachweisbar prosodische Korrelate. Die prosodische Modellierung mithilfe moderner neuronale Netze leistet daher einen wichtigen Beitrag für die Gebiete der Sprachtechnologie, die die Verarbeitung von gesprochener und geschriebener Sprache verbinden.

1 Introduction

1.1 Motivation

Prosody is an aspect of language that plays a significant role for meaning in English and German (Pierrehumbert and Hirschberg, 1990; Féry, 2017). The term *prosody* refers to the intonation, rhythm, intensity and voice quality of a language. These cues are used, for example, to structure and disambiguate utterances and to convey speaker state and emotion. Therefore, it is a property of spoken language that carries important linguistic and paralinguistic information that is not conveyed through text alone.

Since prosody plays a key role for understanding in human discourse, it is straightforward to assume that it also can help in language technology. Motivated by this notion, many studies have explored the use of prosody in speech processing applications. Since natural sounding synthesized speech requires appropriate prosody, the field of text-to-speech synthesis (TTS) was one of the first research directions to deal with the automatic processing of prosody (Hirschberg, 2002). In contrast to applications dealing with speech *generation*, such as TTS, applications for speech *understanding* still do not include standard and systematic methods of integrating prosodic information. As Hirschberg (2002, p. 32) has pointed out, the field has "been slow" to integrate prosody into actual systems despite sufficient motivation. While there has been research towards integrating prosody into automatic speech recognition (ASR) systems in the past (Waibel, 1988; Ostendorf et al., 2003; Shriberg and Stolcke, 2004), state-of-the-art methods of ASR ignore the prosody of speech and focus only on the speech-to-text task. The main reason for this is that prosodic information is not necessary to produce good quality text output. Furthermore, Vassière (1988, p. 96) has mentioned that "[i]t is often stated that prosody is too complex for straightforward integration into an existing system", and that most methods of ASR and parsing are not flexible enough to be augmented by further prosodic knowledge sources. For this reason, the automatic recognition of prosody is usually implemented as a separate task. A number of notable publications on integrating prosody into speech processing and spoken dialog systems first emerged

in the 1990s and early 2000s by American and German research groups, notably AT&T (Hirschberg, 2002), SRI International (Shriberg and Stolcke, 2004), and VERBMOBIL (Wahlster, 2000).

Today, technologies that combine speech and language processing, such as spoken dialog systems or applications for multi-modal understanding, are attracting much interest. Therefore, research that aims to overcome the challenges of combining tasks that traditionally process either spoken or written language, for example natural language processing (NLP) tasks applied to ASR output, is becoming more and more relevant.

Over the course of the last decade, neural networks have become state-of-the-art in most areas of speech and language processing (Collobert et al., 2011). The availability of higher computing power paired with increasing amounts of data have made deep neural networks feasible and powerful learning methods, replacing the former state-of-the-art in machine translation (Bahdanau et al., 2015) and automatic speech recognition (Hinton et al., 2012). Following the increasing popularity and success of neural networks in this field, the question of what these models are learning is an important question for current research (Linzen et al., 2019).

This thesis combines several notions: the use of prosody in speech understanding, the modeling of prosody using state-of-the-art neural network methods, and the analysis of these models. We describe the research context and gaps that motivate the approaches taken in this work in the following.

1.2 Research context

Applications for speech processing can be grouped into two categories: those that take speech as input and aim to recognize and understand the content, and those that generate utterances and convert them into speech. A spoken dialogue system combines both these directions, as illustrated in Figure 1.1. While prosody is relevant for all subtasks, this thesis deals only with tasks related to speech **understanding**.

Prosodic information is not only useful in spoken dialogue systems, but also as an additional knowledge source for text-based tasks in NLP such as parsing (Kahn et al., 2005) or coreference resolution (Rösiger and Riester, 2015). Especially when methods originally designed for processing text, such as slot filling (also referred to as semantic parsing) are confronted with spoken language or even automatically recognized speech, it is often observed that the performance drops substantially, because of recognition errors (Mesnil et al., 2015) and the differences between spoken and written language

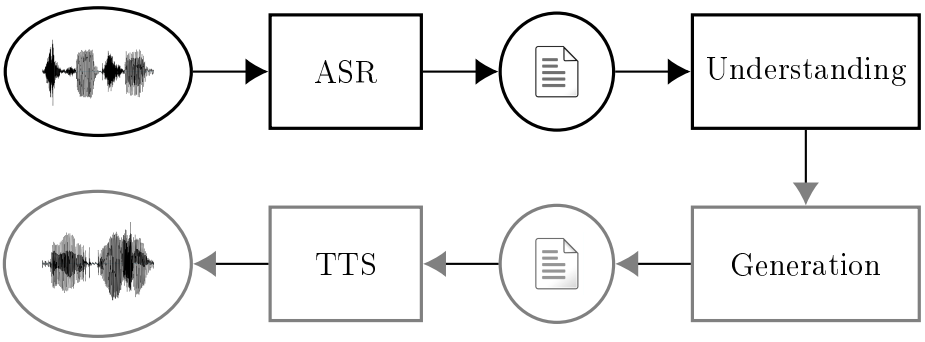


Figure 1.1: Pipeline of a spoken dialogue system. The direction of processing on the top part consists of modules for speech recognition (ASR) and understanding, the direction at the bottom for generation and text-to-speech synthesis (TTS).

(Amoia et al., 2012). Adding knowledge sources not typically encoded in text, such as prosody, may help to bridge this performance gap. Furthermore, since spoken language applications have an ASR system as the first processing component, it would be helpful to take these differences into account by evaluating the downstream systems on recognized text.

Prosody can be automatically recognized using several different approaches. Figure 1.2 provides an overview of strategies of modeling prosody for speech understanding. The approach taken in this thesis is a *pipeline* approach, that is, the prosody recognition step is separate from the other subtasks in the system and inserted between the ASR and actual understanding tasks. Most previous research on this topic adopts a pipeline-style setup, including, for example, the method proposed by Nöth et al. (2002) for spoken dialogue understanding.

In contrast, a direct modeling system (or "end-to-end") skips the intermediate steps by jointly training a large model directly on speech input to solve the end task, typically by using neural networks. For example, Tran et al. (2018) implemented a joint modeling system for neural parsing that takes prosodic phrase boundary information into account. The main idea behind direct modeling is to exploit as much low-level information as possible and to leave it up to the neural network to model the input data in a way that is best suited to solve the task. While such approaches are gaining in popularity, they have not completely replaced standard pipelines since they require large amounts of annotated training data and are not readily interpreted: It is often not apparent what prosodic information was learned since the evaluation is carried out on the downstream task.

The separate prosodic modeling step in a pipeline can further be separated into two

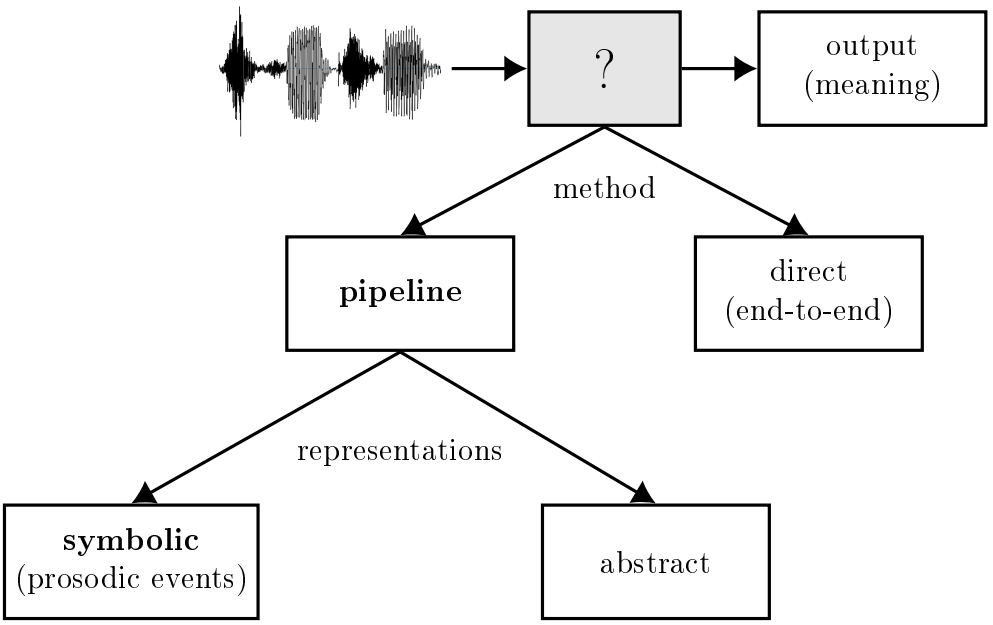


Figure 1.2: Modeling approaches for using prosodic information to help obtain the desired output, usually a representation of meaning used for speech understanding. The approach taken in this thesis the pipeline approach that involves the detection of symbolic prosodic events.

different approaches. This thesis deals with the recognition of **symbolic** representations of prosody, that is, labels that can be interpreted and annotated by human experts. They are distinguished from more higher-level, *abstract* representations, such as features learned by or extracted from the intermediate layers of a neural network. Wang et al. (2016), for example, describe a method belonging to the latter group. Symbolic representations of prosody usually consist of prosodic *events*, as described by the tone-sequence model of intonation (Pierrehumbert, 1980) and the ToBI labeling standard (Silverman et al., 1992). Information on the absence or presence of prosodic events have shown to help NLP tasks such as parsing (Kahn et al., 2005), named entity recognition (Katerenchuck and Rosenberg, 2014) or coreference resolution (Rösiger and Riester, 2015).

In the setting assumed in this work, ASR is treated as a "black box" module that converts speech into text, which is then passed on to downstream tasks (Figure 1.3). Using the time-aligned speech and text information, a separate module for *prosodic event recognition* (or *detection*) serves as an intermediate step between these two components. Prosodic event recognition refers to the task of automatically assigning a prosodic event such as a pitch accent or an intonational phrase boundary to syllables or words in

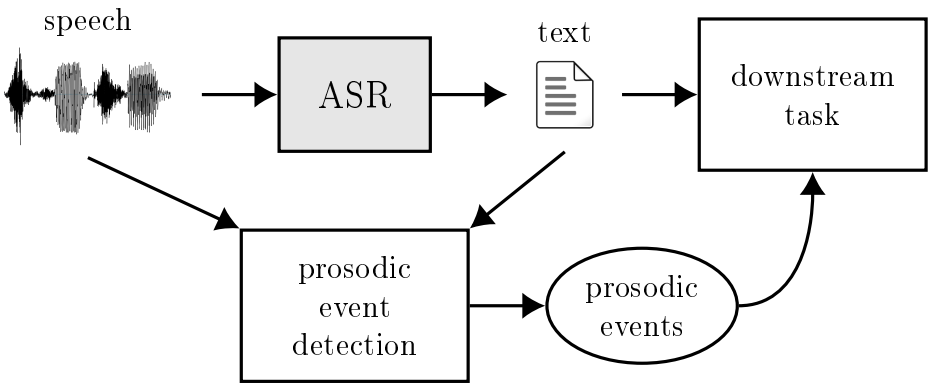


Figure 1.3: A speech understanding pipeline as defined in this thesis, which uses the text output of an ASR sytem and the speech signal to recognize prosodic events and pass these as features to downstream tasks.

transcribed speech data.

There is a large body of research on prosodic event recognition (Wightman and Ostendorf, 1994; Rosenberg, 2009; Schweitzer, 2010), which also often proposes complex systems that require a considerable amount of preprocessing and extensive feature engineering. This makes it challenging to use these methods as standalone modules that can be readily inserted into different experimental setups and used on new data.

Recently, **neural networks** have been used to successfully solve the task of prosodic event detection (Rosenberg et al., 2015; Wang et al., 2016; Li et al., 2018). The reason why they are an attractive method is not only because of their performance, but also because of their ability to replace time-consuming feature engineering. Instead, the neural network itself can be used to learn suitable feature representations automatically. The advantage of this is that the network will directly model patterns necessary for fitting the data, avoiding human error and information loss caused by manual feature selection. The convenience of delegating this task to the network, however, inherently comes at the price of explainability: It is not transparent what information these feature representations consist of. So far, this question has not attracted much research in the area of automatic prosodic modeling. A recent study by Kakouros et al. (2019), published during the course of this thesis, analyzed a neural network for prosodic modeling by analyzing the hidden layer representations. Our work is therefore among the first to investigate the learned representations of neural networks for prosodic modeling.

In sum, the existing body of research on this area still contains certain gaps, which we aim to overcome in this work. We start by defining our research goals in the next section.

1.3 Goals of this thesis

The goals of this thesis pertain to three research areas: **modeling** of prosodic events, **applications** of prosodic event modeling, and **analysis** of the learned models.

Modeling: The first step to integrating prosodic information in a speech understanding pipeline is to develop an efficient method of automatic **prosodic event detection**. The method should be efficient and readily adapted for various research purposes. We use a method based on a **convolutional neural network** (CNN) that requires little preprocessing and detects pitch accents and intonational phrase boundaries in the frame-based acoustic input. We test the prosodic event detector on several English and German corpora. This model can be applied to both languages since they have similar prosody and since the model does not rely on lexical information. We focus primarily on the binary classification task of prosodic event detection rather than the classification of several event types[1], since information on the presence of prosodic events on words is an important first step for understanding.

Applications: The second goal is to investigate the impact and use of prosodic information in applications for **speech understanding**. We assume a pipeline setup in which prosodic events are automatically detected and used as features to improve text-based tasks, especially on ASR output, which can introduce harmful recognition errors. The experimental setups comprise research conditions that pose several challenges, for example the lack of annotated data, different formatting requirements and the generalizability of pre-trained models. The three applications chosen for this work are semantic slot filling and coreference resolution, tasks that benefit from the connection between prosody and information status, and silver-standard corpus annotation, which aims to overcome the lack of gold-standard resources. We compare and discuss the use of a pre-trained prosodic event detector based on neural networks and one based on machine learning with manual feature selection.

Analysis: While the first two goals placed more focus on the performance and usability of the prosodic event detector, the third goal in this thesis is to analyze what the neural network has learned. The neural network takes frame-based acoustic features as input and creates a feature representation that is not straightforward to interpret. Therefore, the question of what information is latent in this **output feature representation**

[1] Both tasks were investigated and compared by Rosenberg (2009).

merits further exploration. We employ a simple method of analyzing the learned feature representation, which can be easily extracted from the output of the network, and analyze how the features that the neural network has learned correlate with features representing acoustic, temporal and context information that a human developer may choose.

1.4 Overview and contributions

We first provide the necessary linguistic and technical background for this thesis in Section 2, as well as give an overview of related work. The main contribution of this thesis is the systematic investigation of the use of a CNN for prosodic event detection, described in Section 3. The method was first introduced in Stehwien and Vu (2017b). This paper demonstrated the performance of the prosodic event detector in experiments that compared speaker-independent and speaker-dependent modeling as well as the classification of different event types, using two different feature sets. An important finding was the effect of adding context information and the outcome of applying a position indicator feature to improve the method's performance. The effects of adding lexical information in the form of word embeddings, as well as first cross-corpus experiments, were reported in Stehwien et al. (2018). We later investigated the role of temporal information and compared the CNN to a feed-forward neural network (FFNN) and a recurrent neural network (RNN) (Stehwien et al., in press).

Section 4 reports on experiments that test the benefit of adding automatically predicted prosodic information on downstream tasks. We focused on the potential of using automatically predicted pitch accents for slot filling on automatically recognized text in two publications. The first paper (Stehwien and Vu, 2016) was a corpus study on the slot filling benchmark dataset ATIS (Hemphill et al., 1990). Motivated by the findings in this study, we tested the use of symbolic pitch accent features in two neural slot filling methods in the second publication (Stehwien and Vu, 2017a).

We used the CNN-based prosodic event detector for two collaborative papers. The first paper tested the use of automatically labeled prosodic events for coreference resolution (Rösiger et al., 2017), and was written in a joint effort with Ina Rösiger and co-authored by Arndt Riester and Ngoc Thang Vu. The second paper brought several research groups within the SFB-732 collaborative research center[2] of the University of Stuttgart together and introduced a silver-standard corpus resource with several linguistic annotation layers (Schweitzer et al., 2018). For this publication, the CNN-based method was compared

[2]https://www.sfb732.uni-stuttgart.de/

to Antje Schweitzer's prosodic event detector (Schweitzer, 2010), which was the method we used in the slot filling experiments.

Section 5 deals with the analysis of the neural network output. The first paper (Stehwien et al., 2019) provided evidence of duration information in the learned feature representation of the CNN. In the second article (Stehwien et al., in press), we took a closer look at the acoustic and context information learned by the network. These two publications were advised and therefore co-authored by Ngoc Thang Vu and Antje Schweitzer. Finally, Section 6 concludes this work; here we summarize the key findings and discuss possible future directions.

2 Background

This section provides the necessary background for this thesis, starting with the basic foundations of prosody in Section 2.1. We first introduce the main concepts of prosody that constitute the linguistic background of this work. It is often pointed out in the prosodic literature that, since there have been different approaches to defining prosody that are not always agreed upon, many terms are not always used consistently (Taylor, 2009; Féry, 2017). This section therefore also aims to clarify the definitions and views on prosody taken in this thesis.

Section 2.2 motivates the use of prosody in speech and language technology by reviewing related work and the state-of-the-art at the time of writing. As this thesis deals with the use of prosody for speech *understanding*, this section provides an overview of notable research on integrating prosody into relevant applications.

We introduce the task of prosodic event detection in Section 2.3, where we describe typical methods as well as an overview of related work. This section serves as the background for the method of prosodic event detection presented in Section 3 and discusses the comparability of different approaches.

As the method of prosodic event detection chosen for this work is based on a neural network, we also give a introduction to the basic concepts of deep learning and the neural network architectures used in this thesis in Section 2.4. We also briefly discuss the issue of neural network explainability that motivates our analysis in Section 5.

2.1 Linguistic background: Prosody

2.1.1 Definitions

Prosody is a property of spoken language that refers to patterns in pitch, loudness, timing and pauses. A straightforward description of prosody is the use of speech melody and rhythm to vary *how* something is said. *What* is said, in contrast, is usually the part of language that can be expressed through text. It is common to distinguish these

two levels of language as the *segmental* and the *suprasegmental* tiers, with phones (i.e. segments) comprising the former and the latter consisting of prosodic patterns spanning several segments. *Intonation* is defined more narrowly as the use of pitch (or *tune*) to convey meaning at the sentence level (Ladd, 1996).

Prosody plays a different role for meaning in different languages, ranging from a distinct lexical role in tone languages such as Mandarin Chinese or Vietnamese, to a partly lexical role in pitch accent languages such as Swedish, to no lexical role at all in, for example, English. It is the latter, the so-called *intonation languages*, that use prosody solely as a means to convey additional information at the phrase level through variations in prosodic patterns. These patterns vary considerably and are often optional, but not arbitrary (Peters, 2014), since they tend to follow typical grammaticalized structures which are extensively analyzed in the literature. The languages taken into account in this thesis are English and German, which belong to the West-Germanic language family (this also includes Dutch) and which are very similar in their use of prosody.

The definition of prosody often also encompasses *paralinguistic*, affective or non-linguistic aspects of speech, such as the effects of emotion and speaker state on the acoustic characteristics of language. Speakers may also express their opinion on what is being said (e.g. doubt; or by using sarcasm) by means of prosodic cues. The paralinguistic aspect of prosody is considered to be less language-specific (Taylor, 2009), since factors such as emotion, arousal, level of interest, or the physiological state of the speaker all have an unconscious effect on the voice. At the computational level, emotion recognition and affective computing are attracting increasing interest (e.g. Interspeech Computational Paralinguistics Challenge[1]). While the automatic processing of prosody plays an important role in these research areas, this thesis does not deal with the paralinguistic relevance of prosody.

The view of prosody taken in this work is concerned with its role as a *linguistic* means to convey meaning. It it therefore similar to the following definition of intonation given by Ladd (1996, p. 1): "Intonation conveys meanings that are applied to phrases or utterances as a whole, such as sentence type or speech act, or focus and information structure." Additional acoustic cues other than intonation, such as temporal cues and loudness, also contribute to prosody and are reviewed in the following.

[1] www.compare.openaudio.eu

2.1.2 Main acoustic correlates

According to the source-filter model of the vocal tract (Fant, 1960), the speech signal is produced by a glottal source signal (i.e. the voice) which is modified by the vocal tract filter (i.e. articulation). Prosody, therefore, is a property of the source signal. The most important acoustic cues of prosody that are connected its linguistic function are fundamental frequency, intensity and duration (Ladd, 1996).

The **fundamental frequency** (F0) describes the glottal source signal. Its perceptual correlate, *pitch*, is often used synonymously. F0 is produced by the vibrations of the vocal folds and can be consciously modified by the speaker through muscle tension. F0 is measured in hertz (Hz) and refers to the number of times per second a cycle of vocal fold opening and closing is completed. The range of an individual speaker's F0 is constrained by the physical characteristics of the vocal tract. Male speakers produce an F0 range of around 80-200 Hz, female speakers around 150-300 Hz, and children produce an even higher F0 range of 200-500 Hz (Clark and Yallop, 1995). One of the most common methods of estimating F0 is using the autocorrelation method for pitch tracking, for example as described in Taylor (2009).

Intensity is a time-domain property, computed using the amplitude of the speech signal as the magnitude of sound pressure variation in the signal (Clark and Yallop, 1995). It is modified by the subglottal air pressure with which a sound is produced and its perceptual correlate, *loudness*, is measured in decibel (dB).

In a natural-sounding utterance, F0 is varied over time, which leads to a perceived "melody". Intensity is also varied to place more emphasis on certain parts of the utterance. As speech is produced, the air is expelled gradually, which leads to a natural downdrift. This can be measured as a decrease in loudness, pitch and energy towards the end of a phrase.

Duration, as well as timing and **pauses**, are the cues that make up the speech "rhythm". Pauses are obvious cues to prosodic phrasing, as groups of words are often separated by breaks or short stretches of silence. Vassière observed that while pauses are natural occurrences in continuous speech when a speaker inhales, they are often placed strategically to create rhythmically fluent speech and that the "majority of pauses [...] are located at grammatical junctures" (Vassière, 1988, p. 82). The duration of phones and syllables is often lengthened when they are more prominent compared to their usual form in the same context. Stressed syllables are also usually longer than unstressed ones. A frequently observed effect in many European languages is pre-boundary lengthening: segments tend to have a longer duration, that is, spoken more slowly, before a prosodic

phrase ends (Klatt, 1976; Wightman et al., 1992).

While the aforementioned correlates can be measured in absolute terms, their contribution to prosody lies in the changes that make up the suprasegmental pattern of an utterance. It is therefore necessary to regard all acoustic correlates relative to their surrounding context, and relative within the acoustic range of the speaker. We describe how this information is represented for machine learning in Section 2.3.3.

2.1.3 Prosodic system of intonation languages

An established way of describing English and German suprasegmental patterns is by means of *prominence* and *phrasing*. Prominent syllables and words are perceived as more salient relative to their context, and the prosodic grouping of words is referred to as phrasing.

Lexical stress refers to the prominence of a syllable relative to the other syllables in a word. The acoustic correlates of lexical stress are similar to those of prominence[2], however, prominence makes a syllable or word more globally salient across a phrase, while lexical stress is a more "local" concept defined for individual words. While lexical stress is inherently linked to the prosodic pattern of an utterance, it plays an entirely different role for meaning. It is part of the correct pronunciation of multi-syllabic words in English and German, and is therefore a fixed part of the lexicon. For this reason, some words in English differ only in their stress pattern (e.g. *perMIT* is a verb, while *PERmit* is a noun). Usually there is one syllable with primary stress for each word; and longer words can also have syllables with secondary stress. Acoustically, the presence or absence of stress affects mainly the vowel of the stressed syllable, resulting in modified length and formants of the nucleus.

At the phrase level, the majority of the phonological literature analyzes the patterns created by prominence and phrasing using **prosodic events**. Prosodic events are linguistically meaningful (phonological) units of prosody that are characterized by the aforementioned acoustic cues that make them stand out from the surrounding context and therefore salient. Pierrehumbert's **tone-sequence model** (Pierrehumbert, 1980) defines well-formed intonational patterns in English as sequences of prosodic events. The term *tone* in this case refers to a specific pitch shape of the two main events: *pitch accents* and *phrase boundary tones*.

Phrase boundaries mark the edges of intonation phrases. The tone-sequence model

[2]excluding F0 according to Sluijter and van Heuven (1996)

describes utterances as consisting of one or several intonation phrases which group words into rhythmically or syntactically meaningful phrases. Intonation phrases are made up of intermediate phrases that are delimited by phrase accents; these are less salient than intonational phrase boundaries. Typically, a pause and a reset of pitch and energy will immediately follow a boundary tone.

Pitch accents are perceived events in the prosodic pattern that mark prominent syllables or words. Acoustically, pitch accents are peaks in the F0 contour which also correlate with increased intensity and energy, as well as a lengthening of the duration of syllables and phones. Consequently, a pitch accented word sounds more prominent relative to its surrounding context at the phrase or sentence level. Only certain salient words that the speaker deliberately emphasizes via a pitch accent on the stressed syllable are perceived as prominent. Thus, a pitch accent will always lie on a stressed syllable, but not every stressed syllable will carry a pitch accent. A *nuclear* accent is the most prominent pitch accent in a phrase.

The **Tones and Break Indices** system (Silverman et al., 1992), short ToBI, was created as an annotation scheme based on the tone-sequence model. ToBI defines the following tones (or event types): High (H) and low (L) tones describe both pitch accents (*) and phrase boundaries (%). Events can be simple or monotonal (e.g. H*) or complex and bitonal (e.g. L-H*). The asterisk denotes the accent tone that is aligned with the stressed syllable. The combination of an *intermediate boundary* (also referred to as a phrase accent and denoted by "-") and a phrase boundary make up an *intonational* phrase boundary (H-L%). The ToBI break indices (0-5) refer to the perceived strength of a break at the end of intermediate and intonational phrases.

Two example phrases, originally taken from Pierrehumbert (1980) and reprinted by Pierrehumbert and Hirschberg (1990), are given in Figures 2.1 and 2.2. In the first example, the phrase *Legumes are a good source of VITAMINS* has a high pitch accent (H*) on the last content word *vitamins*. This is a common pitch accent pattern in English (Ladd, 1996). The phrase ends in a falling-rising boundary tone (L-H%). In the second example, the same word carries the pitch accent, but with a low tone (L*) and the phrase boundary is a high rising tone (H-H%). We discuss how the event placement and tonal shapes affect the meaning of the utterance in the next section.

The advantage of using ToBI is that it has enabled the labeling of corpora, which has helped advance corpus-based research in linguistics and the development of technological applications (Hirschberg, 2002). Since the original ToBI system was developed for Standard American English, several adaptations have been created for other languages with

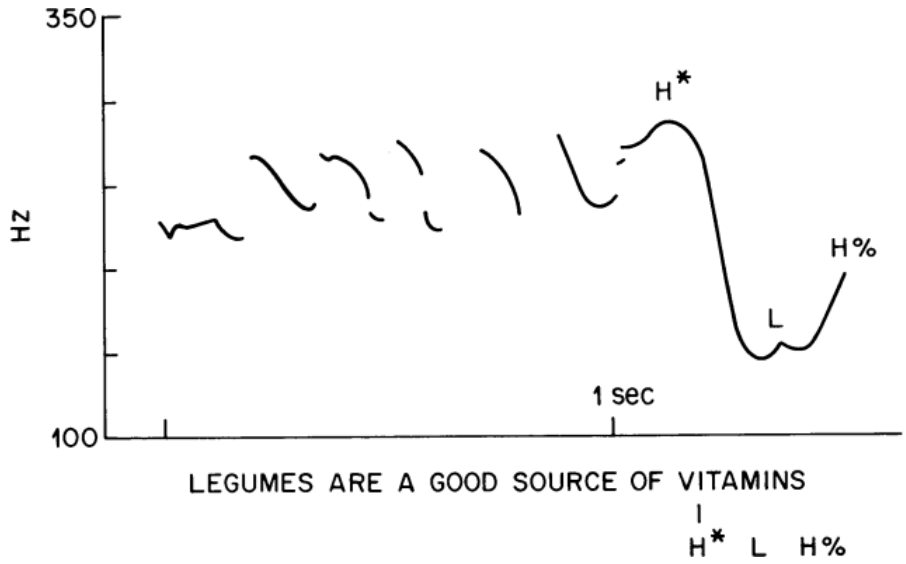


Figure 2.1: Pitch contour of a declarative statement (originally from Pierrehumbert (1980) and re-printed by Selkirk (1995)) with a pitch accent on *vitamins*.

similar prosody. For this thesis, the German GToBI (Grice et al., 2005) and GToBI(S) (Mayer, 1995) systems are the most relevant. The adaptations are necessary since the inventory of tone types can differ across languages. For example, the German ToBI inventory includes a H+L* pitch accent that does not exist in English.

The concepts introduced in this section arose from the perspective of phonology. This means that not only are prosodic events (i.e. pitch accents and phrase boundaries) acoustically relevant, but they play an important role for the interpretation and meaning of utterances. The next section lists some examples of the relationship between prosody, especially prosodic events, and meaning in English and German.

2.1.4 Prosody and meaning

Prosody conveys important cues to the meaning of speech utterances. It is relevant both at the sentence and discourse level and is connected to syntax, semantics and pragmatics.

At the **sentence level**, different intonational contour shapes have been connected to different meanings (Hirschberg, 2002). There exist many cases where the same sentence will have different meanings depending on how prosodic events are placed and what tones are used. We first consider the **sentence modality** of the two example sentences in Figures 2.1 and 2.2. The first sentence (*Legumes are a good source of vitamins*) is a declarative statement that differs slightly from typical declarative utterances since it ends with a fall-rise contour instead of a simple low boundary tone (L%). Selkirk (1995)

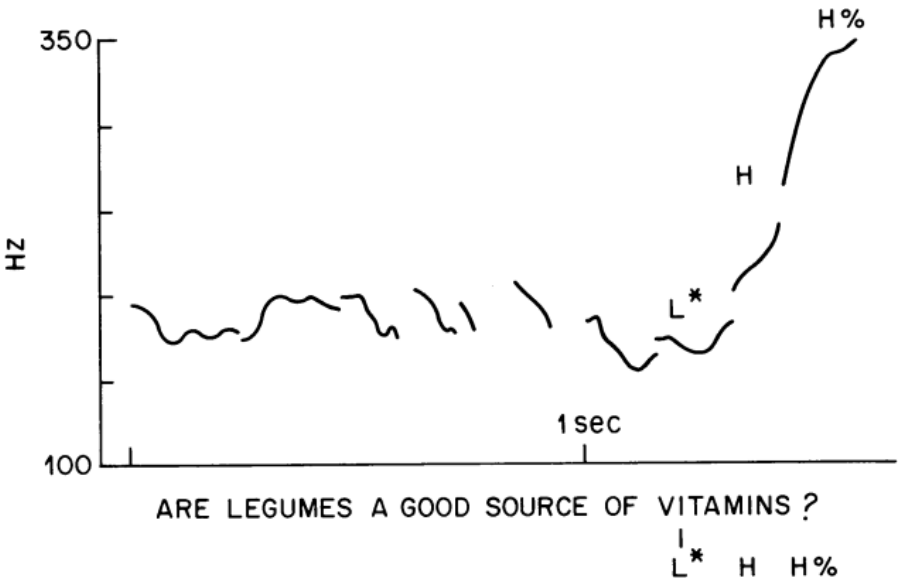


Figure 2.2: Pitch contour of a interrogative statement or question (originally from Pierrehumbert (1980) and re-printed by Pierrehumbert and Hirschberg (1990)) with a pitch accent on *vitamins*.

points out that this makes it appropriate only in certain contexts, for example as a contradiction to a previous statement such as *Legumes aren't good for anything.* The second sentence (*Are legumes a good source of vitamins?*) ends in a high rising boundary tone which is one of the most salient cues to questions or interrogative statements. In fact, even if the two example sentences had the same lexical content, the phrase *Legumes are a good source of vitamins* could be turned into a question by using the same type of contour.

Phrase boundaries can also form **continuation rises**, which convey to the listener that there is more information to come. This is often found in lists, such as in *We bought apples, tomatoes and oranges*, as an L-H% tone (Hirschberg, 2002). In this case, the speaker uses rises to mark that short pauses after *apples* and *tomatoes* do not denote the end of the sentence. In contrast, a falling boundary tone would occur at the end of *oranges*.

Phrase boundaries are often placed at syntactic boundaries, where they can serve as helpful cues to the intended meaning of ambiguous sentences. The well-known problem of **PP-attachment**, for example, is illustrated by the sentence *The spy saw the cop with binoculars* (Nicol, 1996). The prepositional phrase (PP) *with binoculars* can be attached to the verb phrase, resulting in the interpretation that the spy used the binoculars, or it can be attached as a modifier to the noun phrase *the cop*.

This relationship between prosodic phrasing and meaning also applies to German (Féry, 1993). The following example by Hess et al. (1996) shows how German utterances with identical lexical content differ considerably just by marking of syntactic boundaries and sentence modality through prosodic phrasing:

(1) Ja, zur Not geht's auch am Samstag.
(Yes, if necessary it will also be possible on Saturday.)

(2) Ja, zur Not. Geht's auch am Samstag?
(Yes, if necessary. Will it also be possible on Saturday?)

Example (1) would have a continuation rise and a pause after *Ja* and a falling boundary tone on *Samstag*. Example (2) would also have a continuation rise after *Ja*, but with a shorter pause and a falling boundary tone on *Not* to mark the end of the phrase. The second sentence would end in a rising boundary tone to mark a question.

A well-researched role of prosody at the **discourse level** is its relationship with **information structure** (Féry, 2017). For example, pitch accents are used to mark discourse entities with **focus**, and deaccenting is used to mark **givenness**. In Figures 2.1 and 2.2, the pitch accent lies on *vitamins*, which marks this word as focused. Furthermore, this also implies that the word is *new* information, that is, not previously mentioned in the discourse or salient due to the context. If the previous statement was *Nothing in this cupboard is a good source of vitamins*, then the term *vitamins* would be given and hence deaccented, while *legumes* would be focused, new, and accented (Selkirk, 1995).

Anaphors and coreferent noun phrases are also often deaccented, since they are the subsequent and thus *given* mentions of *new* antecedents. The antecedent is the noun phrase that is mentioned first, and the anaphor is, for example, a pronoun referring to the same entity (both are coreferent). The example *MARY saw a LEMON PIE and then she decided to EAT it* used by Féry (2017) illustrates the deaccenting of anaphors (*lemon pie, it*) as well as the highlighting of new information through the pitch accent on *eat*.

Hirschberg (2002) noted that the deaccenting of givenness is a generalization, since there are many exceptions to the rule, for example as reviewed in Riester and Piontek (2015). This is the case for most aspects of the relationship between prosody and meaning. In general, prosody can be said to be variable, highly context- and speaker dependent and that it delivers many different cues that can contribute to meaning. The aforementioned cases are only a few of the more common examples and therefore this section is by no means exhaustive. The examples in this section were chosen to highlight

some of the more straightforward linguistic cases of prosodic variation. The connection between prosody and information structure is especially relevant for this thesis. The next section provides an overview of notable studies on how prosody can be used for applications surrounding the automatic understanding of speech and language.

2.2 Prosody for speech understanding

2.2.1 Overview

The previous section has shown that prosody is connected to several levels of linguistic analysis. It therefore seems natural that these relationships should be exploited in language technology applications that target these linguistic levels. We will show in this section that there is a substantial amount of research guided by this notion. Figure 2.3 sketches an overview of how prosody relates to other areas of linguistics[3] and how these connections motivate the use of prosody in many different tasks in the field of speech and language processing.

The literature over the last decades has discussed may ideas and possible applications (Hirschberg, 2002; Batliner et al., 2001b; Shriberg, 2005). In a review article, Hirschberg (2002) discussed the state of linguistic research and technological applications with regards to integrating prosody into various aspects of spoken dialogue systems in the early 2000s. Hirschberg (2002) reviews prospects and challenges of how information derived from prominence (such as givenness and focus) and prosodic phrasing (such as chunking, scope and syntactic disambiguation) can be integrated into speech technology. She observed that while prosodic modeling is already a key component of text-to-speech synthesis, applications for speech understanding do not consistently make use of prosody when processing syntax, semantics, pragmatics, discourse structure and dialogue. One of the key challenges is that while there is undoubtedly a connection between prosody and various levels of meaning, prosody remains quite variable and therefore, "research on prosody [...] is more a matter of finding likelihoods" (Hirschberg, 2002, p. 32). For example, Hasegawa-Johnson et al. (2005) pointed out that prosodic and syntactic structure are not isomorphic, and Hirschberg (2002) notes that there are several ways to convey focus other than through the use of prosodic cues.

Despite this, or perhaps because of these challenges, prosody modeling is an interesting field of research that has attracted much attention. Among the several possible applica-

[3] We also refer to Shattuck-Hufnagel and Turk (1996, p. 237) for another view of the role of prosody in language grammar.

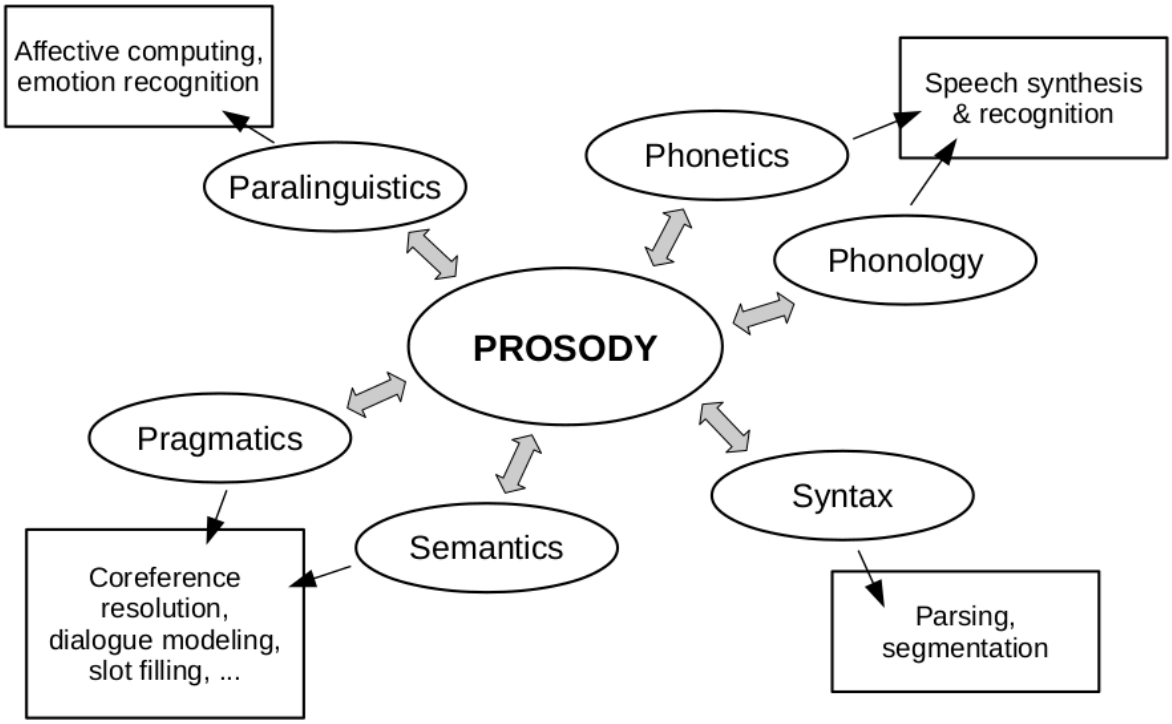


Figure 2.3: Overview of the role of prosody for various levels of linguistic analysis (shown in ellipses) and applications in speech and language processing (shown in boxes).

tions that can benefit from exploiting the connection between prosody and meaning in English and German, this section focuses on the larger research areas in which a number of studies have investigated the impact of prosody: automatic speech recognition (ASR), dialogue systems, speech segmentation and parsing. The following individual studies have shown that prosodic information can help text-based NLP tasks when extended to speech: Katerenchuck and Rosenberg (2014) used automatically predicted ToBI labels for named entity recognition on ASR output. Rösiger and Riester (2015) showed that binary prosodic event labels can improve coreference resolution on German radio news data. Initial experiments on the use of prosodic features for sentiment analysis were presented by Mairesse et al. (2012) and Morency et al. (2011). In this thesis, we also present experiments on coreference resolution as well as slot filling (Section 4). Other obvious uses such as the modeling of prosody for text-to-speech synthesis (TTS) or the use of prosodic cues in affective computing are mentioned in Figure 2.3 for completion, but are not directly relevant to the topic of this thesis.

2.2.2 Automatic speech recognition

Early studies on the use of prosody for speech recognition and understanding were carried out by Waibel (1988) and Vassière (1988). Their aim was to use acoustic-prosodic information to directly improve the accuracy of the recognition at the lexical level, that is, to improve the word-error rate (WER). Here, the additional cues can, for example, help disambiguate sequences involving compound nouns in German, e.g. *zwei Räder* vs. *Zweiräder* (Hess et al., 1996). Both lexical stress and prosodic prominence affect the pronunciation and acoustic properties of a phoneme, as pointed out by Hasegawa-Johnson et al. (2005). For example, English vowels without lexical stress are often reduced to a schwa. However, lexical stress is more deterministic than pitch accent and can therefore be modeled by a pronunciation dictionary. Pitch accents and phrasing can lead to additional phonetic variation (Shafran et al., 2001) which may have an effect on acoustic modeling.

Ostendorf et al. (2003) discussed several approaches to integrating prosody directly into acoustic models of ASR systems. Their experiments focused mainly on including prosody in either the acoustic or the pronunciation models. Hasegawa-Johnson et al. (2005) combined ASR and prosody recognition in a sophisticated dynamic Bayesian network architecture. They integrated prosodic information by modifying both the acoustic and language models to depend on recognized pitch accents and phrase boundaries. An alternative method of directly modifying the ASR models is to use prosody for rescoring word hypotheses, as proposed by Ananthakrishnan and Narayanan (2007) and Jeon et al. (2011). The idea of rescoring word hypotheses was also adopted by Vicsi and Szaszák (2010). Hirschberg et al. (2004) and Goldwater et al. (2010) investigated how prosody impacts word recognition in automatic systems, showing that there are prosodic cues to where words and phrases may be recognized incorrectly.

These studies show that prosody indeed plays an important potential role for ASR. However, there appears to be little drive to actually incorporate this knowledge into ASR systems. This may be because of the redundancy of prosodic information in ASR, as noted by Vassière (1988). Furthermore, state-of-the-art systems do not rely on prosody to produce high-quality text output. Therefore, in the pipeline setting assumed in this thesis (shown in Figure 1.3), ASR is treated as a separate module before the prosody recognition step.

Waibel (1988) distinguished three levels of processing in which prosodic information can be integrated. Apart from the lexical level, the next two levels in his work are not directly related to the task of ASR: At the syntactic level prosody can help reduce the

search space for parsing. An example given for the semantic/pragmatic level is speaker intent recognition. In the context of this thesis, the syntactic and semantic/pragmatic levels are most relevant, since these are the levels at which the downstream tasks are situated in a speech understanding pipeline. The remainder of this section reviews work on using prosody for these tasks.

2.2.3 Dialogue understanding

An example of a more complex application for which prosody can provide important information is a spoken dialogue system. In situational dialogue, prosodic cues can be used as turn-taking cues (Peters, 2014; Hirschberg, 2002). The end of a speaker turn is often indicated by final lowering and a narrowing of pitch range (Hirschberg, 2002). Dialog acts (and speech acts) can also correlate with prosodic patterns. For example, a system that processes machine-directed speech could use prosodic cues to detect when a speaker is requesting information by formulating a question (Nöth et al., 2002; Shriberg and Stolcke, 2004). When processing spontaneous spoken dialogue, prosodic features can help detect backchannels, hesitations and disfluencies (Nöth et al., 2002; Shriberg and Stolcke, 2004).

The VERBMOBIL project[4] (Wahlster, 2000) was a German research project that ran from the early 90s to the early 2000s. The result of this project was a speech-to-speech translation and understanding system for English, German and Japanese. The VERBMOBIL system consisted of several modules that were developed to perform online processing of spoken utterances. One of the notable contributions that this project made was the integration of prosody in all components of the system (Hess et al., 1996). Similar to Waibels's idea, this project recognized that prosody can help disambiguate the meaning of utterances at multiple levels. The main areas of investigation were word recognition (Nöth, 1991), parsing, punctuation insertion and segmentation as well as dialogue act classification (Kompe, 1997; Nöth et al., 2002) and sentence modality (Strom, 1995). VERBMOBIL also contained a repair module that used prosody to identify whether a boundary is inserted as a syntactic boundary or as a hesitation before a correction. The system also used pitch accent information to determine the most salient, (i.e. focused) regions in speech, motivated by the fact that listeners tend to pick up on salient regions of speech and ignore non-salient ones. In order to exploit more specific information on the meaning of utterances, some experiments also dealt with the

[4] www.verbmobil.dfki.de

recognition of focal and contrastive accents (Hess et al., 1996; Haas et al., 1995). Nöth et al. (2002) also reported how prosody was helpful in locating date and time expressions for the integrated information retrieval system.

2.2.4 Segmentation

Several studies have investigated the automatic segmentation of speech data based on prosodic cues, mainly prosodic phrase boundaries. Automatic segmentation encompasses both sentence and topic segmentation, and can be used, for example, to break up long stretches of speech before parsing. Hirschberg and Nakatani (1998) suggested that prosodic information can be exploited for topic segmentation. Stolcke et al. (1998) modeled sentence boundaries along with disfluency detection on ASR output using both lexical and prosodic information. Shriberg et al. (2000) also used prosodic modeling to detect the boundaries of sentences and topics. The general approach taken in these studies was to combine the prosodic information with a lexical language model. Rosenberg (2009) also combined lexical information and automatically predicted phrase boundaries to help segment individual stories in broadcast news speech. Other studies (Shriberg et al., 1997; Ferrer et al., 2003) have shown that prosody alone can aid the automatic detection of utterance boundaries from speech, without the need for performing speech recognition.

A related task to speech segmentation is punctuation insertion. Automatically inserting punctuation symbols is highly relevant as a post-processing step to ASR, since standard systems do not include this as a component. Prosody is an important cue to different types of punctuation: a declarative statement ending with a period (".") tends to end in a falling boundary tone, a comma may be preceded by a continuation rise, and an exclamation mark ("!") is expected to occur in regions of higher arousal or loudness. Similar to the sentence segmentation methods listed above, Cho et al. (2015) combined prosodic and lexical cues in a language model for automatic punctuation insertion.

2.2.5 Syntactic parsing

Price et al. (1991) showed how prosody helps to disambiguate ambiguous sentences (involving, for example, PP-attachment and parenthetical clauses) in perceptual experiments. Nicol (1996) provided an overview of how syntactic parsers could benefit from prosodic information. The main disambiguation problems she discusses are similar to the examples given in Section 2.1.4, where the word string, without any further information

on punctuation or prosody, can result in different parse trees. The prosodic information primarily associated with these syntactic elements are the placement of prosodic phrase boundaries.

(Veilleux and Ostendorf, 1993) investigated the use of prosody for its use in resolving syntactic ambiguities in the Airline Travel Information Systems (ATIS) corpus (Hemphill et al., 1990). In this study, the authors used prosodic break scores as well as binary indicators of prominence to re-rank recognition hypotheses according to a joint syntactic and prosodic probability score. The output was found to improve the downstream processing in the ATIS benchmark task.

Parsing a well-researched task in natural language processing (NLP), but methods optimized for written language usually perform much worse on spoken language, especially on spontaneous speech. Based on this motivation, Kahn et al. (2005) used prosodic boundary information to improve constituency parsing for spontaneous speech. They implemented a method of dealing with disfluencies and speech repairs prevalent in spontaneous speech by detecting them based on prosodic cues and by removing false starts that should not be taken into account by the parser. By using the probabilities of detected phrase boundaries along with syntactic features in a decision-tree-based method of re-ranking candidate parses, they were able to improve the performance of the system.

A more recent approach to using prosody for parsing was presented by Tran et al. (2018). This paper describes how both lexical and acoustic information can be jointly modeled for neural-network-based constituency parsing. They proposed an architecture to train both a lexical model (based on word embeddings) and an acoustic-prosodic model based on cues to prosodic phrase boundaries. The authors were able to improve the performance of the parser over the baseline system and showed cases where the joint model correctly parsed ambiguous cases of PP-attachment.

While the aforementioned paper proposed a joint modeling technique that skips over the symbolic prediction step, most methods adopt an approach similar to the pipeline setting we assume in Figure 1.3, that is involving a separate system for prosodic event detection. We describe this task in the following section.

2.3 Prosodic event detection

Many approaches to modeling prosody in speech technology use the tone-sequence model as a theoretical basis and therefore model prosody as prosodic events. Prosodic event detection is the task of automatically detecting these events, namely pitch accents and

phrase boundary tones. In this thesis, it is the core element of a modular setup for incorporating prosody into speech understanding systems. The ToBI annotation scheme has defined a standard that many researchers can agree on, and therefore made it possible to annotate large amounts of data. Although the ToBI standard still poses challenges and shortcomings (Wightman, 2002), it has been used fairly consistently to annotate speech corpora. The availability of prosodically annotated data makes it possible to apply machine learning to solve prosodic event detection.

While the automatic recognition and labeling of prosodic events has attracted much attention, there is no official benchmark or task description. Thus, the scientific literature displays a wide range of methods. The approach adopted in this work similar to that taken by Rosenberg (2009). In the following, we define the task of prosodic event detection for this thesis and provide an overview of a typical methods and related work.

2.3.1 Task definition

Prosodic event *recognition*, which processes speech data, needs to be set apart from prosodic event *prediction*, which is usually done from text for TTS (Taylor, 2009). The task usually consists of several stages: data collection and preprocessing, feature extraction, and a machine learning method (Wightman and Ostendorf, 1994). Methods are often specifically tailored to the purpose they are used for; from automatic corpus labeling (Rosenberg, 2010a; Elvira-García et al., 2016) to facilitating large-scale statistical linguistic analyses (Schweitzer, 2010) to providing features for other tasks.

Prosodic event recognition distinguishes two subtasks: **detection** typically refers to the binary classification task of recognizing the presence or absence of a prosodic event, while **prosodic event classification** encompasses the full multi-class labeling of prosodic event types (Rosenberg and Hirschberg, 2009). These types can be ToBI labels or more coarse-grained classes, e.g. Chen et al. (2004). Prosodic event type classification is often desired for linguistic analysis and annotation. A few systems have been developed specifically to perform prosodic event classification for annotation purposes: The AuToBI system (Rosenberg, 2010a) labels a range of ToBI pitch accent and phrase boundary types. Elvira-García et al. (2016) developed a system for labeling Spanish and Catalan. Many applications in speech understanding, however, already benefit from knowing where prominent words or prosodic boundaries are located, e.g. coreference resolution (Rösiger and Riester, 2015). In this case, prosodic event detection can be sufficient.

Typically the recognition of pitch accents is modeled separately from phrase bound-

aries, as in (Rosenberg, 2009; Schweitzer, 2010), although some systems label both jointly (Wightman and Ostendorf, 1994; Chen et al., 2004; Kumar et al., 2008). Prosodic events can be defined at the **word level** or the **syllable level**. Modeling at the word-level is the easier task, since it is easier to label an entire word as carrying an event than locating the specific syllable with which it is aligned. Furthermore, prosodic events rarely span a single syllable (Rosenberg, 2009). For a further discussion on the advantages of detecting prosodic events on words, we refer to the paper by Rosenberg and Hirschberg (2009).

Prosodic event detection at the syllable level is closely related to **lexical stress detection**. Both tasks can be approached in a similar manner and rely on similar acoustic cues (Tepperman and Narayanan, 2005; Shahin et al., 2016). Tamburini (2003) used lexical stress detection as a preprocessing step before pitch accent detection. The advantage of this method is that it limits the amount of instances that need to be labeled since only stressed syllables can carry a pitch accent.

2.3.2 Data

Prosodic event recognition is usually a data-driven task that uses both speech and text as input and outputs symbolic prosodic event labels. As a (usually supervised) statistical learning task, automatic prosodic event detection needs to be trained on annotated speech corpora.

The **annotation** of gold-standard prosodic labels is done exclusively by hand. This is an expensive and time-consuming task; and human labelers require expert knowledge and annotation experience. Most of the prosodically annotated corpora available for English and German are annotated according to the ToBI standard or other label inventories based thereon (Ostendorf et al., 1995; Eckart et al., 2012).

While prosodic event recognition uses information primarily extracted from the acoustic signal, it often also takes text-based information into account. This is possible when the **transcriptions** are already given or are recognized by an ASR system.

Most methods, whether they use text-based information or not, rely on knowledge about the location of word boundaries.[5] **Force-alignment** (also referred to as *segmentation*) is therefore the first step. Speech datasets often contain manual orthographic

[5]There is a limited amount of research to detecting prosodic events from the acoustic signal only, without any access to word boundary information (Strom and Widera, 1996; Hasegawa-Johnson et al., 2005). These publications show mixed results: Strom and Widera (1996) were not successful, and Hasegawa-Johnson et al. (2005) obtained good results but on limited data.

transcriptions but are segmented via force-alignment, for example the DIRNDL corpus Eckart et al. (2012). The task of automatic segmentation is similar to ASR, except that the lexical content is given (Rapp, 1995). In this task, phone, syllable and word boundaries are aligned to timestamps in the speech signal. There can be several "degrees" of automation for this process: Using an ASR system, the speech signal can be automatically recognized and segmented in one step. This procedure has the advantage that no manual work is necessary and it is therefore an efficient method of segmenting large amounts of data. The drawback is that potential recognition errors must be taken into account. However, the quality of the output is usually acceptable and it has become a standard tool in phonetic research and speech technology (Schiel, 1999; McAuliffe et al., 2017).

2.3.3 Features

For any machine learning method, the data must be represented as *features*, that is vectors of either binary or continuous values that each represent different relevant attributes of the input. These attributes typically consist of known correlates of prosodic events. While the exact choice of features varies in the literature, usually a set of acoustic and temporal features is extracted from the speech data, and additional lexical or lexico-syntactic features can be extracted from the text. Some examples of frequently used features are given in the following.

Depending on the method used, acoustic features can be represented at different levels. The most "low-level" information used for prosodic event recognition are simply **frame-based** representations of the speech signal, for example log-Mel frequency bands (Tran et al., 2018; Shahin et al., 2016). Slightly more informative frame-based representations, referred to as *descriptors*, can also be computed using signal processing tools, for example F0 and energy or amplitude-based features such as intensity. Frame-based features can be extracted directly from the speech signal without any reference to syllable or word boundaries. Since many methods of prosodic event recognition use time-aligned speech data, it is also common to use the boundary information to compute phone, syllable, or word-level acoustic features. Features that represent an entire unit (i.e. syllable or word) are also referred to as **aggregated** features (Rosenberg, 2009). These are often computed by taking, for example, the maximum, minimum or mean of an acoustic descriptor across a word or a syllable (Kompe, 1997; Rosenberg and Hirschberg, 2009; Jeon and Liu, 2012).

The fundamental frequency **F0**, as one of the most important acoustic cues of prosody,

is used as a feature in almost all published studies on prosodic event recognition. In addition to frame-based or aggregated representations, some studies have used higher-level features based on F0. For example, Schweitzer (2010) used PaIntE parameters (Möhler and Conkie, 1998) for syllable-based prosodic event classification. Wang et al. (2016) computed wavelet representations for each frame as features for a neural network. Despite its widespread use, the utility of F0 as a feature in automatic prosodic labeling has been questioned: Batliner et al. (2001a) stated that errors in automatic pitch extraction methods make F0 a comparably weak feature. Hasegawa-Johnson et al. (2005) also discussed the effect of estimation errors and applied optimization techniques to obtain better F0 features.

Energy and **intensity** are used in the majority of related work, e.g. Kompe (1997); Levow (2005); Rosenberg et al. (2015). Rosenberg (2009) trained successful models using features representing energy and Rosenberg and Hirschberg (2007) were able to improve pitch accent detection performance by leveraging pitch information to correct energy measurements in order to obtain stronger features. Soto et al. (2013) found that especially phrase boundaries correlate with a decrease in intensity.

Zero-crossing rate, that is, the number of times the signal crosses the horizontal axis, was used by Hasegawa-Johnson et al. (2005). Other popular features are computed from the spectrum and include spectral tilt or emphasis (Rosenberg et al., 2015; Tamburini, 2003; Hasegawa-Johnson et al., 2005) and spectral balance (Rosenberg et al., 2012; Ren et al., 2004),

Since prosodic events are relatively local phenomena, it is helpful to include features that represent changes over time. These are often referred to as **delta** features. For example, Rosenberg et al. (2015) used delta features that describe changes in acoustic features across word boundaries in addition to aggregated features that describe single words.

If segmental and word-boundary information is given, it is straightforward to compute various **duration** features that capture prosodic effects such as speaking rate or pre-boundary lengthening. These can simply represent the duration of the entire word (Rosenberg and Hirschberg, 2007) or syllable (Wightman and Ostendorf, 1994; Sun, 2002), or more detailed information such as the duration of the syllable nucleus (Schweitzer, 2010). Batliner et al. (2001c) found that the correlation between word duration and lexical content was the main reason why it is a strong feature compared to other acoustic features. They reported that raw duration is more helpful than normalized duration, since it correlates with parts-of-speech: content words tend to be longer

and are more frequently pitch accented.

Prosodic patterns also include **pauses** and other regions of silence (e.g hesitations and disfluencies). Pauses are often inserted in between intonation phrases and therefore co-occur with phrase boundaries. Soto et al. (2013) found that pauses represented as a boolean value or by their duration, as in (Kompe, 1997), are a very strong features for detecting phrase boundaries, even across languages.

Since prosody is realized relative to a speakers own acoustic range and speaking rate, both acoustic and duration features are usually **normalized** (Rosenberg and Hirschberg, 2009; Schweitzer, 2010; Batliner et al., 2001c). An acoustic descriptor is normalized across all values measured for a speaker by subtracting the mean value and divide by the standard deviation across the speaker's range:

$$\text{z-score} = \frac{\text{value} - \mathit{mean}}{\mathit{stddev}} \tag{2.1}$$

The z-score indicates the number of standard deviations the value in question is away from the mean value of that specific feature.

The feature sets in related work are rarely limited to the representations of just the word or syllable in question, since prosody spans longer stretches of speech and since prosodic emphasis and events are realized relative to the acoustic context. Most studies use a **context window** around the current word (or syllable) and include feature representations of the neighboring as well as the current word. A typical window size is three to five words (or syllables), that is, one or two preceding and one or two following words in addition to the current word (or syllable) (Schweitzer, 2010; Rosenberg, 2009; Margolis et al., 2010). Levow (2005) investigated how much context before and after the current word yields the best improvements for pitch accent detection in English and Mandarin Chinese. She found that, for the chosen method, the preceding context is more important than the following context.

Depending on how much feature engineering and preprocessing is desired and feasible, more **higher-level** features that represent the phonological structure can be computed. Schweitzer (2010) for example, used knowledge about the syllable structure, including information on the syllable nucleus, the neighboring segments and whether the syllable is word-final. In this case, automatic methods of syllabification and syllable nucleus detection were used.

A significant body of work has shown that prosodic event recognition also benefits from the addition of **lexical** or **lexico-syntactic** information (Rosenberg, 2009; Schweitzer,

2010). Studies on the relationship between text and prosody in automatic detection methods date back to early research in the field (Pan and McKeown, 1999; Hirschberg, 1993). Thus, many methods rely on a combination of both acoustic and textual features (Wightman and Ostendorf, 1994; Jeon et al., 2011; Sun, 2002; Ananthakrishnan and Narayanan, 2008). Lexical information can take the form of part-of-speech (POS) tags (Schweitzer, 2010; Rosenberg et al., 2012) or POS classes (Batliner et al., 2001c). Yuan et al. (2005) and Nenkova et al. (2007) observed that information on how frequently a word is accented in the training data is a strong feature for text-based pitch accent prediction. Lexico-syntactic features tend to comprise more higher-level information, for example by using parse trees or supertags to exploit the correlation between the syntactic and prosodic structure of speech (Domínguez et al., 2016; Chen et al., 2004; Kompe, 1997; Kumar et al., 2008). Sridhar et al. (2008) and Nenkova et al. (2007) investigated how features representing information status such as givenness or focus affect the automatic recognition of pitch accents.

In contrast to the choice of features, which are usually quite similar across previous methods, the applied learning methods can vary considerably. In the next section we list examples that have been popular in related work.

2.3.4 Machine learning methods

The most common approach to prosodic event recognition is to use **supervised machine learning**. There are many different methods that have been employed in previous work, from simple logistic regression models (Rosenberg and Hirschberg, 2009) to support vector machines (Rosenberg, 2010a; Levow, 2005). Decision trees-based methods have been a popular choice, in part because of their interpretability and the fact that the most informative features can be easily tracked (Wightman and Ostendorf, 1994; Batliner et al., 2001a; Nenkova et al., 2007; Margolis et al., 2010). Schweitzer and Möbius (2009) tested various methods in the WEKA machine learning toolkit (Hall et al., 2009) and found random forests to be among the most powerful learning methods. Rosenberg and Hirschberg (2007) used majority voting classifiers and Sun (2002) obtained good results using ensemble methods.

The above methods have been effective in learning to detect and classify prosodic events from syllable- or word-level features and aggregated acoustic features. However, they require considerable feature engineering, and therefore much of the experimental focus has been on finding the best feature representations, e.g. Nenkova et al. (2007), or improving existing ones, e.g. Rosenberg and Hirschberg (2007).

Some of this effort can be delegated to the machine learning method itself by using **neural networks**. Neural networks are a type of supervised classification models that have recently become a standard machine learning method in speech and language processing. This is also the type of model that we use for prosodic event detection in this thesis. In the following, we describe related work.

An early example of using neural networks to locate pitch accents and phrase boundaries using frame-based features was presented by Taylor (1995). This method involved a **recurrent neural network (RNN)** and frame-based acoustic features such as F0 and energy. Hasegawa-Johnson et al. (2005) used time-delay RNNs. The network was used to detect pitch accents at the frame level without prior word or syllable boundary information, by using a target function that models the sonorant portion of every syllable. They also performed prosodic event detection as a sequence labeling task on word sequences. For this, they used a multilayer perceptron with aggregated acoustic and lexico-syntactic features. A more recent example of using RNNs for prominence detection was published by Rosenberg et al. (2015). The main motivation for this method was the fact that RNNs can model entire sequences of words. They therefore represented each word using aggregated acoustic and context features, and let the RNN decide how much context to model.

The network type primarily used in this thesis is a **convolutional neural network (CNN)**. CNNs have been used for prosodic modeling in the following studies: Shahin et al. (2016) used CNNs to detect lexical stress in English and Arabic. They combined a CNN that learns high-level feature representations from 27 frame-based Mel-spectral features with global (i.e. aggregated) F0, energy and duration features across syllables to perform lexical stress detection. Wang et al. (2016) used CNNs to detect word-level pitch accents and phrase boundaries for one speaker of BURNC and then used the learned feature vectors for speech synthesis. They trained the model on continuous wavelet transformations of F0 for the detection of pitch accents and phrase boundaries in a speaker-dependent task. Tran et al. (2018) used a CNN to model phrase boundaries using an approach that is in many ways comparable to the CNN-based modeling described in this thesis, but integrated this technique into a joint modeling system for syntactic parsing. They used frame-based pitch and energy features extracting with the Kaldi toolkit (Povey et al., 2011) and combined these with word duration and bucketed pause duration features. The best features and hyperparameter settings were determined by evaluating on the parsing task.

Recently, Li et al. (2018) used multi-distribution neural networks for lexical stress and

pitch accent detection of non-native English speech at the syllable level. The features used in their method were aggregated acoustic features computed across syllables as well as lexico-syntactic features.

The fact that prosodically annotated corpus resources tend to be sparse has motivated the use of **unsupervised** or semi-supervised methods. Jeon and Liu (2012) used two support vector machines to model acoustic and lexico-syntactic properties of prosodic events, respectively, and used a co-training method to enable learning from both labeled and unlabeled data. Fully unsupervised methods often employ clustering algorithms such as k-means, e.g. (Ananthakrishnan and Narayanan, 2006; Levow, 2006). Huang et al. (2008) used a generative mixture model to detect prosodic breaks in Mandarin.

Finally, Fernandez et al. (2017) proposed a method of overcoming the lack of annotated resources by using synthetic data. They artificially created more training data by performing voice transformations on existing speaker recordings and used the same RNN system as Rosenberg et al. (2015). With this method, they were able to reduce the classification error rate on most speakers in BURNC.

2.3.5 Evaluation and comparability

The performance of machine learning methods is measured using **evaluation metrics** such as *accuracy*, *F1-score*, *precision* and *recall*. Accuracy is simply the percentage of all datapoints that have been labeled correctly. The other scores take the rate of true and false positives (*tp*/*fp*) and true and false negatives (*tn*/*fn*) into account:

$$\text{F1-score} = \frac{2 \times (\text{precision} \times \text{recall})}{\text{precision} + \text{recall}} \tag{2.2}$$

$$\text{precision} = \frac{\text{tp}}{\text{tp} + \text{fp}} \tag{2.3}$$

$$\text{recall} = \frac{\text{tp}}{\text{tp} + \text{fn}} \tag{2.4}$$

The large range of different approaches and subtasks of prosodic event recognition renders the comparison of performance quite difficult. This issue has previously been pointed out in several publications (Sun, 2002; Nöth et al., 2002; Rosenberg, 2009). For example, the performance for pitch accent detection ranges from around 80% to 90% accuracy, depending on the employed method and evaluation procedure. Methods that combine acoustic and lexical features tend to yield better performance than those

that use just one of the two knowledge sources. Furthermore, some studies use data from one speaker, while others combine data from different speakers either within or across genders, or focus on speaker-independent modeling. This is an important point to consider when comparing the performance of different methods, since speakers can vary considerably in the way they produce and use prosody. It should also be noted that syllable-based and word-based methods cannot be directly compared.

The only corpus that could be considered a benchmark dataset is the Boston University Radio News Corpus (BURNC) (Ostendorf et al., 1995) because of its widespread use. For this reason, this section primarily lists the main results of related work that is comparable in terms of datasets, methods and experimental focus to the approach taken in this thesis. We also refer to the work by Rosenberg (2009) for an overview of methods and results for prosodic event recognition and an extensive literature review, as well as the overview in Jeon and Liu (2012).

The following publications evaluated **prosodic event detection** on BURNC: A frequently cited study was published by Sun (2002), in which an ensemble classifier was used to detect pitch accents at the **syllable level** on **one speaker** of BURNC. This method achieved 89.9% accuracy using **acoustic features**, which increased to 92.8% using **lexical features**. Extending this evaluation to several female speakers, Ren et al. (2004) obtained 83.6% for pitch accent detection using a time-delay RNN. Jeon and Liu (2009) used a neural network and performed cross-validation on the entire BURNC corpus. They reported 83.3% accuracy for syllable-based pitch accent detection and 91.2% for phrase boundary detection using an acoustic model. The performance increased to 89.9% for pitch accents and 93.3% for phrase boundaries after combining the acoustic model with a support vector machine based on lexico-syntactic features. Rosenberg and Hirschberg (2009) showed that the accuracy of their pitch accent detector, using acoustic features and a logistic regression classifier, increased from 81.9% when applied at the syllable level to 84.2% at the **word level**. They therefore conclude that word-level pitch accent detection is the better method. These numbers were obtained with 10-fold cross-validation on BURNC. Chen et al. (2004) performed **leave-one-speaker-out cross-validation** (which yields *speaker-independent* models), but modeled pitch accents and phrase boundaries jointly using an acoustic and syntactic model. Their model yielded 84.2% accuracy for pitch accent detection at the word level and around 93% for phrase boundary detection. Using a convolutional neural network and purely acoustic features, Wang et al. (2016) obtained 86.9% (word-level) accuracy for pitch accent detection and 89.5% for phrase boundary detection on one speaker of BURNC.

The resulting models were therefore *speaker-dependent*. Hasegawa-Johnson et al. (2005) performed prosodic event detection as a word sequence labeling task using a multilayer perceptron and aggregated acoustic features as well as lexico-syntactic features. They trained a speaker-independent classifier and obtained around 84% accuracy for pitch accents and around 93% accuracy for phrase boundaries.

There is far less work on **prosodic event classification**. Sun (2002) and Levow (2005) both performed syllable-level classification of four pitch accent types in BURNC. The first paper reports 74.3% accuracy using acoustic features and 79.5% using lexical features, and the second paper obtained 81.3% using acoustic features. Rosenberg (2009) obtained 60.9% and 79.0% accuracy for the classification of 5 ToBI types of both pitch accents and phrase boundaries, respectively, using acoustic features and evaluated with 10-fold cross-validation on BURNC. In another publication, Rosenberg (2010b) used quantized contour modeling to classify ToBI types from given locations and obtained almost 67% accuracy for pitch accent classification and 72.9% for phrase boundary classification.

In order to obtain robust models and to assess their **generalization** capabilities, the evaluation can be done across corpora, or even across languages with related prosody. A specific challenge for speech processing models is the difference between read speech and spontaneous speech. Read speech is typically considered to be "clean" data, consisting of complete sentences and fewer speech errors. Spontaneous speech often does not consist of complete sentences, and can contain hesitations, disfluencies, and speech repairs. Dialogue speech can also contain backchannels or laughter. Yuan et al. (2005) investigated the effects of speech genre and speaker differences on pitch accents in four different corpora including BURNC. They observed that read speech contains more accented content words, while spontaneous speech contains more accented function words. Margolis et al. (2010) performed cross-genre experiments for automatic detection of pitch accents and phrase boundaries on BURNC and the Switchboard corpus (Godfrey et al., 1992). They found that the use of textual features increased the amount of cross-genre error, while the acoustic features showed a better overlap across genres. Levow (2009) tested how the proficiency levels of non-native English speech affected the results of automatic pitch accent detection, and concluded that it does not affect the performance significantly.

Rosenberg et al. (2012) and Soto et al. (2013), reported results for **cross-language** detection of pitch accents and phrase boundaries, respectively. They applied the AuToBI toolkit (Rosenberg, 2010a) to different corpora including the Boston Directions speech corpus (BDC) (Hirschberg and Nakatani, 1996) and the DIRNDL German radio

news corpus. The cross-corpus pitch accent detection accuracy for these two datasets was comparably low: around 76%. For phrase boundary detection, Soto et al. (2013) observed a drop in F1-score from around 91% when training and testing on DIRNDL to 89% when training on BDC. In the opposite direction, it appeared that training on DIRNDL and testing on BDC (79%) was no worse than the within-corpus case. Rosenberg et al. (2012) also investigated language-specific differences in acoustic features describing pitch, intensity, duration and spectral balance, but did not find patterns that predicted cross-language performance.

The performance of prosodic event recognition (specified as the full classification of event types) suffers from an unequal distribution of instances for each class, as pointed out in the literature (Rosenberg, 2010b). The H* pitch accent is by far the most frequent type in most corpora, and therefore models tend to "overpredict" this class. This issue of **unbalanced class sizes** also affects phrase boundary detection, since phrase boundaries tend to comprise less than 20% of the words in a corpus (see Section 3.1). While this does necessarily pose a problem for all learning methods, it should be kept in mind when choosing the performance measure. For example, Rosenberg's AuToBI system (Rosenberg, 2010a) achieves a phrase boundary detection accuracy of 90.8%, but an F1-score of 81.2% (in cross-corpus evaluation). For phrase boundary detection on one speaker, Wightman and Ostendorf (1994) obtained around 94% accuracy and an F1-score of 76.2%, using a Hidden-Markov-Model and acoustic information. Therefore, if *accuracy* is to be used to evaluate these tasks, the majority class baseline should also be taken into account.

Another issue that is frequently mentioned and should be kept in mind when evaluating automatic prosodic labeling techniques is the **inter-annotator** agreement on the gold-standard labels the model is trained on. Despite their expertise in the field, human annotators often disagree on how prosodic events should be labeled, especially when it comes to the distinction of different types. This is because prosodic patterns are often subjective and human listeners must rely on many acoustic cues to clearly identify fine prosodic detail. Compared to other linguistic annotation tasks, the inter-annotator agreement for prosodic annotation is relatively low. Generally speaking, listeners agree more on the presence or absence of a prosodic event than on specific tonal shapes. Pitrelli et al. (1994) reported an inter-annotator agreement of around 81% for pitch accent placement and 64% for tonal categories by testing on several corpora. Although they did not consider the agreement on the labeling of intonational phrase boundaries, they reported higher levels of near-agreement for ToBI break indices, namely 92% (the exact agree-

ment was 66%). The BURNC corpus also comes with information on inter-annotator agreement, namely 91% for pitch accent presence. While there is no information on phrase boundary agreement in BURNC, Hasegawa-Johnson et al. (2005) suggest that it should be higher than for pitch accents, similar to the results reported by Pitrelli et al. (1994).

The relatively low inter-annotator agreement has several implications for prosodic event recognition. First, the performance of models trained on human-annotated labels can also be expected also reflect human error. Second, that prosodic event detection is an easier task than prosodic event classification, both for humans and automatic methods. On the one hand, the low agreement on prosodic event types is an additional disadvantage for training models, since the quality of the data may not be sufficiently high, and may contain inconsistencies in labeling. On the other hand, it can also be considered as motivation for the use of automatic methods. Since the quality of both manually annotated and automatically predicted labels can not be expected to be "perfect", automatic methods provide the advantage of requiring less effort. A statistical learning method may also be able to model more complex cues than a human would. This is one of the main reasons why, given a sufficient amount of data, neural networks and deep learning methods are employed. As reviewed in Section 2.3.4, they have shown to be effective for prosodic event detection.

2.4 Technical background: Neural Networks

This section covers the basic technical background of the neural network architectures and training methods used in the experiments described in this thesis. Most of the formal descriptions and notations are taken from Goodfellow et al. (2016) and Goldberg (2017).

2.4.1 Basic feed-forward neural network

The idea behind the term *neural* is that it is loosely based on the processing of information in the human brain: it is an architecture of many connected individual neurons with activation functions that control the information flow through the network. The neurons receive input from other units, and if the resulting output of the activation function exceeds a certain threshold, then the information is passed on to the next neurons.

A neural network consists of an input layer, an output layer, and one or more hidden

layers. The term *deep learning* stems from the convention of stacking several hidden layers to create deeper networks with a larger learning capacity. The neurons are weighted by parameters that the model tunes via supervised learning: given enough training instances x_i and their respective labels y_i, the parameters $\Theta = (W, b)$ are tuned so that the output of a cost function, expressed as the overall difference between the reference labels and the predicted labels ($\hat{y}$), is sufficiently small.

The basic architecture of a neural network is a **feed-forward neural network**. Mathematically, the input is a vector x_i, and W is a weight matrix that is multiplied at each layer (b is the bias, as used in typical linear models).

One of the properties that makes neural networks powerful models is that they are nonlinear; the **activation functions** in the units of the hidden layer perform nonlinear transformations Φ of the input, such that $y = f(x; \Theta, w) = \Phi(x; \Theta)^{\mathsf{T}} w$. An activation function is the function governing the behavior of each individual neuron in the network. Nonlinear transformations are useful to project the data such that it becomes separable (as illustrated by the famous XOR problem). The hidden layers are denoted by h, and the activation function by g. The activation function takes the input x from one or several neurons from the previous layer and outputs a value $g(x)$ that is passed on to the next layer.

A two-layer feed-forward network is thus formally expressed as in Goldberg (2017):

$$NN(x) = y \tag{2.5}$$

$$h_1 = g_1(W_1 x + b_1) \tag{2.6}$$

$$h_2 = g_2(W_2 h_1 + b_2) \tag{2.7}$$

$$y = W_3 h_2 \tag{2.8}$$

All neurons of a single layer have the same activation function; often this is kept consistent across all hidden layers in a deep network. However, different activation functions can be chosen for the hidden layer and the output layer, respectively. A *rectified linear unit (ReLU)* is the default activation function of hidden neurons in many standard applications. It is a piecewise linear activation function that returns 0 if $x < 0$, and x otherwise. Another frequently used function is the hyperbolic tangent ($\tanh$), which squashes the input into the range [1,-1].

The activation functions in the output layer are responsible for making the actual class predictions. Therefore, the first part of the network performs nonlinear feature

transformations made by the neurons, and the last part of the network can be regarded as a basic logistic regression classifier, as used in standard statistical learning algorithms. The output layer of a neural network is usually controlled by a *sigmoid* or *softmax* activation: A sigmoid function is a logistic function that “squashes" the input to a value between 0 and 1:

$$\sigma(x_i) = \frac{1}{1 + e^{-x}} \tag{2.9}$$

While the sigmoid function is only used for binary classification, the softmax function is a generalization of the sigmoid function: The softmax function yields a probability distribution over several classes.

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_j e^{y_j}} \tag{2.10}$$

The output for each category of labels is a value between 0 and 1, and the probabilities for all output classes must sum to 1.

2.4.2 Optimization and training

The training of a neural network is an iterative process, consisting of a forward pass and a backward pass. The term *feed-forward* refers to the forward propagation of information through the network by computing the output from the input using the aforementioned functions. In order to tune the parameters of the model, the predictions must be compared to the reference labels y (this is what makes neural networks *supervised* learning methods). This is done using an objective function, and the information is transported backwards through the entire network to update the neurons accordingly. **Backpropagation** is the stepwise computation of the gradient of the **objective function** (or *cost*, *loss*) with respect to the parameters of the network.

The objective function of a neural network is commonly the maximization of the likelihood of the training data. This is equal to minimizing the negative log-likelihood, referred to as the *categorical cross-entropy*. It is suitable for multi-label classification and is used with a softmax activation function in the last layer. Other commonly used functions are binary cross-entropy, which is used for binary classification (and combined with a sigmoid activation function) or ranking loss.

Categorical cross-entropy assumes that the output is a true multinomial distribution over all labels $y_1 ... y_n$. The predicted output $\hat{y}$ is the conditional probability that the

reference label belongs to class i: $y = P(y = i|x)$. The objective function of categorical cross-entropy is defined as the sum of the log probability over all outputs in the set of possible labels:

$$L(\hat{y}, y) = \sum y_i * log(\hat{y}_i) \tag{2.11}$$

Since the objective function of multilayer models is nonconvex, the training process is not guaranteed to converge. Thus, the aim of the training algorithm is to minimize the error as much as possible. There exist several strategies of minimizing the error and optimizing the network; the most common are listed in the following.

The actual task of finding the optimal parameters is referred to as *optimization*, which is solved using **gradient-based optimization** algorithms. The loss over the training set defines the overall training error, and the gradient provides information on the direction in which the parameters should be updated in order to minimize the error. The gradient indicates the steepness of the objective function towards the extrema points of the function. Since the objective is to minimize the output of the loss function, the parameters are moved in the direction opposite of the gradient. Gradient-based methods for training neural networks exploit the chain rule of derivation, that is each layer is a function of the next: $f(x) = f_3(f_2(f_1(x)))$ (Goodfellow et al., 2016). The most common optimization algorithm is *Stochastic Gradient Descent* (SGD), which computes a stochastic approximation of the optimal parameters Θ. SGD takes the input x, the output y and the function $f(x; \Theta, W)$ and sets Θ to minimize the estimated loss L over the entire training set.

Optimization is usually done in small sets of training samples called *minibatches*. This makes the computation more efficient compared to the option of computing the objective function for all data simultaneously, and makes the estimation more robust compared to computing each sample individually. The average gradient is computed over each minibatch. At each iteration, the parameters are updated according to the direction of the gradient. The update to the parameters at each is weighted by the *learning rate* n.

Other optimization algorithms based on adaptive learning rates often yield good performance, for example the *Adam* algorithm (Kingma and Ba, 2015). In this case, the learning rate is updated for each minibatch in a sophisticated manner: In areas where the gradient is large, the learning rate is decreased to a larger degree to enable faster learning in more shallow regions. The Adam algorithm also adopts the idea of "forgetting" past gradients in the case of nonconvexity by using an exponential moving average.

One caveat of the strong modeling capacities of large deep learning models is that the

large number of parameters also makes them prone to *overfitting*. Overfitting occurs when the training data is modeled so strongly that the model performs poorly on the test data (and later, on unseen data). For this reason, it is necessary to **regularize** the models, that is, to control the parameter update to slightly increase the training error and, at the same time, reduce the test error. This is achieved by adding a regularization function R to the objective function.

Thus, the optimal parameters are determined by minimizing the loss function and taking the regularization function into account (Goldberg, 2017):

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}}\, L(\Theta) + \lambda R(\Theta) \tag{2.12}$$

The parameter λ controls the amount of regularization and is tuned during training on the validation set. A common regularizer is the L_2 norm or *ridge regression*, which uses the squared vector norm to regularize the parameters of W towards zero.

$$R_{L_2} = \|W\|_2^2 = \sum (W_{i,j})^2 \tag{2.13}$$

A popular technique of limiting the amount of overfitting is *dropout* (Srivastava et al., 2014). This method makes the model robust to missing information or noise by randomly setting a certain amount of the neurons in a specific layer to zero (during optimization). This way, the model learns to distribute its weights and to not rely too strongly on certain neurons. The dropout rate p determines the probability with which a neuron is "corrupted", and is typically set to 0.2, 0.5 or 0.8. For example, a dropout rate of $p = 0.5$ means that half of the neurons will be corrupted.

While the parameters Θ of a neural network are tuned automatically during training, the **hyperparameters** must be specified beforehand. The standard hyperparameters of a feed-forward neural network are the number of layers and the number of neurons per layer, as well as the choice of activation, optimization and regularization functions. These need to be set manually by the developer, since different applications on different datasets require a custom network design to achieve the best possible performance.

Due to the nonlinearity and thus nonconvexity of the learning problem, the optimization algorithm can get "stuck" in local minima. Neural networks are therefore sensitive to how the parameters are **initialized**. This needs to be kept in mind during implementation; and methods such as random sampling aim to overcome this. The training procedure is then typically repeated several times with different random seeds. Afterwards, the results are averaged.

A neural network is trained for several epochs and the optimization is tracked by computing the accuracy (the percentage of correct predictions over all samples) of the predictions on a held-out validation set. Often a strategy called *early stopping* is employed to avoid overfitting by terminating the training procedure when the validation accuracy stops decreasing, or instead starts to increase as the training error keeps decreasing.

Modern deep learning methods are typically implemented using standard deep learning libraries such as Theano (Theano Development Team, 2016) and Tensorflow[6]. These libraries enable the code to run on CPUs (central processing units) or more powerful GPUs (graphical processing units). At a more higher level, libraries such as Keras (Chollet et al., 2015) enable faster implementation using their built-in layer functions. These include several default and specialized options for optimization and architecture design, and therefore it is possible to readily implement successful models using these tools.

Apart from FFNNs, two major types network architectures have proven very successful in the field of speech and language processing: These are convolutional neural networks (CNN) and recurrent neural networks (RNN).

2.4.3 Convolutional neural network

Convolutional neural networks (CNNs) were originally inspired by the primary visual cortex of the brain, and have proven very effective at solving image processing tasks in the field of computer vision (Goodfellow et al., 2016). The input to a CNN is arranged in a grid-like fashion, or matrix. CNNs can operate on *tensors*, or multidimensional arrays, of up to 4 dimensions. This section describes the standard 2-dimensional case, as illustrated in Figure 3.2.

The idea of a CNN is to capture local patterns in the input, independent of their actual position in the matrix. This is accomplished by shifting smaller matrices, referred to as *kernels* or *filters*, across the input matrix and repeatedly performing matrix multiplications (that is, linear transformations) on the input. Each position in the kernel is assigned a weight, and local patterns are thus learned by applying the same kernel to different areas of the input matrix. Formally, the 2-dimensional input consists of values x_i, x_j and the much smaller 2-D kernels K are weighted by parameters W. A convolution operation involves taking the dot product between the kernels and the weight matrices

[6]https://www.tensorflow.org/

and transforming the input as follows:

$$(W * K)(x, y) = \sum_{i=1}^{d} \sum_{j=1}^{|K|} W(i, j) \cdot K(x - i, y - j) \tag{2.14}$$

The distance with which the kernels are moved after each operation is called *stride*. The stride is usually selected to be smaller than the width of the kernel, and the resulting overlap ensures that the kernel takes more of the input areas into account. The input to a CNN is usually padded with zeros so that the kernels have more "space" to capture the edges of the input matrix. The output of the convolution after a single kernel has been shifted across the entire input is referred to as a *feature map*. CNNs typically involve the use of several kernels, resulting in a 2-D output layer consisting of several feature maps. The idea of using several kernels is to let them each specialize on different patterns.

After a convolution layer, additional convolution layers may be stacked or a summarizing layer can be applied to perform *pooling*. Pooling combines the most salient neurons into one output vector. This is usually done by selecting the neuron with the highest activation within a shifted window. This procedure is referred to as *max pooling* Average pooling, that is taking the average within the sliding window, is a frequently-used alternative. The aim of pooling is to create a more general representation by making the model less sensitive to small changes in the input. It also reduces computational cost by summarizing the feature maps into much fewer pooling units. This effect can be increased by selecting a larger stride. The hyperparameters to be chosen when implementing a CNN are the number, size and stride of the kernels as well as the size and stride of the pooling windows.

Despite the many matrix operations that are involved in training a CNN, they are, computationally, relatively efficient. This is enabled by employing three main ideas, as listed in Goodfellow et al. (2016): The first is *sparse connectivity*. Since the kernels are much smaller than the input, the number of parameters involved is reduced, which in turn means that fewer operations and less memory are required. The fact that the convolution operation applies the same kernel to many different areas of the input is in line with the idea of *parameter sharing*. Finally, the concept of learning local, position-independent patterns pertains to the idea of *equivariance*: if the same pattern occurs in different areas of the input matrix, the CNN will create the same representation in different areas of the output matrix. These properties make a CNN a good intuitive choice for image processing tasks in the field of computer vision, but they have also been successfully employed in the field of NLP (Collobert et al., 2011; Goldberg, 2017).

2.4.4 Recurrent neural network

Many tasks in NLP and speech processing exploit the sequential nature of language, for example language models or automatic speech recognition. For this reason, network architectures referred to as recurrent neural networks (RNN) that model sequential data have become very popular in recent years. This network type processes input sequences $x_1, ..., x_n$ and models the input at each time step as a function of the previous inputs. RNNs can be used for sequence labeling tasks such as part-of-speech-tagging (Goldberg, 2017), in which each input x_i is associated with a label y_i. Alternatively, the whole sequence $x_1, ..., x_n$ can be mapped to one single label y, such as in sentence or dialogue act classification. Separate RNNs that process the input sequence in forward and backward directions, respectively, are often combined to a single bidirectional architecture. This often yields better performance, especially in cases where the future context is useful for making good predictions, as in the natural language setting. An illustration of a bidirectional RNN is given in Figure 4.3.

A **simple recurrent network**, or *Elman*-type (Elman, 1990), has recurrent connections between hidden units. A formal description of the forward pass through a simple RNN is provided by Goodfellow et al. (2016) (2.15 through 2.18). The weight matrices are denoted, respectively, by W for the connections between the input and the hidden units, U for the connections between hidden units, and V between the last hidden layer and the output units. The linear transformations at the hidden layers take the previous hidden layer into account h_{t-1}. The activation function at the output layer in this case is a softmax activation function.

$$a_t = b + Wh_{t-1} + Ux_t \tag{2.15}$$

$$h_t = g(a_t) \tag{2.16}$$

$$o_t = c + Vh_t \tag{2.17}$$

$$y_t = softmax(o_t) \tag{2.18}$$

Another widely-used RNN architecture, referred to as a *Jordan*-type RNN, has recurrent connections between the output at the previous time step and the hidden units at the next time step. In this case, the equivalent equation to Equation 2.15 is:

$$a_t = b + Wo_{t-1} + Ux_t \tag{2.19}$$

As in other neural network architectures, several hidden layers can be stacked to create more complex interactions between the neurons of the network. The backward pass through an RNN is called *backpropagation through time*.

Since the entire sequence history is taken into account, RNNs allow models to abandon the Markov assumption that the conditional probability of a future element only depends on a limited amount of previous elements. For this reason, RNNs are strong learning methods for various applications in speech and language processing, since they are able to model long stretches of speech and text. However, RNNs face the difficulty of effectively modeling long-term dependencies due to the frequently observed *vanishing gradient* problem, which occurs when many derivatives are multiplied. The computed weights become exponentially smaller during backpropagation, with the result that the updates no longer reach the earlier input nodes in the network. This problem can be overcome by the use of *gated* recurrent architectures, such as the long-short term memory network (LSTM), described below.

A **long short-term memory network** (Hochreiter and Schmidhuber, 1997) is a variant of an RNN in which the hidden units consist of *cells* that introduce a number of additional parameters. The flow of information through the cell is controlled by other hidden units, called *gates*. A gate is a vector that is multiplied element-wise with another vector. It "controls" the computation of updates in the cell by functioning as an array of "on/off switches" that are also tuned during training. In an LSTM, the cells are split into two parts: a memory component c_j and a hidden state component h_j. The updates are controlled by an input gate i, an output gate o and a forget gate f. The three gates as well as the update candidate z (representing the new input information) each take the current input x_j and the previous hidden state $h_{(j-1)}$ as input, and the output is transformed by an activation function (σ or tan$h$). A value from the update candidate may be passed into the cell according to the activation function of the input gate. Inside the cell, the memory is controlled by a forget gate, which determines how much information from the previous memory to keep and how much is forgotten, that is, overwritten by the update candidate. Finally, the updated memory is used to compute y_j if permitted by the output gate.

The components of an LSTM are formally expressed as follows (Goldberg, 2017):

$$s_j = [c_j, h_j] \tag{2.20}$$
$$c_j = f \circ c_{j-1} + i \circ z \tag{2.21}$$
$$h_j = o \circ tanh(c_j) \tag{2.22}$$
$$i = \sigma(x_j W^{xi} + h_{j-1} W^{hi}) \tag{2.23}$$
$$f = \sigma(x_j W^{xf} + h_{j-1} W^{hf}) \tag{2.24}$$
$$o = \sigma(x_j W^{xo} + h_{j-1} W^{ho}) \tag{2.25}$$
$$z = tanh(x_j W^{xz} + h_{j-1} W^{hz}) \tag{2.26}$$
$$y_j = h_j \tag{2.27}$$

This architecture solves the vanishing gradient problem by allowing the gradient to "flow" through long stretches of the network; in other words, derivatives are multiplied only when determined by the gating mechanism.

2.4.5 Neural network analysis

The fact that neural networks consist of (deep) hidden architectures, which perform non-linear transformations resulting in large parameter spaces, make them powerful learning methods. However, this also makes it difficult to interpret how and what a network has learned from the training data. Recently, the machine learning community has therefore turned more attention towards research that seeks to gain insight into this question. The notion of analyzing what and how neural networks have learned is described by the terms *interpretability* (Lipton, 2016), *explainability* (Montavon et al., 2018), or *introspection* (Krug and Stober, 2018).

There exist several approaches to interpreting neural networks, and many analyze the "inner workings" of the network. That is, the aim is to either understand the role of hidden units, activations and entire layers, or to explain the decisions that the model makes at test time. An overview of methods is found in, for example, the work by Montavon et al. (2018).

Most work on neural network analysis so far has been in the field of computer vision. These analyses often involve the visualization of neural networks, for example CNNs for image classification (Zhang and Zhu, 2018). Bach et al. (2015) proposed a method called layer-wise relevance propagation. This method decomposes the nonlinear interactions backwards through the network, computing the relevance of each input pixel, and

visualizes the overall contributions in the form of a heat map.

The field of speech and language processing has also started to direct more attention to this topic, resulting in the emergence of venues catering specifically to this field (BlackboxNLP workshop[7], Interspeech 2018 special session[8]). For NLP tasks, the focus is often placed on the analysis of RNN models (Adi et al., 2017). Some examples of work on models of speech processing are given in the following: Nagamine et al. (2016) investigated how the nonlinear transformations in each layer of a neural network acoustic model learn to separate phoneme categories. Becker et al. (2018) used layer-wise relevance propagation to visualize models of audio classification on spectrogram data. Methods initially developed for analyzing image classification models are not always suitable for analyzing models of speech, since a visual interpretation is not always useful for interpreting relevant parts of speech input. Krug and Stober (2018) also pointed out this issue and adapted the above technique to analyze CNN models for automatic speech recognition.

In this thesis, we take a different approach: Instead of analyzing the neural network itself, we focus on analyzing the *output feature representation* that is learned by the neural network. We view the neural network from the perspective of representation learning (Bengio et al., 2013), since the use of such methods is motivated by the notion of letting the model learn the best feature representation on its own. This sets neural networks apart from other machine learning tasks, in which the features used for classification must be determined by the developer.

The analysis in this thesis is similar in idea to the method used by Adi et al. (2017) for analyzing sentence embeddings in NLP. Embeddings are vector representations that are been extracted from an neural network architecture that has been trained to create fixed-length representations of words or sentences. Thus, the "black box" that is analyzed is not the neural network itself, but the feature representation that it creates. The idea of this method is to train separate classifiers on the neural network's output representation to predict other properties of the data. If the feature representation can be used to predict a specific property of the data, then it can be concluded that is encoded in the representation and therefore, that the neural network has learned these properties. Similar methods were also published during the same time by Ettinger et al. (2016) and Shi et al. (2016) which analyzed semantic and syntactic representations, respectively. Since then, this method has also been adopted for analyzing speaker embeddings (Wang

[7] https://blackboxnlp.github.io/

[8] Deep neural networks: how can we interpret what they have learned? https://www.isca-speech.org/archive/Interspeech_2018/

et al., 2017). To the best of our knowledge, our analysis, which we present in Section 5, was the first of this type to be applied to prosodic models.

Recently, Kakouros et al. (2019) have also proposed a method of analyzing neural network models for prominence classification. They extracted hidden layer representations to determine how well each layer of a feed-forward network discriminates between speaker, gender and prominence class label, and analyzed autoencoder bottleneck representations to determine how well F0, energy and spectral tilt contribute to the classification task on a Dutch speech corpus. In our work, we analyze the final output representation instead of the hidden layers of the network and investigate a wider range of input information, including temporal and context information. We also focus on more specialized architectures: Our main method of prosodic event detection uses a convolutional neural network, but we also compare it to a recurrent neural network.

3 Prosodic Event Detection using Convolutional Neural Networks

This section first introduces the datasets used in this thesis (Section 3.1) describes the method of prosodic event detection using convolutional neural networks developed for this thesis in Section 3.2. The method is introduced as a supervised model for detecting pitch accents and phrase boundaries that requires very little preprocessing: The only input information is word-level aligned speech data and a set of frame-based acoustic descriptors extracted from the speech signal. Section 3.4 gives an overview of the model performance on the BURNC dataset, focusing mainly on optimization using slightly different experimental setups. Section 3.5 discusses the generalization ability of the model, and Section 3.6 takes a closer look at the performance of different acoustic features. The advantage of using only acoustic features is that the model is readily applied to new datasets and languages, but we also describe experiments on the effects of including lexical features in Section 3.7. The choice of using a convolutional neural network for prosodic event detection was made because it learns local patterns in the frame-based matrix representation of the speech signal. Intuitively, this means finding local peaks or valleys in F0 or energy that correspond to a prosodic event. Nevertheless, we also compare the CNN to two other network architectures: The first is a feed-forward neural network as a method that does not take frame-based features, but representations aggregated over words (Section 3.8). In this section, we also test the effect of including different forms of temporal information. The second architecture that we use for comparison is a recurrent neural network (LSTM) in Section 3.9 that takes the identical frame-based input information as the CNN.

3.1 Data

The datasets chosen for this work all have previously been used for prosodic event detection and have been manually annotated with ToBI (Silverman et al., 1992) or GToBI(S)

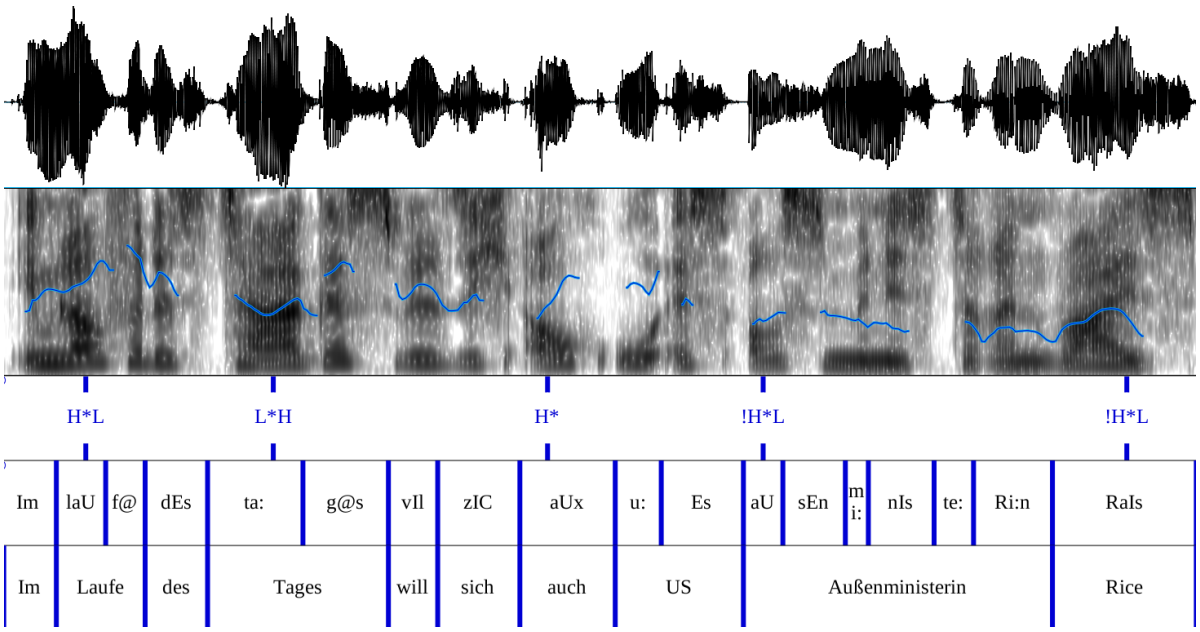


Figure 3.1: An excerpt of the DIRNDL speech corpus with word, syllable and pitch accent annotations, displayed using the Praat speech analyzer (Boersma, 2001).

labels (Mayer, 1995). Figure 3.1 shows an excerpt from one of the corpora introduced in this section. Most corpora contain word and syllable alignments. For most experiments, the word-level alignments as well as the binary absence or presence of prosodic events are the only information required for preprocessing. (The only experiments that use the syllable annotations are in Section 4.) Pitch accents and phrase boundaries were modeled in separate classification tasks. Since this work focuses on binary classification, we grouped all words with a pitch accent (*) into the positive *accented* class and all intonational phrase boundaries (%) into the positive *boundary* class. Unsure events, that is events where the annotator was not certain of the presence of a pitch accent, were grouped into the *none* class. A description of the ToBI types used in the few classification experiments in this work are given in the respective section, 3.4.

The Boston University Radio News Corpus (BURNC) The first English corpus is the Boston University Radio News Corpus (BURNC) (Ostendorf et al., 1995). It is one of the most widely used corpora for research on automatic prosodic labeling. This corpus consists of recordings of radio news broadcasts, read by professional speakers. The recordings were orthographically transcribed, automatically force-aligned and manually corrected. In our experiments, we used a subset with 3 female and 2 male speakers that

amounts to around 2.5 hours of speech.

Table 3.1 lists the number of words and the ratio of words labeled with events in BURNC. Around half the words in the dataset are pitch accented (51.5%) and 19.3% carry phrase boundaries. Due to some inconsistencies during preprocessing, we obtained slightly different total word counts for the pitch accent and phrase boundary datasets. Because these datasets were used separately (pitch accent and phrase boundary detection are two separate tasks), however, this was inconsequential for our experiments.

The Boston Directions Corpus (BDC) The second corpus is the Boston Directions Corpus (BDC) (Hirschberg and Nakatani, 1996), which was also used by Rosenberg (2009). This corpus is smaller than BURNC; it contains around 2 hours of recordings of one female and three male non-professional speakers giving directions. The monologues were first recorded as spontaneous speech and then the transcriptions were re-read by the same speakers. This corpus represents an entirely different speech genre, with the main difference being that the utterances are shorter and contain a comparably limited vocabulary compared to radio news. We used both the read and spontaneous subsets as one dataset in this work. As shown in Table 3.1, this dataset contains 55.0% pitch accents and 15.3% phrase boundaries.

LeaP corpus of non-native prosody The LeaP corpus (Milde and Gut, 2002) is a dataset that was collected for prosodic research on non-native speech. In our experiments we used part of the LeaP corpus that consists of read stories and story retellings by non-native speakers with different L1 backgrounds. The subset contains speech recorded from 35 different speakers (22 female, 8 male) across varying proficiency levels. In total the subset comprises 14 different L1 backgrounds. Previous work (Levow, 2009) has shown that despite the lower proficiency, the automatic pitch accent detection performance on LeaP is comparable to that on native speech. This corpus was only used for pitch accent detection experiments in Section 3.7.[1]. The LeaP corpus contains around 57% pitch accents (Table 3.1).

DIRNDL corpus of German radio news We used a subset of the DIRNDL German radio news corpus (Eckart et al., 2012) that is prosodically annotated with GToBI pitch

[1] In order to use LeaP in cross-corpus experiments, the speech signal sampling rate required downsampling to 16kHz to match that of the other two corpora. In preliminary experiments, downsampling did not affect the results, whereas the training and testing using features extracted using two different sampling rates respectively slightly harmed the performance.

accents and phrase boundaries. The radio news in this corpus was read by several female and male professional speakers. Thus, the genre is similar to that of BURNC and can be considered a cross-language equivalent. As in the above cases, the manual transcriptions were force-aligned to the audio files. The subset of DIRNDL that contains manual prosodic annotations is larger than BURNC and amounts to nearly 5 hours of speech. The distribution of prosodic events is similar to the previous corpora: 50.7% pitch accents and 12.3% phrase boundaries (see Table 3.1).

corpus	words	pitch accents (%)	phrase boundaries (%)
BURNC	26742	51.5	–
	28468	–	19.3
BDC	19225	55.0	15.3
DIRNDL	35358	50.7	12.3
LeaP	14651	43.3%	–

Table 3.1: The total word count and percentage of pitch accents and phrase boundaries in the datasets used in this work. The differences in word counts in the BURNC datasets were the result of missing annotations removed after cleaning.

3.2 Method

3.2.1 Convolutional neural network

The input to the CNN is a matrix $X \in R^{d \times s}$ containing frame-based acoustic features extracted from the audio signal. We use a d-dimensional feature vector for each frame. The total number of frames s in the input matrix is determined by the longest context window in the dataset. Zero padding is added to the trailing end of the matrix to ensure that all input matrices have the same size.

The convolution operation involves a set of 2-dimensional kernels K (with width $|K|$) that are shifted across the input matrix. The kernels consist of matrix weights W that are tuned during training. The output is a set of feature maps. The model contains two convolution layers, that is, an additional set of kernels is applied to the feature maps that are created in the first convolution layer. The first layer of the CNN consists of 100 2-dimensional kernels of the shape $6 \times d$ and a stride of 4×1, with d as the number of features. We use a feature set of size $d = 7$ in most experiments (that is, 6 acoustic descriptors and one position indicator, as explained in the following two sections). The kernels span the whole feature set to ensure that all features are learned

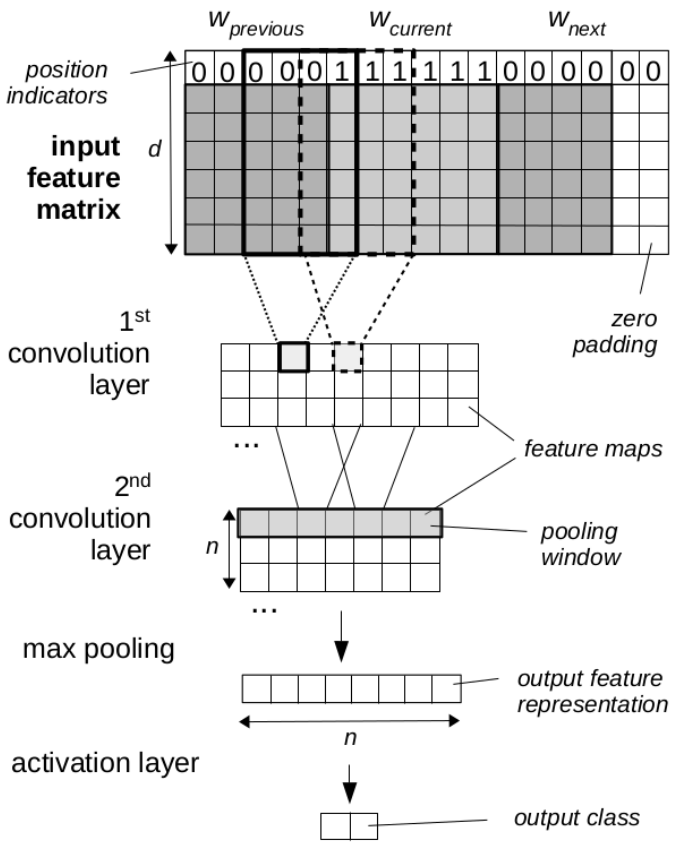


Figure 3.2: Convolutional neural network (CNN) for prosodic event detection with an input window of 3 words, the current word $w_{current}$ to be labeled and the two surrounding words $w_{previous}$ and w_{next}, and position indicating features. Using a set of kernels that span the entire input feature set d (the span in the other dimension is shown only schematically), the model learns an n-dimensional output feature representation which is used to classify $w_{current}$ as carrying a prosodic event or not.

simultaneously. The second layer consists of 100 kernels of the shape 4×1 and a stride of 2×1. (These hyperparameters had been optimized in preliminary experiments, that is, tuned by training on the BURNC dataset and using a held-out part for validation. For example, kernels with length 20 were tested for phrase boundary detection with the idea of capturing longer stretches of speech, however, this did not improve the performance.)

Afterwards, we apply max pooling, which is a simple selection of the maximum value in each feature map. The output values are concatenated to one final output representation. Thus, after max pooling, the final feature vector has the length $n = 100$. For regularization, we also apply dropout (Srivastava et al., 2014) to this last layer, which

corrupts a specified percentage of the neurons with the aim of making the model more robust to unseen data. We set the dropout rate to either $p = 0.2$ or $p = 0.8$ (e.g. after experimenting with both in Section 3.7, we found that a higher dropout rate lead to better generalization across datasets, see discussion in Section 3.3). The resulting feature vector is fed into the softmax activation layer which consists of 2 units for binary classification, and can be extended to consist of several units for multi-class classification.

The models were trained with an adaptive learning rate (Adam (Kingma and Ba, 2015)) and $l2$ regularization. We tracked the validation accuracy on the development set and after all training epochs are completed, we applied the overall best model to the test data. We initially set the model to train for 50 epochs (for example in Section 3.4), but later found that 20 epochs were sufficient since we did not observe a significant increase in validation accuracy past this point. We assume this to be an effect of the relatively small size of the dataset relative to the size of the model.

In order to take the slight range in performance due to random initialization of the parameters into account, all experiments were repeated 3 times and the results were averaged. The network was implemented using the Keras library (Chollet et al., 2015)[2] using the Theano (Theano Development Team, 2016) backend. Training and testing was performed on a single GPU.

3.2.2 Acoustic features

As shown in the top part of Figure 3.2, the input to the CNN is a set of frame-based features. The features are fairly "low-level" in the sense that they are descriptors at the frame level, that is, they are very local representations of only a few milliseconds of speech. The idea is for the CNN to "scan" the input matrix and find relevant patterns across the frames on its own, including interactions between the features. Our method uses a set of acoustic features that were chosen to be simple and fast to obtain. We used the openSMILE toolkit (Eyben et al., 2013), a command-line tool that efficiently extracts customizable acoustic descriptors from audio files. We chose this toolkit because it is efficient to use and provides a wide range of signal processing methods with which acoustic features can be extracted and customized. It also comes with various "recipes" for pre-implemented features, including a set of prosodic features that consists of descriptors that are known to correlate with prosody.

In our experiments we used the following acoustic features and frame sizes (the

[2]The experiments in Section 3.4 were implemented in Lasagne (Dieleman et al., 2015)

openSMILE feature names are given in brackets):

- root mean square signal frame **energy** (*pcm_RMSenergy*)
- **loudness**, approximated from auditory weighting of the Mel spectrum (*loudness_sma*)
- **F0**, computed and smoothed via subharmonic summation (*F0final_sma*)
- **voicing probability**, computed from pitch (*voicingFinalUnclipped_sma*)
- **harmonics-to-noise-ratio (HtNR)**, logarithmic in dB and computed from the cepstrum at F0 position vs. total energy (*HarmoncisToNoiseRatioACFLogdB_sma*)
- **zero-crossing-rate (ZCR)** of the time signal (*pcm_zcr*)

All features were extracted with 10ms shift size. Thus, the kernel width of 6 frames in the first convolution layer corresponds to a filter that spans 60ms of speech.

The frame sizes for each feature had been optimized in previous experiments on BURNC. A first version of this feature set was inspired by the pre-implemented prosody features provided by openSMILE and consisted of only the first 5 features, which we extracted for 20ms frames (based on the typical duration of phonetic segments). Since we obtained good performance results using these features (in Section 3.4), we decided to keep this feature set for the prosodic event detector and only made small updates to the feature set during experimentation. In all other experiments we therefore extracted energy, loudness and ZCR with a 20ms frame size, and F0, voicing probability and HtNR with a 50ms frame size.

Finding optimal input features is not the focus of this work, and we did not consider it feasible to test all possible acoustic descriptors that can be extracted using openSMILE and that correlate with prosody. We therefore limited our experiments to the investigation of the following input features: We tested an additional *Mel* feature set consisting of 27 features extracted from the Mel-frequency spectrum in Section 3.4 as an alternative feature set and in combination with the prosody feature set. We also compared different F0 features extracted using different methods implemented in OpenSMILE in Section 3.6. Since we did not find these alternative features or the combination of Mel and prosody features to show any improvement over our chosen feature set, we kept the prosody features as our standard feature set.

It is common to normalize acoustic features since they tend to vary considerable from speaker to speaker. Therefore, the descriptors were z-scored in most experiments. We

observed, however, that corpus-wide speaker-normalization lead to worse results than simply using raw features, and that normalizing across each individual utterance *file* lead to the best results, since the latter method provides not only a normalization per speaker but also per individual recording. We also refer to the results and discussion in Section 3.4.4.

3.2.3 Context information and position indicator feature

Most methods of prosodic event detection take a larger acoustic context into account (Levow, 2005; Rosenberg, 2009). This is necessary since prosodic patterns span longer stretches of speech, and because prosodic events are perceived relative to their acoustic context. Previous work has demonstrated the benefits of adding context information to PER (Rosenberg et al., 2015; Levow, 2005). The most straightforward approach is to add features that represent the right and left neighboring segments to form a type of acoustic context window (Rosenberg and Hirschberg, 2007; Schweitzer and Möbius, 2009; Wang et al., 2016). We take a similar approach, but since our input is frame-based, we extended the input features with a *position indicator feature*. This type of feature has been proposed for use in neural network models for relation classification (Vu et al., 2016a; Zhang and Wang, 2015). The method we use in this work and the reason why it is required explained as follows: The time intervals that indicate the word boundaries provided in the corpus are used to create the input feature matrices by grouping all frames for each word into one input matrix. We use a context window of three words ($w_{previous}$, $w_{current}$, w_{next}). Afterwards, we add zero padding to the end so that all inputs have the same size. Since the words have different lengths, there is no fixed position in the input matrix that pertains to each input word. Therefore, we append position indicators as an extra feature to the input matrices (see Figure 3.2). These consist of ones for the current word and zeros for the neighboring words. In the first convolution layer we set the kernel size to span the entire feature dimension, so that the model is constantly informed whether the $|K|$ current frames represent the current word or not.

In Section 3.4 we show experimental results comparing the performance of the network when using no context and when using context with and without the position indicator features, demonstrating that these are required to reach good performance levels. How this effect is connected to the max pooling method is analyzed in Section 4.

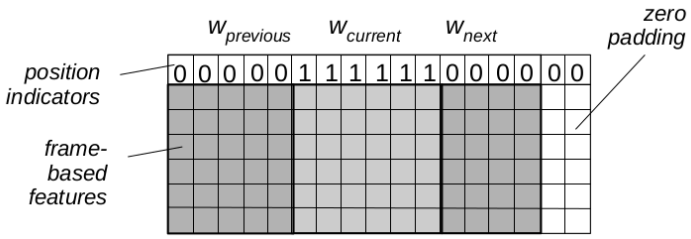


Figure 3.3: The input to the CNN is a frame-based representation of the speech signal and a 3-word context window with position indicator features

3.3 Preliminary remarks on comparability within this work

During experimentation, the model was continuously optimized, which is why certain model details (specifically the acoustic features, dropout rate, and training epochs) were updated during the course of this work. The model architecture and hyperparameters of the CNN described in the previous section remained the same. In early experiments (Section 3.4) we started with 5 acoustic features, each extracted with a 20ms frame size and without any normalization. In Section 3.7 we also used the early settings, but with 6 features. Later experiments used utterance normalization and 6 acoustic features with combination of 20 and 50ms frames (as described in Section 3.2.2). The dropout rate was increased from $p = 0.2$ in early experiments to $p = 0.8$ in later experiments that involved different datasets, since a higher dropout rate slightly increased the performance across corpora. Training was done for 50 epochs in the early experiments, and later for 20 since this was found to be sufficient: more epochs did not significantly increase the validation accuracy. We also mention which settings were used in the respective sections.

Most experiments were evaluated using cross-validation. For experiments that merely tested within-corpus performance and that did not take speaker-independence into account, the models were trained using either 10-fold (Section 3.7) or 5-fold cross-validation (all other sections). Since the splitting of the corpora into k folds for cross-validation was done randomly (except in the speaker-independent BURNC experiments), it could not be guaranteed that the splits were always the same across experiments throughout this work. We ensured, however, that directly comparable results, that is results occurring in one table or from a single experiment, were obtained from the same splits. The datasets for pitch accents and phrase boundaries were always used separately, and thus did not have the same splits.

Although the above changes lead to many of the results being not strictly comparable, we consider this effect to be minimal. We observed that end results can be increased slightly by manually optimizing individual parameters and settings, but also that the range of performance across experiments and general trends is more important for drawing meaningful conclusions.

It should also be noted that the question of statistical significance is generally not addressed in this work. This may seem unusual when dealing with statistical learning methods. We posit, however, that significance testing is not necessary in this work since we consider only sufficiently large performance changes of around 1 percentage point to be meaningful. The results of significance testing in Section 3.8.3 support this position. Furthermore, the random initialization differences stemming from the neural network algorithms inherently lead to a "natural" range in model performance, which we dealt with by running each experiment three times and averaging the results. For this reason, very small performance changes are not seen as significant. In contrast, performance increases as a result of different experimental setups are often quite noticeable and also tend to hold across different datasets.

3.4 Method performance on BURNC

This section describes the first set of experiments on the recognition of prosodic events on BURNC using convolutional neural networks. We describe different setups and experimental results. Most of this section (until Section 3.4.4) was published as a conference paper (Stehwien and Vu, 2017a).

In these experiments, the dropout rate was $p = 0.2$ and the models are trained for 50 epochs (these settings were updated in Section 3.4.4). The results are given in accuracy (the percentage of correctly labeled words over all words), which is why the majority class baseline is taken into account. This is especially important in cases where the classes are unbalanced, namely for phrase boundary detection.

We show results for each experiment with three different settings with respect to how context information is added. The first setting is without context, that is, only frames pertaining to the current word (*1 word*) are used as input. Next, we add frames for the right and left context words (*3 words*) without the position indicators introduced in Section 3.2.3. The final setting is with both right and left context words with position indicator features (*3 words + posind.*). These experiments show the large drop in performance in all cases, down to the majority class baseline level, when extending the input

to include the right and left context words. After adding the position indicating features, the accuracies of all tasks increased and exceeded those obtained from the single-word input in the speaker-independent case. This demonstrates that the position indicators are crucial for this method.

3.4.1 Speaker-dependent and speaker-independent evaluation

One important point to consider for any acoustic model is the fact that there are speaker-specific differences in speech, which means that a model trained on certain speakers will not necessarily generalize well to other speakers. In this step we therefore tested how the model performs when testing on a held-out speaker by performing the commonly used method of leave-one-speaker-out cross-validation. The BURNC dataset contains 5 speakers, and thus both speaker-dependent and speaker-independent evaluation is frequently done in related work. Most experiments in this work, however, use random k-fold cross-validation. We discuss the reasons for this decision in Section 3.5.2.

For our speaker-dependent experiments, we used the largest speaker subset f2b in line with previous methods (Sun, 2002; Wang et al., 2016). We tested the speaker-dependent models using 10-fold cross-validation and validated on 1000 held-out words from the respective training set. In the speaker-independent case, the models were trained and tested using leave-one-speaker-out cross-validation and validated on 500 words from a speaker of the same gender for early stopping (this way we avoid a too large mismatch between the validation and test data). Table 3.2 shows the number of words for each speaker in the datasets used for pitch accent and phrase boundary recognition.

Speakers	f1a	f2b	f3a	m1a	m2b
pitch accents	4375	12357	2736	3584	3607
phrase boundaries	4362	12606	2736	5055	3607

Table 3.2: Number of words in each subset of BURNC used in this work for pitch accent recognition and phrase boundary recognition.

Table 3.4 shows the results for pitch accent recognition on the single-speaker dataset and Table 3.5 shows the results obtained in speaker-independent experiments. We obtained up to 86.3% accuracy in pitch accent detection on f2b, which is comparable to the best previously reported results on purely acoustic input. The phrase boundary detection model for speaker f2b also yields a comparably high accuracy of 90.5%. Although leave-one-speaker-out cross-validation is a more challenging task, the performance that

we obtained is still quite high, namely 83.6% accuracy for pitch accent detection (see Table 3.5) and 89.8% for phrase boundary detection (see Table 3.8). Table 3.6 shows that even though the speaker f2b constitutes the largest speaker subset, leaving the least amount of data for training, the model did not perform much worse than on data from other speakers. Overall, there does not appear to be a distinctively "easy" or "difficult" speaker. In contrast to the pitch accent recognition results, we observe that the accuracies are lowest on speaker f1a, and highest on speaker m1b in both tasks (see Table 3.9).

Next, we compared how speaker-independent cross-validation compares to random splits. For this, we ran the baseline experiments (3 words and position indicators, prosody features sets) using random 10-fold cross-validation. In this setup, the training and test folds can overlap not only in terms of lexical content but also in terms of speakers. Table 3.10 shows the results for pitch accent and phrase boundary recognition. The results clearly show better numbers than obtained in the speaker-independent experiments, however the differences are limited to 1-3 percentage points, which means that the performance is still within similar range.

3.4.2 Detection of events vs. classification of event types

Although the classification of different event types is not the focus of this work, we included an experiment to test how this model would perform out-of-the box on prosodic event type labeling for comparability. For the detection task (binary classification of either pitch accent or phrase boundary presence) all labels were grouped together as one class. For the classification task, we distinguished 5 different ToBI types of pitch accents and phrase boundaries, as e.g. in (Rosenberg, 2010b), where the downstepped accents are collapsed into the non-downstepped ones: The pitch accent classes are (1) H* and !H*, (2) L*, (3) L+H* and L+!H*, (4) L*+H and L*+!H and (5) H+!H*. The boundary tones considered in this work mark the boundaries of intonational phrases: L-L%, L-H%, H-L%, !H-L% and H-H% (with %H). Uncertain events, where the annotator was unsure if there is an accent or boundary tone, are ignored for both detection and classification. Uncertain types, where the annotator was unsure of the event type, were ignored for classification.

The results for these experiments are shown in the Tables 3.4 to 3.10. Classification is always the more difficult task, for both pitch accent and phrase boundary detection. Following a similar pattern than the detection task, the best results were obtained for pitch accent classification on speaker f2b, namely 76.3% accuracy with a majority class

pitch accents	H*	L*	L+H*	L*H	H+!H*
% of events	70.8	4.4	20.6	0.4	3.8
phrase boundaries	L-L%	L-H%	H-L%	!H-L%	H-H%
% of events	56.6	38.5	3.3	0.4	1.3

Table 3.3: Percentage of prosodic event types in BURNC used for classification.

Feature set	prosody	Mel	prosody + Mel
Detection			
1 word	84.2	84.2	84.0
3 words	58.3	53.1	53.6
3 words + posind.	**86.3**	83.3	83.9
Classification			
1 word	74.4	72.7	73.5
3 words	52.4	47.8	47.8
3 words + posind.	**76.3**	72.3	72.9

Table 3.4: Results (accuracy) for pitch accent recognition on speaker f2b using 10-fold cross-validation. The majority class baseline for detection is 52.1%, for classification 48.2%.

baseline of 48.2%. The most difficult task is speaker-independent phrase boundary detection, where we only reached 87.3% accuracy with a majority class baseline of 80.7%. This is most likely an effect of the imbalanced classes and the fact that more fine-grained classification reduces the number of training instances which, in the case of phrase boundaries, is already quite low.

While we do not intend to go further into the analysis of individual event type classification, some remarks on the class proportions should be added: One of the reasons why the performance of this task is quite low is the fact that the classes are severely unbalanced (see Table 3.3). For pitch accents, the H* class is by far the most frequent, making up around % of the pitch accents. Similar to the observations made and discussed in (Rosenberg, 2009), we found that the model overpredicts this class.

As this method was not optimized for the classification task, this experiment is only for reference and demonstration purposes. Further work is necessary to adapt this model to classify prosodic event types with acceptable accuracy.

Feature set	prosody	Mel	prosody + Mel
Detection			
1 word	81.9	78.3	79.3
3 words	58.2	54.3	55.3
3 words + posind.	**83.6**	80.3	81.1
Classification			
1 word	68.0	64.7	64.5
3 words	50.5	48.4	48.4
3 words + posind.	**69.0**	65.9	65.3

Table 3.5: Results (accuracy) for pitch accent recognition with leave-one-speaker-out cross-validation. The majority class baseline for detection is 51.5% accuracy, for classification 48.8%.

Speaker	f1a	f2b	f3a	m1b	m2b
detection	85.6	82.9	83.5	81.4	84.8
classification	70.6	71.8	67.7	68.4	66.6

Table 3.6: Pitch accent recognition accuracies for each speaker using prosody and position features.

3.4.3 Comparison of acoustic feature sets

Next, we tested and compared two different acoustic feature sets. The *prosody* feature set consisted of 5 features extracted using openSMILE (smoothed F0, energy, loudness, voicing probability and HtNR), each extracted for 20ms frames with a 10ms shift. We compared this to a *Mel* feature set consisting of 27 features extracted from the Mel-frequency spectrum. These features are similar to those used for lexical stress detection by Shahin et al. (2016). All features were extracted for 20ms frames with a 10ms shift. In these first experiments, the features were not normalized. We tested the feature sets separately and combined to one feature set.

The results in Tables 3.4, 3.5, 3.7 and 3.8 show that in both the speaker-dependent and speaker-independent settings, the prosody feature set performs best, while the Mel and combined feature sets both yield similar results. In general, the prosody feature set appears to work best. The reason for this may be the fact that these features are a bit more "higher-level" than the Mel features, and because they represent more than just what is extracted from the Mel spectrum: they are features known to correlate with prosodic cues. Furthermore, the amount of data used in these experiments may not be enough to train robust models using Mel-spectral features. The only case where the

Feature set	prosody	Mel	prosody + Mel
Detection			
1 word	87.6	89.2	89.8
3 words	80.3	75.4	75.4
3 words + posind.	90.2	90.4	**90.5**
Classification			
1 word	85.6	87.6	88.0
3 words	79.7	74.5	74.6
3 words + posind.	87.8	88.7	**88.8**

Table 3.7: Results (accuracy) for phrase boundary recognition on speaker f2b with 10-fold cross-validation. The majority class baseline for both tasks is 77.9% accuracy.

Feature set	prosody	Mel	prosody + Mel
Detection			
1 word	86.5	85.3	86.1
3 words	82.7	81.0	80.8
3 words + posind.	**89.8**	88.3	88.8
Classification			
1 word	85.1	84.4	84.9
3 words	82.5	81.4	81.5
3 words + posind.	**87.3**	86.2	86.7

Table 3.8: Results (accuracy) for phrase boundary tone recognition with leave-one-speaker-out cross-validation. The majority class baseline for both tasks is 80.7% accuracy.

Mel features were as effective as the prosody features is phrase boundary recognition, but only in the speaker-dependent case. Here, we obtained the best results using the combined Mel and prosody feature sets. This may be explained as follows: In the speaker-dependent case, the higher-dimensional feature sets may facilitate overfitting to this specific speaker. When generalizing across speakers, fewer features may be better. Another explanation may be the fact that the Mel features are more speaker-specific. Especially for pitch accents, the drop in performance from a speaker-specific to a speaker-independent setting is quite large using the Mel features. We assume this effect to be magnified by the fact that we did not perform z-scoring in these experiments. Section 3.6 reports results using a normalized version of this feature set.

Speaker	f1a	f2b	f3a	m1b	m2b
detection	88.4	88.8	91.1	91.4	89.3
classification	86.0	86.1	87.7	89.0	87.6

Table 3.9: Phrase boundary recognition accuracies for each speaker using prosody and position features.

Task	pitch accents	phrase boundaries
detection	82.3	90.5
classification	72.2	87.9

Table 3.10: Prosodic event recognition results with 10-fold cross-validation, using the prosody feature set and with 3 word windows and position features.

3.4.4 Effects of normalization

A widely-used measure to help generalize prosodic models across speakers is normalization in the form of z-scoring (Rosenberg and Hirschberg, 2007; Chen et al., 2004; Schweitzer et al., 2010). In our experiments we observed a large drop in performance after z-scoring the features across all utterances for each speaker. This worsened the results for both for the speaker-dependent and the speaker-independent case and this was found to hold across tasks. Table 3.11 only shows the result for the prosody feature set (5 acoustic features), but we observed the same effect for the Mel features as well.

At first glance, this may be attributed to the fact that the CNN looks for relative patterns in the data independent of their absolute position and values; and prosodic events are characterized by relative changes in speech. Normalizing the values may lead to a loss of fine differences in the data since the range of the values is decreased by z-scoring. The CNN performance in our experiments, however, appears to benefit from the original differences, which is why the features were used without normalization. Margolis et al. (2010) also reported a performance decrease after normalizing acoustic features, but did not investigate this issue further.

The results in Table 3.12, however, show that that normalizing per utterance (that is, a recording of one speaker only) does improve the detection results. This may be an effect of differences in the individual recordings in the corpus that were not accounted for previously. Due to these findings, we updated the acoustic feature set by normalizing across each utterance file, and were thereby able to increase the overall performance of the model. We believe that this does not invalidate any other previous findings.

For the experiments in Table 3.12, we also added further updates: The dropout rate

was increased to $p = 0.8$, and the acoustic feature set was extended to 6 features extracted with 20/50ms frames. As shown in the differences in performance levels, the updated parameters led to an increase in detection accuracy. The number of training epochs was reduced to 20 since this was found to be sufficient.

	non-normalized	normalized
pitch accents		
detection	83.6	77.0
classification	69.0	62.6
phrase boundaries		
detection	89.8	83.0
classification	87.3	83.2

Table 3.11: Effects of z-scoring in speaker-independent experiments using 5 acoustic features and position indicator features.

	pitch accents	phrase boundaries
raw features	83.8	88.3
z-scored per speaker	76.9	82.9
z-scored per utterance	84.3	91.7

Table 3.12: Averaged leave-one-speaker-out detection results for pitch accents and phrase boundaries on BURNC (using 6 acoustic features) after normalizing per speaker or per utterance (accuracy in %)

3.4.5 Adding data for phrase boundary detection

As we have shown in in the experiments so far, the performance for phrase boundary detection is lower compared to the respective accuracy than is the case for pitch accent detection. This is most likely attributed to the unbalanced class sizes: Table 3.1 shows that the three datasets contain only up to 20% phrase boundaries, while pitch accents comprise around 50% of the prosodic event labels. This constitutes a problem for our method, since neural networks are especially sensitive to the amount of available training instances. Therefore, we conducted a brief experiment that tested one method of increasing the percentage of labeled phrase boundaries. For phrase boundary detection, we consider only full intonational phrase boundaries (ToBI "%"). The corpora do contain, however, intermediate phrase boundaries (or phrase accents, denoted in ToBI by "-"). Intermediate phrase boundaries are also defined as breaks, but are not as strong

as those at the end full intonational phrases (Veilleux et al., 2006). Although these are, strictly speaking, not the same structural event, the acoustic cues may be similar enough to model both as one class, since both intonational and intermediate boundary tones mark the end of prosodic phrases.

In this experiment, we addressed the question of whether we can improve the intonational phrase boundary detection performance by adding intermediate phrase boundaries as training examples. The test data remained the same, that is, we did not aim to label intermediate boundaries as intonational ones. As in the following section, we used a dropout rate of $p = 0.8$ and the feature set consisted of the 6 normalized acoustic features. These experiments were run using 5-fold cross-validation. The results are shown in Table 3.13. By adding intermediate boundaries, we increased the number of positive labels in the training data from around 19% to around 28%. This corresponds to an increase of almost 50%. The classification accuracy did not increase by a notable amount after training on both intermediate and intonational boundaries, but it did improve the recall and F1-score. It appears that this method helps the model to find more positive examples, at the cost of precision: considering weaker breaks as positive instances has the effect that the model learns to label unclear cases as intonational phrase boundaries. This result also reflects how the performance of the CNN is affected by the class distribution in the training data and shows how important well-balanced training data is for training robust models. Whether this is a suitable strategy for improving phrase boundary detection, however, is open to further investigation.

training labels	boundaries in training data	accuracy	precision	recall	F1-score
BURNC-intnl	19.3 %	92.6	86.6	73.3	79.3
BURNC-all	28.4 %	92.9	84.6	78.0	81.0

Table 3.13: Intonational phrase boundary detection results, trained on either intonational only (BURNC-intnl) or both intonational and intermediate phrase boundaries (BURNC-all)

3.5 Performance across datasets

The question of generalizability is especially important when modeling with deep learning methods, which are frequently prone to overfitting. The previous section described experiments on one single corpus (BURNC). In this secion, we describe experiments

in which we applied the prosodic event detector to two other copora that consist of a different speech genre and language, respectively: BDC and DIRNDL. We first report the within-corpus performance on each dataset using 5-fold cross-validation, and subsequently discuss why we chose this method over leave-one-speaker-out cross-validation. We then perform cross-corpus experiments on each pairing of datasets. These experiments used the updated settings (dropout $p = 0.8$; 6 normalized features) and were performed on the same splits.

3.5.1 Within-corpus performance

All models were evaluated using 5-fold cross-validation. We randomly split the data into training, development and test sets, keeping the splits consistent across each task. 500 held-out words from the training data in each cross-validation split were used as the development set for computing the validation accuracy.

Table 3.14 shows the within-corpus detection performance on the three datasets for pitch accents and phrase boundaries. We obtained good results on BURNC and DIRNDL; around 86% detection accuracy for pitch accents and over 90% for phrase boundaries, which shows that this method is suitable for both English and German data. The BDC dataset appears to be harder to classify using this method, since we were not able to achieve the same performance levels as on the other two corpora. We believe that the reason for the lower performance on this dataset is because of its smaller size ($\approx$ 19k vs. $\approx$ 27k in BURNC), which may be considered a disadvantage to the neural network. Another aspect to consider is that the model was optimized on the BURNC dataset, which consists of comparably "clean" speech compared to BDC.

This table lists not only accuracy, but also precision, recall and F1-score, which are more informative, especially for imbalanced classes, such as phrase boundaries. An observation that we made for all three datasets is that pitch accent detection yields a higher recall and phrase boundaries a higher precision. We assume that this is because the distribution of pitch accents in the data is balanced, while phrase boundaries are more sparse (see Table 3.1. This leads to an overall lower F1-score for phrase boundary detection, as discussed in the previous section.

3.5.2 Evaluation with k-fold cross-validation

The experiments in this section were evaluated using 5-fold cross-validation. This means that the models cannot be regarded as speaker-independent, and the results are not in-

	accuracy (%)	precision	recall	F1-score
pitch accents				
BURNC	86.8	85.5	89.7	87.5
BDC	78.5	74.8	78.8	76.7
DIRNDL	86.6	85.8	88.2	87.0
phrase boundaries				
BURNC	92.5	86.4	73.4	79.3
BDC	90.5	81.7	48.9	61.0
DIRNDL	98.1	96.7	87.1	91.6

Table 3.14: Within-corpus results for pitch accent and phrase boundary detection on BURNC, BDC and DIRNDL.

dicative of how well the models generalize to unseen speakers. In the previous section, we performed leave-one-speaker-out cross-validation to take this issue into account. Using the CNN with the settings used in this section, the pitch accent detection performance on BURNC in leave-one-speaker-out experiments is 84.3% accuracy (84.0 F1-score), and for phrase boundaries 91.7% (74.3 F1-score). As expected, these results are slightly lower than those obtained using simple random 5-fold cross-validation in Table 3.14.

Despite this, we performed most experiments in this work using random k-fold cross-validation for several reasons. The first reason is that the above results, in comparison to related work that use the same evaluation method, still show that our model performs well (we refer to the discussion in Section 3.5.4).

An issue that occurs in most of the datasets using in this work is the fact that there is an amount of lexical overlap across the speaker-specific subsets. This limitation especially concerns news corpora, which Rosenberg (Rosenberg, 2009) points out in the context of prosodic event recognition using lexico-syntactic features on BURNC, but which also applies to the DIRNDL corpus. This may also be a problem when using only acoustic features since reoccurring lexical patterns will also correlate with acoustic patterns, although the effect may not be as large. Separating the datasets for both speaker and lexical overlap would, however, decrease the amount of available data and was therefore not taken to be a suitable strategy.

Consequently, a more reliable judgement of the robustness of models is made by measuring their performance on entirely unseen data, that is, cross-corpus evaluation. Therefore, we evaluate our models across corpora in in the next section.

3.5.3 Cross-corpus performance

The three corpora used in the previous within-corpus experiments represent different genres. BURNC and DIRNDL both consist of radio news speech (that is, read speech) recorded by professional speakers. Hasegawa-Johnson et al. (2005) describe this style as *natural but controlled*. The within-corpus results have shown that this type of data is easier to model. The BDC corpus contains both read and spontaneous speech, but by non-professional speakers and the lexical content is much more limited. Spontaneous speech is always a challenge for speech and language processing due to more noise, disfluencies, hesitations, and incomplete phrases.

In addition to the differences in speech genre, our experiments also compared American English to German data. Since these two languages have similar prosody, it is possible to test models trained on either language against each other. For example, Rosenberg et al. (2012) and Soto et al. (2013) have tested both DIRNDL and BDC in cross-language prosodic event detection.

For these experiments, the training and development sets were both taken from the source corpus and we tested the models on each target corpus. Table 3.15 shows that, as expected, the cross-corpus detection performance is considerably worse than the within-corpus results. Furthermore, it appears that the cross-language task here is more difficult than cross-genre modeling: We obtained better results when training on BURNC and testing on BDC and vice versa, compared to the cross-corpus cases involving German data. The slightly unexpected observation that all models perform worse when testing on the BDC dataset, even in the within-corpus case (models trained and tested on BDC), can be attributed to the large genre differences between the BDC, a smaller corpus that includes spontaneous speech, and the two radio news corpora, which are easier to model using this method.

Since we trained and validated the models on the respective source corpus and thus tested on completely unseen data, the results are expected to be quite low. The results obtained here show that the prosodic event detector performs reasonably well across the BURNC and DIRNDL datasets, however the BDC corpus still poses a challenge. There are several ways to improve the generalizability of machine learning models using domain adaptation, that are not the subject of this thesis. A simple method of improving the results is to validate on held-out data from the target corpus. Table 3.16 shows the result of using this method. Note that since the results here are given in accuracy, the numbers are not directly comparable to Table 3.15. This experiment demonstrates that the results can be improved by around 1-2 percentage points, provided that target-

training \ test	BURNC	BDC	DIRNDL
pitch accents			
BURNC	*87.5*	70.5	82.8
BDC	85.5	*76.7*	81.7
DIRNDL	78.4	59.9	*87.0*
phrase boundaries			
BURNC	*79.3*	52.5	72.8
BDC	63.7	*61.0*	79.5
DIRNDL	60.3	43.0	*91.6*

Table 3.15: Cross-corpus results for prosodic event detection, given in F1-score. The cross-corpus detection results are lower compared to the within-corpus results (shown in italics).

domain development data is available.

training \ test	BURNC		BDC		DIRNDL	
validated on	source	target	source	target	source	target
pitch accents						
BURNC	86.8	–	76.0	76.5	81.4	81.8
BDC	83.9	84.2	78.5	–	78.4	80.8
DIRNDL	79.6	82.2	71.2	73.3	86.6	–
phrase boundaries						
BURNC	92.5	–	86.5	87.1	92.6	94.3
BDC	88.4	88.8	90.5	–	95.1	96.1
DIRNDL	87.9	88.2	85.0	84.9	98.1	–

Table 3.16: Cross-corpus detection results in accuracy (%) after validating on held-out data from either the source corpus or the target corpus.

3.5.4 Comparison with related work

In the previous sections we have shown the baseline within-corpus performance of the CNN-based prosodic event detector, and compared how the model generalizes to unseen speakers and corpora. In the following, we highlight similar related work and compare the performance to our model. We note that the results reported in the literature can

differ in terms of evaluation measure (accuracy or F1) and evaluation procedure (speaker-independent vs. speaker-dependent). We therefore list the respective setup along with each result. Furthermore, we only make direct comparisons to methods that use acoustic features only and that perform prosodic event detection at the word level.

Our results obtained using within-corpus evaluation on BURNC outperform those obtained by Rosenberg (2009) using acoustic features only (Rosenberg provides accuracy measures for pitch accent detection and both accuracy and F1 for phrase boundary detection): For pitch accent detection on BURNC, we obtained 86.8% accuracy, while they reported 85.5% accuracy for their best results using 10-fold cross-validation. Our method also outperforms for phrase boundary detection, where we obtained 92.5% accuracy and 79.3% F1-score, and they reported 89.5% accuracy and 73.6% F1-score (Table 3.14).

On the BDC dataset, we found that the CNN-based method yielded a lower performance than as reported by Rosenberg (2009). They evaluated on the spontaneous and the read speech portions of the dataset separately, and obtained 83.2% and 84.4% accuracy, respectively, for pitch accent detection. For phrase boundary detection, their method produced an accuracy of 93% (80.9% F1) on spontaneous speech and 95.7% (82.6% F1) on read speech. In our experiments, we used both parts of the BDC as one dataset, and obtained an accuracy of 78.5% for pitch accent detection and 90.5% (61.0% F1) for phrase boundary detection.

We believe that the reason for the lower performance on this dataset is because of its smaller size ($\approx$ 19k vs. $\approx$ 27k in BURNC), which may be considered a disadvantage to the neural network. Another aspect to consider is that the model was optimized on the BURNC dataset, which consists of comparably "clean" speech compared to BDC. Further optimization effort in combination with more spontaneous speech data may help to increase the accuracy of our method on this corpus.

Kumar et al. (2008) also reported lower results using acoustic features on the BDC dataset: their method correctly detected 74.5% of pitch accents and 83.5% of phrase boundaries. The results in this study were obtained using leave-one-speaker-out cross validation, which is why the performance cannot be directly compared. However, we can compare their leave-one-speaker-out results on BURNC to ours: They obtained 80.1% accuracy for pitch accent detection and 84.1% for phrase boundary detection. Our results (mentioned in Section 3.5.2) were 84.3% and 91.7%, respectively.

Our results on the DIRNDL corpus are similar to the results reported by (Soto et al., 2013) for phrase boundary detection, namely around 91% F1-score in the within-corpus case (Table 3.14). This study divided the corpus into training and test data without

speaker overlap. Therefore, these numbers are only partially comparable to ours. Nevertheless, their cross-corpus experimental results also reflect the fact that their method was stronger on the BDC within-corpus case: they did not observe a such a large performance drop when training on DIRNDL and testing on BDC. For pitch accent detection, Rosenberg et al. (2012) obtained around 76% accuracy in cross-corpus experiments on BDC and DIRNDL in experiments involving training and testing in both directions. As in the study cited above, they evaluated speaker-independently. Our method achieved a lower performance when training on DIRNDL and testing on BDC, namely around 71.2% accuracy (Table 3.15 shows F1-score only), but higher performance (78.4%) in the other direction.

In sum, although a direct comparison with related work is quite difficult because of the wide range of modeling and evaluation methods, we consider the performance of our CNN-based method on radio news speech (i.e. BURNC and DIRNDL) to be at least as good as (and in some cases outperforming) previous methods. On the BDC dataset, our prosodic event detector was not found to reach the performance levels of Rosenberg (2009), but outperforms the method described in Kumar et al. (2008). While cross-speaker and cross-corpus application of the CNN-based prosodic event detector yields a lower performance than the within-corpus case, our results also fall within range of those obtained using similar previous methods.

3.6 Performance of different acoustic features

In the next set of experiments, we aimed to gain insight into which acoustic features used in the CNN perform better than others, and to whether the model's performance is negatively affected if single features are left out.

3.6.1 Individual acoustic features

We first took a closer look at the impact the individual acoustic features have on the classification performance. For this investigation, we tested the prosodic event detector using only one of the 6 features at a time, along with the position indicator feature. Figure 3.4 compares the results for each acoustic features on the three datasets used in the previous section: BURNC, BDC and DIRNDL. Testing on several datasets helps to make more general observations, but also shows corpus-specific differences. The above figure (3.4a) illustrates the results for pitch accent detection, and the bottom figure

(3.4b) for phrase boundary detection. Table 3.17 lists the results in accuracy and F1-score. These numbers can be compared to the within-corpus results in Table 3.14.

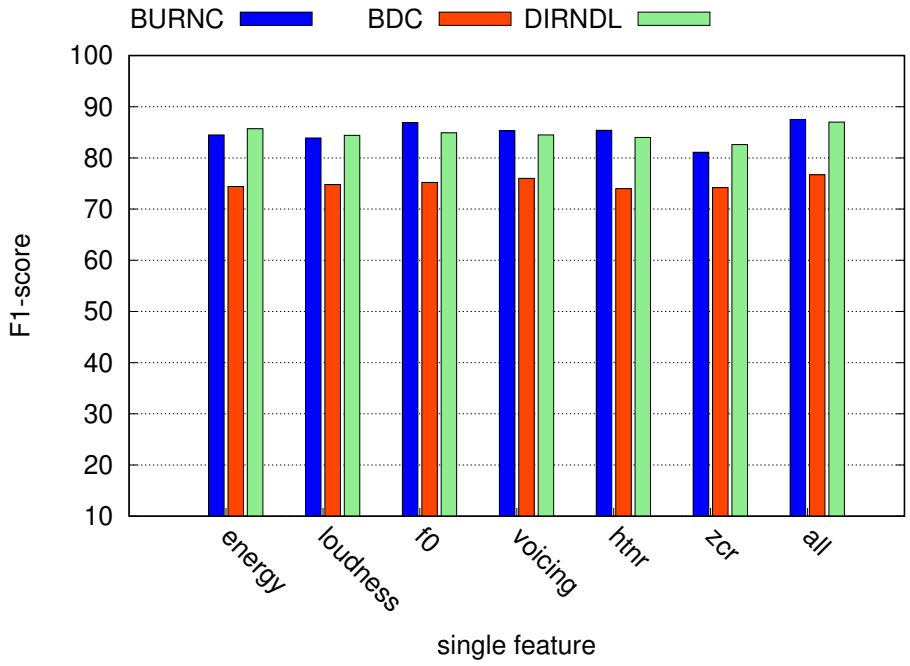
A clear first observation is that despite using only one acoustic input feature, the results for pitch accent detection are still remarkably high (see Figure 3.4a). The results for pitch accent detection are more robust while the performance on phrase boundary detection displays more variation depending on the dataset and feature selection. As discussed in Section 3.4, this is likely an effect of the class distribution. We assume that the fact that the phrase boundary models are weaker in general is responsible for the higher variation in performance when using one feature only.

Comparing the individual features shows that F0 is one of the strongest features for pitch accent detection, but does not perform well for phrase boundary detection. This result is not surprising, since F0 is one of the primary acoustic correlates of pitch accents. However, the variation is not large enough to draw meaningful conclusions for pitch accent detection. For modeling phrase boundaries, energy and loudness appear to be good features, which was also in line with our expectations. While the results clearly show corpus-specific differences, the observed trends are visible across all tested corpora.

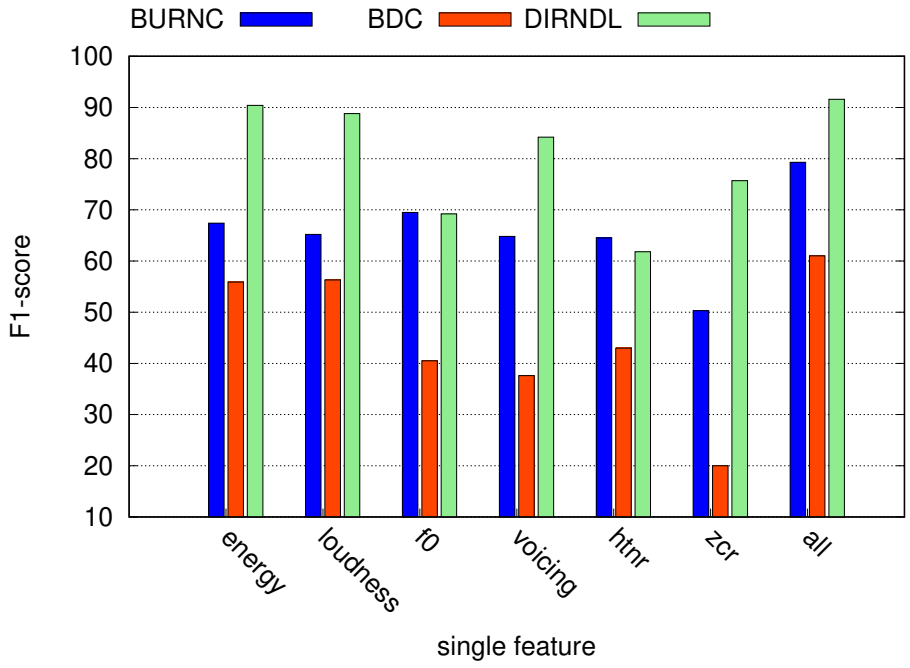
	BURNC		BDC		DIRNDL	
	acc	F1	acc	F1	acc	F1
pitch accents						
energy	83.3	84.5	77.1	74.4	85.1	85.7
loudness	82.4	83.9	76.8	74.8	83.5	84.4
F0	86.4	86.9	77.4	75.2	84.4	84.9
voicing	84.2	85.3	76.2	76.0	83.5	84.5
htnr	84.6	85.4	76.6	74.0	83.5	84.0
zcr	78.5	81.1	74.1	74.2	81.3	82.6
phrase boundaries						
energy	89.5	67.4	89.5	55.9	97.7	90.4
loudness	89.0	65.2	89.4	56.3	97.5	88.8
F0	90.0	69.5	87.6	40.5	92.8	69.2
voicing	88.9	64.8	87.5	37.6	96.4	84.2
htnr	88.8	64.5	87.6	43.0	92.0	61.8
zcr	86.5	50.3	85.9	20.0	94.9	75.7

Table 3.17: Performance of individual acoustic features for prosodic event detection in accuracy (acc) and F1-score.

Table 3.18 shows the performance of the models when leaving out each individual acoustic feature. The results show that the performance drops very slightly when compared to the results obtained using the full feature set (3.14). The variation is so low,



(a) Single features for pitch accent detection



(b) Single features for phrase boundary detection

Figure 3.4: Detection performance (F1-score) for pitch accents and phrase boundaries using single acoustic features. The rightmost bars show the performance using the full feature set (*all*). Pitch accent models are more robust, while phrase boundary models show higher corpus- and feature-specific variation.

however, that it can be concluded that the model is not sensitive to leaving out single features. This may also imply that the features may be slightly redundant, which is also expected, for example, for F0 and voicing probability. Although some features, such as harmonics-to-noise-ratio and zero-crossing-rate, appear to be comparatively weak when used as single features, the best results were obtained using the full feature set. Therefore it is likely that the CNN benefits from a combination of several sources of acoustic information.

left-out-feature	BURNC		BDC		DIRNDL	
	acc	F1	acc	F1	acc	F1
pitch accents						
energy	86.7	87.3	78.4	76.3	86.2	86.7
loudness	86.7	87.3	78.2	75.9	86.0	86.6
F0	86.1	86.7	78.6	75.6	86.2	86.7
voicing prob.	86.8	87.2	78.1	76.5	86.8	87.1
HtN	86.6	87.3	78.1	76.1	86.3	86.7
ZCR	86.8	87.2	78.4	76.3	86.3	86.6
phrase boundaries						
energy	92.7	79.8	90.3	58.9	97.8	90.9
loudness	92.4	79.5	90.3	59.1	97.9	90.9
F0	92.3	77.8	90.4	60.5	97.9	91.0
voicing prob.	92.9	80.0	90.3	59.6	97.8	90.7
HtN	92.6	78.7	90.4	59.8	97.9	90.9
ZCR	92.5	78.6	90.5	59.8	97.8	90.6

Table 3.18: Leave-one-feature-out experiments for pitch accents and phrase boundary detection, shown in accuracy (acc) and F1-score.

We did not pursue the question of what the ideal combination of features would be, or whether specific combinations of features may reach the same performance as obtained when using all features. This would be a question for future investigations. Since the use of the full feature set does not have any effect on the efficiency while yielding the best results, we conclude that this is a suitable choice of acoustic features.

To complete our study on the performance of the acoustic features chosen for the prosodic event detector, the next two sections address the following open questions: The first concerns the generalizability of the Mel feature set and the effect of normalization, mentioned in 3.4. The second question is whether the quality of the F0 features can be improved by using different extraction algorithms.

3.6.2 Mel-spectral features

After updating the model settings, we tested the Mel feature set in cross-corpus experiments. We compared the performance to the prosody feature set (with all 6 features) and across the three corpora. This time, we extracted the same features for 20ms frames, but utterance-normalized the features first, as in Section 3.4.4. Table 3.19 shows the within-corpus results obtained in 10-fold cross-validation experiments and using a dropout rate of $p = 0.8$. Despite the updated settings and normalization, the results obtained using this feature set are worse than the overall performance range of the prosody feature set observed in the previous sections. The Mel features yield an accuracy that is consistently around 1-2 percentage points lower. This also holds for the cross-corpus setting shown in Table 3.20. These findings are in line with the conclusions made in Section 3.4 that the prosody feature set is a better choice for this method.

Mel features	BURNC	BDC	DIRNDL
pitch accents	85.24	76.99	84.32
phrase boundaries	91.18	88.71	97.16

Table 3.19: Performance of the normalized Mel-spectral feature set across the three corpora in accuracy (%).

TRAIN on TEST	Mel	Prosody
BDC on BURNC		
accents	81.05	83.86
boundaries	83.29	88.64
BURNC on BDC		
accents	75.17	76.29
boundaries	85.38	86.70
BURNC on DIRNDL		
accents	78.23	81.50
boundaries	89.66	91.87
DIRNDL on BURNC		
accents	76.73	79.81
boundaries	85.75	87.29

Table 3.20: Cross-corpus performance of the Mel-spectral feature set compared to the prosody feature set, shown in accuracy (%).

3.6.3 F0 features

It has been pointed out by Batliner et al. (2001c) that one of the reasons why F0 features are often found to yield poor performance for prosodic event recognition, despite the fact that they are one of the main acoustic cues of prosody, is the error introduced by F0 estimation methods. In this experiment we therefore tested whether the F0 descriptor used for our prosodic event detector can be improved using other pitch extraction algorithms implemented in openSMILE. The first pitch extraction methods that the toolkit offers are an autocorrelation or cepstrum-based method (ACF) with temporal smoothing[3]. The second method uses subharmonic sampling (Shs) and Viterbi smoothing. Both methods can be combined with a final step that smoothes the values with a moving average filter with a window length of 3 (*sma*). The default method used for our feature set is the Shs method with final smoothing.

We extracted the 4 different types of descriptors from 50ms frames, z-scored them per utterance, and used one feature at a time, in combination with the position indicator, in the CNN. We tested the features for both the BURNC and BDC corpus to control for corpus-specific differences. Table 3.21 shows the results. The largest difference in performance stems from the two pitch extraction methods: the features obtained using subharmonic sampling yield a better accuracy for both pitch accent and phrase boundary detection. The beneficial effect of final smoothing is consistent, but small. While these results do not imply that the descriptors used in our prosodic event detector are devoid of estimation error, it shows that they are the best choice given any readily available alternatives.

3.7 Effects of adding lexical information

As mentioned in Section 2.3, many methods of prosodic event detection use not only acoustic features, but also lexical features derived from the transcriptions. Especially for pitch accents, it has been shown that lexical features are useful since pitch accents tend to correlate with certain words (Yuan et al., 2005; Nenkova et al., 2007; Batliner et al., 2001c). In this section, which was published as a conference paper (Stehwien et al., 2018), we extended the pitch accent detector to include word embeddings (Pennington et al., 2014; Mikolov et al., 2013a). Word embeddings are vector representations of words that are a standard method of encoding text in state-of-the art neural network

[3]As described in the official documentation: https://www.audeering.com/opensmile/

feature	BURNC	BDC
pitch accents		
F0final (shs)	86.43	77.09
F0final_sma (shs)	**86.53**	**77.61**
F0 (acf)	85.15	75.88
F0_sma (acf)	85.20	75.72
phrase boundaries		
F0final (shs)	90.04	87.43
F0final_sma (shs)	**90.16**	**87.65**
F0 (acf)	87.99	86.74
F0_sma (acf)	87.86	86.62

Table 3.21: Performance in accuracy (%) of single F0 features using different extraction methods in openSMILE.

models. Trained on large amounts of data in an unsupervised fashion, neural network algorithms "embed" words into vector space according to their textual context. They follow the idea of *distributional semantics* that similar words appear in similar contexts. Word embeddings have shown to encode both syntactic and semantic similarity (Mikolov et al., 2013b). While research in neural-network based TTS synthesis has made use of word embeddings to predict prominence from text (Wang et al., 2015; Rendel et al., 2016), to the best of our knowledge this paper was the first to test word embeddings for pitch accent detection on transcribed speech. We examined the effect of adding word embeddings to within-corpus and cross-corpus experiments on three English corpora: BURNC, BDC and LeaP.

3.7.1 Experimental setup

The settings for the baseline acoustic model were a combination of previous settings: These experiments used the full set of 6 acoustic (*prosody*) features, but without normalization. The intensity (energy, loudness) features were computed for each 20 ms frame and the remaining features for each 50 ms frame. The dropout rate for the CNN was set to $p = 0.2$. The models were trained usind 10-fold cross-validation for 20 epochs.

Table 3.1 lists the number of words and accented words in each dataset used in the experiments. As in Section 3.4, it should be noted again that the splitting of the corpora into training and test sets were carried out randomly, which means that the within-corpus experiments are neither evaluated independent of speaker nor of lexical content. A fair

amount of lexical overlap can be found especially in the BURNC and LeaP corpora, in which different speakers often read the same text. In the LeaP dataset, around half of the utterance files were recorded from speakers reading out the same story. Since the experiments focus on adding lexical information, we did expect the model to learn observed textual features. For this reason, strictly unseen test data was used only in the cross-corpus experiments.

For the within-corpus experiments, we split the datasets into 10 fixed cross-validation splits, and use 500 held-out words from the respective training split as the development set. In the cross-corpus experiments, the same 10 test sets were used for comparability with the within-corpus versions. Training was carried out on an entire source corpus and the development set consists of 500 held-out words from this dataset. Here, we also tested a setting in which we trained on all three training datasets and tested on a test set from each target corpus.

3.7.2 Lexico-acoustic model

The lexico-acoustic model is an extension of the baseline acoustic model. The extension consists of an additional feed-forward network that learns features from word embeddings, shown in Figure 3.5. The input consists of one word embedding vector per input word. The vector values are used as non-trainable matrix weights. This is fed into the hidden layer, which has only a few output units n that form a "bottleneck". Dropout with $p = 0.8$ is applied to the input before feeding it through the hidden layer. Afterwards, the two output layers of the acoustical and lexical models are regularized separately before they are concatenated into one final feature representation. This is fed into the softmax classification layer.

As input features, we tested two different types of word embeddings: GloVe (Pennington et al., 2014) and word2vec (Mikolov et al., 2013a). The respective algorithms are usually applied to very large datasets to train the embeddings. Since the datasets used in this paper are comparatively small, we used the pre-trained 300-dimensional versions available online[4]. These embeddings were used to represent the words in the datasets. Since our datasets contained words not included in the original pre-trained embeddings, both GloVe and word2vec yielded a number of out-of-vocabulary (OOV) words. We removed noise from the word labels such as special characters and we split contractions, e.g. *she'll* → *she* and hyphenated words, e.g. *eighty-eight* → *eight*. In both cases we

[4]Word2Vec trained on the Google News dataset https://code.google.com/archive/p/word2vec/
GloVe trained on Wikipedia and Gigaword: https://nlp.stanford.edu/projects/glove/

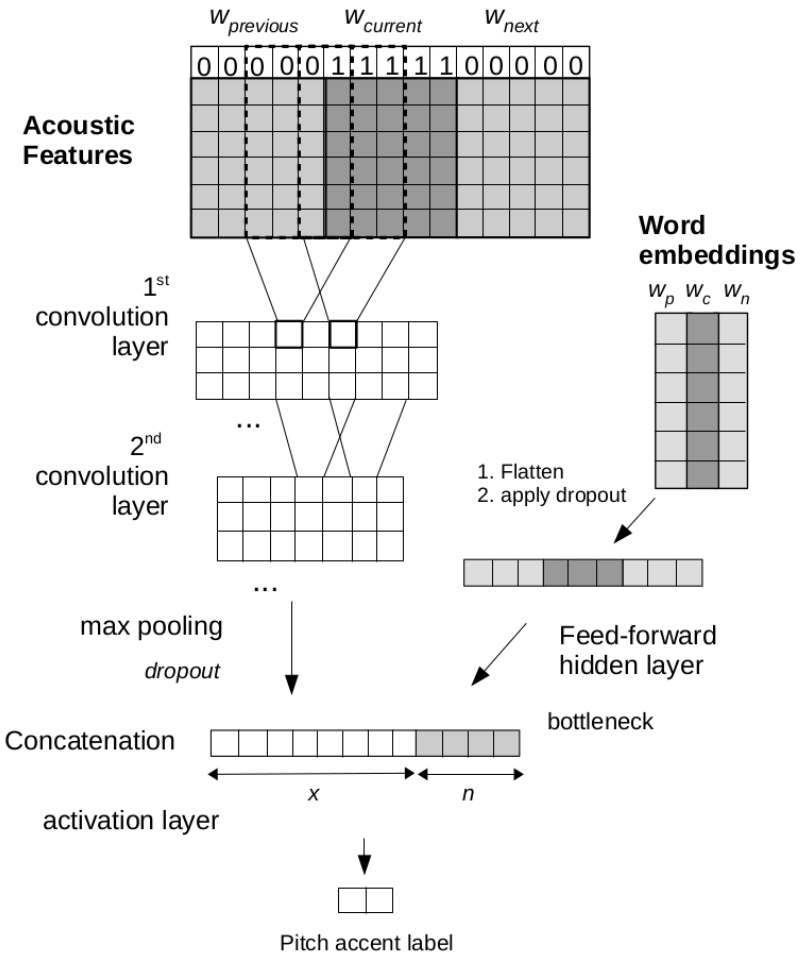


Figure 3.5: Model overview for pitch accent detection using a CNN for acoustic features and extended by adding a feed-forward network for word embeddings, using an input context window of 3 words.

kept the part of the word expected to be relevant for the presence or absence of a pitch accent. This way, the OOV rate was reduced to the numbers shown in Table 3.22.

We set the embedding vectors for OOV words to consist of ones to ensure that it was a unique representation[5]. This way we ensured that OOV embeddings received a separate representation without having to custom-train them. Word embeddings for OOV words can be custom-trained during the experiments if they are used as trainable matrix weights, however, this is not the way they are used in this work since preliminary experiments had shown this method to be less effective.

The pre-trained word2vec embeddings did not include the frequent words *a, and, of* and *to*, referred to as stopwords. We took this difference in to account in Table 3.22. These stopwords make up most of the OOV words in the word2vec embeddings, and are very rarely accented. The BDC corpus contains more accented stopwords, which is likely because of the fact that in direction-giving speech, the word "to" is more frequently accented. The remaining OOV words, often carry pitch accents. The effect of stopwords represented as OOV in word2vec is discussed in Section 3.7.3.

Similar the context window used to create the acoustic input, we also used a context window for the lexical features. We used unigram and trigram word embeddings, that is we used the word embedding vector of either only the current word or along with its two neighboring words. The latter case also required zero padding if no neighboring word is present, as is the case at the end of an utterance or speaker turn.

We tested different bottleneck sizes n, so that the final feature representation along with the acoustic features of size $x = 100$ had the dimensionality $100+n$ (see Figure 3.5). The sizes were set such that the output of the lexical model was much smaller than that of the acoustic model, for example $n = 10, 20, 30$. By restricting the number of lexical features, we aimed to ensure that it did not overpower the acoustic information, as we expected the lexical features to be quite strong based on the effects observed in related work.

3.7.3 Results

Baseline Table 3.23 shows the accuracy obtained in the within-corpus and cross-corpus experiments. As in previous experiments, BURNC was found to be by far the easiest corpus to model, resulting in accuracy levels obtained while training on BDC or LeaP that are higher than those obtained on the respective within-corpus test sets. The BDC

[5]The zero vector was reserved for padding

	BURNC	BDC	LeaP
GloVe OOV			
tokens	233	19	4
types	64	11	4
accent rate	93%	74%	50%
w2v OOV			
tokens	3375	2493	1822
types	231	66	6
stopword rate	86.5%	87%	99.9%
accented stopwords	3%	13%	6%
accented remaining	81.9%	83%	100%

Table 3.22: Out-of-vocabulary words for both word embedding types on all three datasets.

dataset, in contrast, is the most difficult. The accuracy on the LeaP corpus is similar to previously reported numbers obtained at the syllable level (Levow, 2009), however, these experiments are not directly comparable. This results on this corpus are worse than for BURNC, which may be because the speakers were non-professional and non-native, but better than BDC, which can be attributed to the fact that LeaP contains read speech only. The *ALL* experiments, in which we trained on all 3 corpora, show a slight rise in accuracy that does not quite reach the baseline within-corpus performance, except when testing on LeaP. This means that adding data to the training set that does not come from the same domain as the test data slightly harms the performance of the model.

Effect of adding word embeddings First, we compared the effect of adding word embeddings to both the within-corpus and cross-corpus setting. We used unigram word embeddings and 10 bottleneck features because, assuming that these to be to be less corpus-specific than the trigram features since unigrams generally have a higher probability than trigrams. In Table 3.23 we listed only the results using GloVe embeddings because the results using word2vec were similar. We observed an increase in accuracy on all corpora (marked in bold) in the corpus-dependent setting, which shows that the model can learn representations from word embeddings that are helpful for detecting pitch accents. In contrast, the results obtained on cross-corpus experiments led to the conclusion that adding word embeddings to the baseline model cannot reliably improve performance. The results obtained from training on the LeaP corpus showed a large drop in accuracy when using word embeddings. This is an effect of the strong lexical

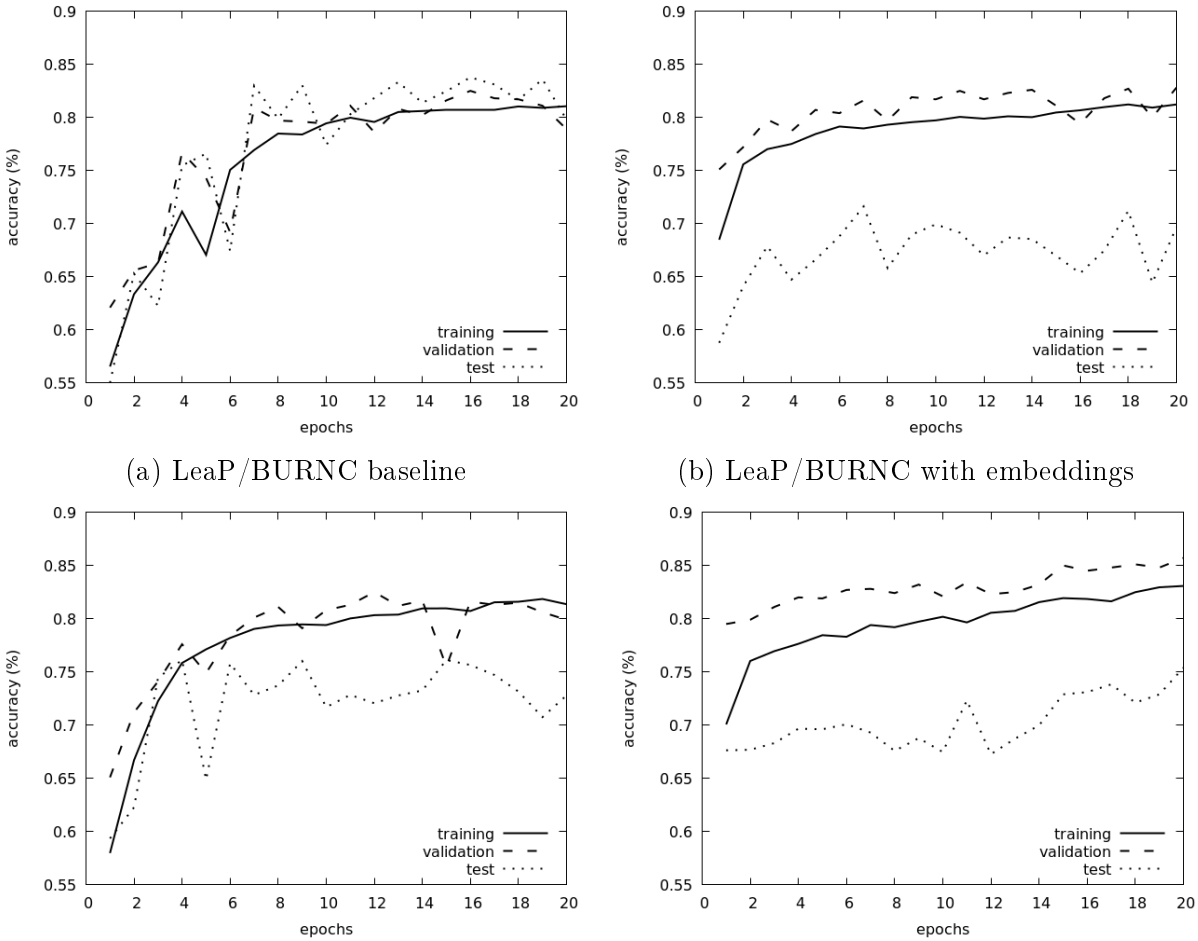
Train \ Test	BURNC	BDC	LeaP
BURNC			
acoustic	87.1	74.2	79.2
acoustic+embs	**87.5**	75.5	78.6
embs-only	*78.5*	*71.6*	*76.0*
BDC			
acoustic	82.3	78.0	76.3
acoustic+embs	82.6	**81.2**	77.5
embs-only	*75.0*	*76.0*	*74.5*
LeaP			
acoustic	82.6	72.1	80.5
acoustic+embs	77.7	73.0	**83.5**
embs-only	*67.7*	*68.0*	*80.9*
ALL			
acoustic	86.6	77.4	80.8
acoustic+embs	87.0	80.6	83.4
embs-only	*75.2*	*72.7*	*77.6*

Table 3.23: Results (accuracy in %) for within-corpus and cross-corpus pitch accent detection using the baseline acoustic model, with added unigram GloVe word embeddings and 10 bottleneck features, and using only the embeddings as features (shown in italics).

features causing the model to overfit to the respective training corpus.

Figure 3.6 illustrates this effect when the accuracy is plotted during training, displaying diverging training and test curves. In Figure 3.7, we show a different effect observed when adding embeddings to the setting in which we trained on BURNC and tested on BURNC: in this case, the learning curve is more stable, and the cross-corpus accuracy using embeddings is sightly higher (82.6% vs. 82.3%, see Table 3.23).

A larger improvement on BDC and LeaP was be observed when target corpus data held out from the test set is seen in training (*ALL*). In fact, the performance of *ALL* on LeaP was as high as the within-corpus results using word embeddings. The results on BURNC, however, were largely unaffected by these changes. The embeddings-only results were added to give an impression of how these features would perform alone. In all cases, these features alone are not as strong as the acoustic features used alone, except on the LeaP corpus. We consider this finding to confirm that the lexical overlap in LeaP greatly facilitates the detection of pitch accents.



(a) LeaP/BURNC baseline

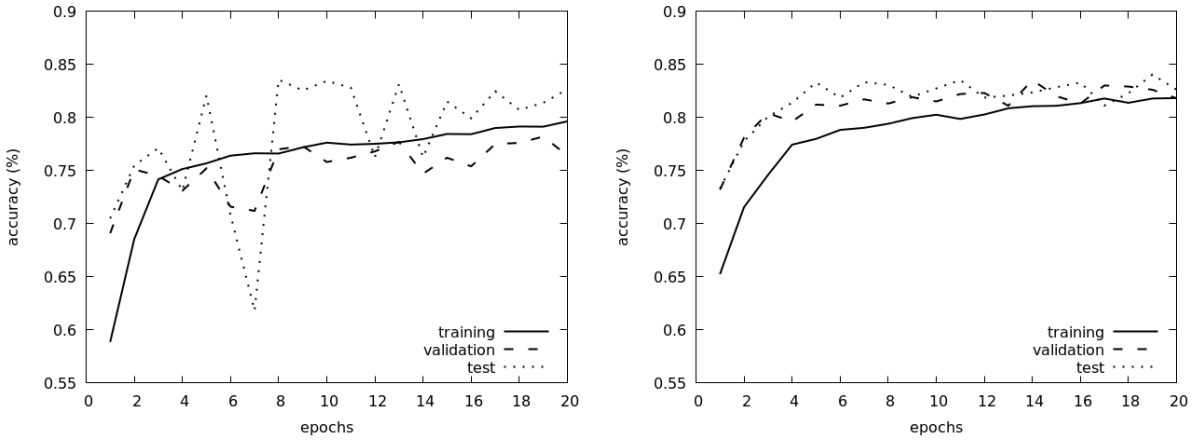
(b) LeaP/BURNC with embeddings

(c) LeaP/BDC baseline

(d) LeaP/BDC with embeddings

Figure 3.6: Training, validation and test accuracy (%, y-axis) during 20 training epochs (x-axis) for pitch accent detection using *glove* word embeddings (1 word, 10 units). Plots 3.6b and 3.6d show overfitting effects compared to the respective baselines in plots 3.6a and 3.6c, visible as diverging training and testing curves. This is also reflected in the final cross-corpus performance.

Effect of N-gram and bottleneck sizes In the next experiment we tested different variations of the lexical feature extension. Table 3.24 shows the results of adding unigram and trigram embeddings to the baseline acoustic model, and of using different bottleneck sizes n. We found that the word embeddings improved the accuracy over the baseline on all corpora and across all tested settings. Overall, the results do not show a specific pattern. Therefore, we can conclude that neither the choice of N-gram and bottleneck size, nor the embedding type is critical in our experiments. The corpora that benefit the most from these lexical features are LeaP and BDC. In this case we observed an increase of around 3 percentage points in all experiments. We assume that this effect is caused



(a) BDC/BURNC baseline

(b) BDC/BURNC with embeddings

Figure 3.7: Training, validation and test accuracy during 20 training epochs for pitch accent detection using *glove* word embeddings (1 word, 10 units). Plot 3.7b shows a more stable learning curve compared to plot 3.7a, which results in a slightly improved cross-corpus performance.

Corpus	BURNC		BDC		LeaP	
Embeddings	glove	w2v	glove	w2v	glove	w2v
unigram						
$n = 10$	87.5	87.6	81.2	80.6	83.5	83.9
$n = 20$	87.4	87.7	81.5	81.1	83.6	83.8
trigram						
$n = 10$	87.7	87.7	82.4	81.1	83.9	83.6
$n = 30$	87.8	87.5	82.7	81.4	83.7	83.8

Table 3.24: Results (accuracy in %) for within-corpus pitch accent detection with word embeddings using unigrams, trigrams and varying bottleneck sizes.

by the lexical overlap in LeaP and the lower baseline accuracy in BDC.

To provide more insight into these numbers, we also show the precision, recall and F1 scores for the accent class in Table 3.25 when using unigram embeddings and 10 bottleneck features. The word embeddings lead to a gain in F1, precision and recall in almost all cases.

Performance on stopwords As mentioned above, the two embedding versions differ with respect to the representation of certain stopwords. In word2vec, these were represented by a "dummy" representation, that is, a vector of ones. Since these words were not OOV words in GloVe, they were part of the standard vocabulary and thus represented

Experiment	Precision	Recall	F1
BURNC			
baseline	88.0	86.7	87.3
w2v	87.2	88.9	88.1
GloVe	88.1	87.7	87.7
BDC			
baseline	74.5	77.8	76.0
w2v	77.6	80.9	79.2
GloVe	77.4	82.5	79.8
LEAP			
baseline	76.2	80.4	78.1
w2v	79.7	84.6	82.0
GloVe	79.8	82.8	81.2

Table 3.25: Precision, recall and F1 scores for the accent class using unigram word embeddings and 10 bottleneck features.

	BURNC	BDC	LeaP
baseline	98.2	88.9	86.9
GloVe	98.2	92.7	94.2
w2v	97.8	92.7	94.3

Table 3.26: Accuracy (%) of the lexico-acoustic model on stopwords, using unigram embeddings and 10 bottleneck features.

with an actual embedding vector. We were interested whether this difference has any effect on pitch accent detection results and therefore computed the labelling accuracy on these stopwords. These numbers are shown in Table 3.26. On the LeaP and BDC datasets, we did not observe a large difference: both types of embeddings improved the performance on stopwords. This means that they are more likely to be labelled correctly (which is usually not accented) when there is some information about the word identity available, regardless of how they are represented as vectors. The results on BURNC, however, do not show a clear pattern. In general, these numbers appear to reflect the overall increase in performance obtained by adding the lexical information: LeaP and BDC benefit the most from adding word embeddings.

3.7.4 Discussion

The within-corpus experiments show that a simple extension of the neural network to include word embeddings can help the model can use to learn useful features that cor-

relate with pitch accents, as the gain in accuracy in these cases demonstrates. However, the fact that the more textual overlap is found in training and test data (as in the case of the LeaP corpus), the better the increase in performance, must be kept in mind when using lexical information.

As shown in Table 3.23, the cross-corpus case is already challenging for the model. Adding the word embeddings has the potential to amplify this affect and bears the risk of overfitting. The observation that text-based features can increase cross-genre error was also made by Margolis et al. (2010).

One option that may help avoid this issue is to create more general lexical features, for example, by using more abstract part-of-speech tags, as used in Schweitzer (2010). The challenge of this strategy is that part-of-speech tags require a standard tagset, which makes it difficult to readily apply them to other datasets, and it adds a further preprocessing step to the prosodic event detector. Due to these reasons, and the findings in this section, we do not recommend the use of lexical features for our method of prosodic event detection.

3.8 Effects of adding temporal information

Apart from lexical features, temporal features are also part of a typical feature set for automatic prosodic event detection. Pitch accents often occur on content words, which tend to be longer; and prominence itself can cause lengthening of certain segments. Phrase boundaries often occur before pauses and breaks and cause pre-boundary lengthening at the subword level. Therefore, adding features that describe the duration of words usually increases the detection performance (Batliner et al., 2001c).

Our model does not explicitly use temporal features, such as word duration or the presence of pauses, for two reasons: The primary reason is that since these are not frame-wise representations, but rather global features that refer to entire segments of speech, they are not suitable as input to a CNN. Secondly, the performance of the model is still quite high using the acoustic features only. The experiments so far, however, have not ruled out for certain that temporal information has an effect on the model. Therefore, in this section, we investigated how word duration and the presence of pauses can be included in the prosodic event detector and show experimental results.

Furthermore, we aimed to verify an interesting assumption, namely that the frame-based input already provides information on word duration. This is because the size of the input words is implicitly marked by the position indicators. The experiments in this

section also served as a first quantitative analysis towards our assumption that the CNN may be learning temporal information on its own.

3.8.1 Preprocessing pauses

For our method, we do not have to perform any additional preprocessing steps to extract pauses or regions of silence, since the time-aligned transcriptions also contain pauses. The default setting in our experiments "ignores" these pauses since we found that these were not required to reach good performance levels. It is also likely that the CNN can learn to identify pauses since they are indicated by the acoustic descriptors having either zero or very low values.

In this section, we tested two settings. The original preprocessing method that ignores pause information is the method that was used in all previous sections. Since the transcription format provided the ending timestamp for each word, any pauses in the recorded speech were included in the input matrix pertaining to the following word. This ensured that the original sequence of speech signal frames was preserved and no non-speech regions, which also provide cues to prosodic boundaries, were omitted. Furthermore, the acoustic context always included actual words, with the drawback of having pauses within certain word boundaries.

We compared this to a setting that makes use of pause information by using the pause transcriptions provided with the corpus annotations. We treated pauses as words and used them to create the 3-word input matrices, but did not include them in the final dataset as data points to be labeled in order to keep the original word count and class proportions. This means that pauses could appear as $w_{previous}$ or w_{next} in the context windows but not as $w_{current}$. This way, we still included any non-speech regions that were part of the original signal but made sure they could be indicated as separate context "words" by the position indicators.

Preprocessing pauses differently yielded slightly different absolute word counts due to minor inconsistencies, resulting in differences of up to 10-20 words per dataset but not changing the class proportions. For this reason, different random cross-validation splits were used for pitch accents and phrase boundaries and for the versions with and without pauses. The splits were kept consistent across each task.

3.8.2 Comparison to a feed-forward network with aggregated features

An effect that we expected to observe if the CNN is already learning to encode duration is that this method should not be affected when duration is added as an extra feature. We tested this assumption by comparing the CNN to a model that does not have access to this information: a simple feed-forward neural network (FFNN) that uses acoustic features computed for entire words as input. This model does not serve as a baseline detection method but rather as a model for comparing the CNN, which uses frame-based input, to a method that uses features which do not represent the duration of segments in any form. These features are manually pre-computed for entire words and are thus also referred to as "aggregated" features Rosenberg (2009). Thus, any temporal information indicated e.g. by the number of frames per word is discarded. The total set of features is a fixed-length vector representation for each word. Therefore, this type of method does not require position indicators.

The manually aggregated features used in the FFNN were derived from the same frame-based acoustic features used for the CNN. The acoustic feature set consisted of the 6 prosody features extracted for 20ms and 50ms frames and subsequently z-scored. We computed four aggregation types: minimum, maximum, mean and standard deviation of each feature for each of the 3 words in the input matrix. Similar features have been used by Rosenberg et al. (2012) and Soto et al. (2013). Our feature set comprises $3*6*4$=72 features. Additionally we computed $2*6*4$=48 delta features consisting of the absolute difference between each aggregated feature across the two word boundaries the context window, respectively. In total, the input vector consisted of 120 features. The FFNN consisted of one fully-connected layer shown in Figure 3.8. The network was trained in the same way as the CNN, with $l2$ regularization and dropout with $p = 0.8$ applied before the hidden layer.

In order to test the effects of adding explicit duration information, we concatenated duration features with the acoustic feature vector in both the FFNN and the CNN. Therefore, we adapted the CNN model slightly to match the FFNN setup without notably affecting the baseline performance: We increased the amount of the CNN kernels to 120 in all of the following experiments to match the input vector size in the FFNN. In the FFNN, the duration features were added directly to the aggregated feature vector. In the CNN, they were concatenated to the output feature representation after max pooling.

We compared the effect of two types of duration features: the raw duration of each word measured in seconds, and normalized duration. We normalized by dividing the raw duration by the number of characters in the word to obtain a rough estimate of whether a word is lengthened acoustically compared to words of similar character length.

We also tested what effect the two modes of preprocessing pauses described in Section 3.8.1 have on the performance of the CNN and the FFNN. We assumed that separating pauses from words in the input matrix should have a larger effect on the FFNN, since this will affect the values of the aggregated features. Ideally, pauses would not be taken into account when computing aggregated acoustic features across words. However, since this experiment serves as a "proof-of-concept" comparison to the CNN, we computed the aggregated features from the same frame-based input, which did contain pauses in the original setting.

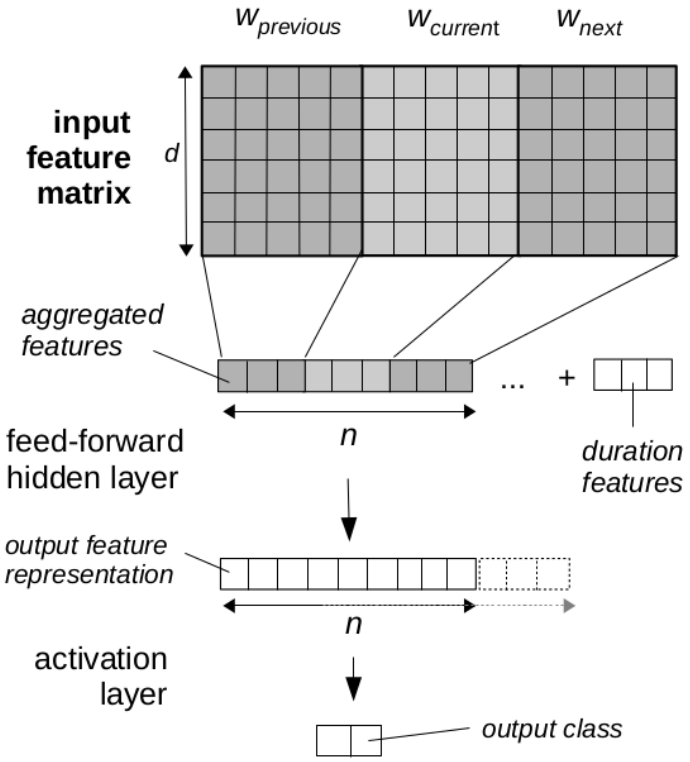


Figure 3.8: Feed-forward neural network (FFNN) that uses manually aggregated acoustic features and optional duration features as input. The input features are derived from the same acoustic feature matrix that is used for the CNN but are a fixed-length vector representation.

3.8.3 Experimental results

Tables 3.27 and 3.29 each compare the results of adding raw and normalized duration to the CNN and the FFNN for pitch accent and phrase boundary detection, respectively. Table 3.28 and 3.30 show the results when using pause information. Since all experiments were repeated 3 times, the results were tested for statistical significance using a paired t-test as a difference of means.

Our general observation is that, as expected, the duration features have no notable effect on the performance of the CNN. In contrast, raw duration helps the FFNN the most; both the pitch accent detection as well as phrase boundary detection models benefit significantly from this information. Normalized duration can only significantly improve the performance of phrase boundary detection. These results are in line with our assumption that the FFNN, as a model with no access to temporal information, benefits from adding duration as a feature. The fact that the CNN does not may suggest that this model can learn duration information on its own. Since phrase boundaries are generally regarded to be correlated with lengthening effects to a larger extent than pitch accents are, it is also not surprising that phrase boundary detection was found to benefit more from the duration features than pitch accent detection, in the sense that both feature types (raw and normalized) can significantly improve the FFNN performance. We discuss this further in Section 3.8.5.

The two settings in which pauses were treated differently during preprocessing did not show a consistent improvement for the CNN model on BURNC and DIRNDL (compare Table 3.27 with 3.28 and Table 3.29 with 3.30). The pitch accent detection models trained on BDC, however, appear to benefit from using pause transcriptions. When using the FFNN, using pause information improved the results slightly in the case of phrase boundary detection and for pitch accent detection on BDC. The fact that the CNN shows no improvements supports our hypothesis that the CNN may be learning to implicitly recognize regions of pauses and silence. Additionally, the FFNN is likely more sensitive to how pauses are included since the aggregated features are less precise if pauses are not treated as separate words. For example, not separating a non-speech region from a word would lead to a lower average energy.

The CNN model yields a higher performance than the FFNN model on all corpora when using acoustic features only, demonstrating the advantage of using a CNN with frame-based features over a simple network with aggregated features. The fact that the CNN is a larger model should also be taken into account: the model has around 15k parameters and is thus smaller than the CNN model, which has around 63k parameters

when using 120 output features. The original baseline CNN with 100 output features has around 44k parameters. The difference in performance, however, is not very large, especially for pitch accent detection, and further decreases when duration is added to the FFNN. This finding shows that well-motivated manual features, in this case aggregations, can be sufficient to obtain good results. The fact that the CNN still generally outperforms the FFNN also suggests that there is useful information that the CNN is learning that is not encoded in the aggregated features. The DIRNDL phrase boundary results are an exception in these experiments; as the results do not change at all regardless of what features are added. We assume this may be an effect of differences in the dataset, but remains a topic to be explored in future research.

feature combination	BURNC		BDC		DIRNDL	
	acc	F1	acc	F1	acc	F1
CNN	87.0	87.5	78.3	76.8	86.6	87.0
+ raw dur	86.5	87.4	78.1	77.4	86.3	87.0
+ norm dur	86.6	87.4	78.0	77.1	86.2	86.9
FFNN	86.0	86.7	76.0	73.8	84.8	85.1
+ raw dur	87.0*	87.4*	78.2*	76.0*	87.1*	87.3*
+ norm dur	86.0	86.6	76.3	73.6	85.1	85.3

Table 3.27: Pitch accent detection results (accuracy and F1-score) with and without raw and normalized word duration features in the FFNN and CNN models. *$p < 0.05$ according to a paired sample t-test between each *+ dur* model and the respective baseline (CNN or FFNN).

feature combination	BURNC		BDC		DIRNDL	
	acc	F1	acc	F1	acc	F1
CNN using pauses	87.1	87.3	80.9	79.4	86.8	86.9
+ raw dur	87.1	87.7	80.0	79.0	86.4	86.9
+ norm dur	87.0	87.7	80.4	79.0	86.6	87.0
FFNN using pauses	86.0	86.4	78.1	74.5	84.9	85.3
+ raw dur	86.8*	86.9	81.3*	79.0*	86.7*	86.9*
+ norm dur	86.1	86.4	78.6	74.5	85.2	85.5

Table 3.28: Pitch accent detection results (accuracy and F1-score) with and without raw and normalized word duration features in the FFNN and CNN models, using pauses.

feature combination	BURNC		BDC		DIRNDL	
	acc	F1	acc	F1	acc	F1
CNN	92.7	79.6	90.6	62.5	98.1	91.7
+ raw dur	92.0	77.1	90.2	60.7	97.9	91.1
+ norm dur	92.1	77.5	90.1	61.4	98.0	91.1
FFNN	88.7	66.8	90.0	58.1	97.1	87.2
+ raw dur	89.6*	69.7*	91.1*	66.1*	97.2	87.8
+ norm dur	89.7*	68.9*	91.0*	63.5*	97.2	87.8

Table 3.29: Phrase boundary detection result with and without raw and normalized word duration features in the FFNN and CNN models.

feature combination	BURNC		BDC		DIRNDL	
	acc	F1	acc	F1	acc	F1
CNN using pauses	93.0	80.4	91.0	65.7	97.5	89.7
+ raw dur	92.4	78.6	90.5	63.5	97.5	89.6
+ norm dur	92.4	79.0	90.9	65.3	97.5	89.6
FFNN using pauses	89.9	70.9	90.5	65.4	97.5	89.6
+ raw dur	90.6*	74.0*	91.7*	70.0*	97.5	89.8
+ norm dur	90.3	72.7*	93.0*	76.0*	97.5	89.8

Table 3.30: Phrase boundary detection results with and without raw and normalized word duration features in the FFNN and CNN models, using pauses.

3.8.4 Further effects of temporal features

Generalization of duration features Since duration features did not have an effect on the within-corpus experiments, we also tested whether they have the potential to improve the cross-corpus performance of the CNN. Table 3.31 shows the generalization performance of raw duration features on the CNN when using pause information. The results show an increase in detection accuracy in the cases where the cross-corpus performance drops considerably with the baseline model. For example, for pitch accent detection, when training on DIRNDL and testing on BURNC, the accuracy drops from around 87% when using BURNC to almost 81% when using DIRNDL. When adding duration, however, the accuracy is increased to 83.2%. In the case of phrase boundary detection, the within-corpus baseline for DIRNDL, for example, is 97.5% accuracy. When training on BURNC and testing on DIRNDL, we obtain only around 92%. The accuracy rises to 94% if raw duration is added. We can conclude that since words with prosodic events are generally longer than those without is a property of all datasets, word duration appears to add additional helpful cues to prosodic events when the acoustic information is too corpus-specific. We also tested the effects of normalized duration, but

training \ test	BURNC		BDC		DIRNDL	
setting	baseline	raw dur	baseline	raw dur	baseline	raw dur
pitch accents						
BURNC	87.1	–	77.6	78.8	82.3	81.7
BDC	84.3	83.4	80.9	–	75.4	78.1
DIRNDL	80.7	83.2	72.4	75.8	86.8	–
phrase boundaries						
BURNC	93.0	–	86.7	86.9	91.9	94.0
BDC	88.8	89.2	91.0	–	93.8	95.7
DIRNDL	87.8	87.8	84.3	84.4	97.5	–

Table 3.31: Cross-corpus detection results (accuracy in %) of the CNN with raw duration features (using pauses).

as this lead to the same outcome, we did not include the numbers in this table.

Pause duration features The above experiments tested two different ways of preprocessing pauses and we assumed that further pause information is not necessary. The fact that the acoustic features values indicate regions of silence was taken to be sufficient for the model, since the CNN should learn to identify these regions. To test whether explicit pause information is useful for the model, we added the duration of pauses in the same way that we added the duration of words. No pause is represented by a duration of zero. Table 3.32 shows the results of added pause duration to the CNN and the FFNN for pitch accent detection. We did not test these results for statistical significance, but refer to the main results in the previous section that show that these models require a performance increase of around 1 percentage point to be considered as improved. Neither the CNN nor the FFNN are affected by adding pause duration. The only notable improvement is found in Table 3.33 for phrase boundary detection using the FFNN. In this case, we observed an increase in F1-score on all corpora, but mainly for BURNC and BDC (these results are marked in bold). This result is not surprising, since phrase boundaries correlate with pauses. Similar to the conclusions made in the previous section, the finding that adding pause features to the CNN indicates that it is redundant information that the model can learn on its own.

Combining temporal features The *+ all* setting was added to show the effects of combining all additional temporal features: raw duration, normalized duration and pause

feature combination	BURNC		BDC		DIRNDL	
	acc	F1	acc	F1	acc	F1
CNN using pauses	87.1	87.3	80.9	79.4	86.8	86.9
CNN + pause dur	87.2	87.7	80.5	79.5	86.6	87.2
CNN + all	87.4	87.8	80.3	79.3	86.4	87.1
FFNN using pauses	86.0	86.4	78.1	74.5	84.9	85.3
FFNN + pause dur	85.8	86.3	78.4	75.4	84.7	85.0
FFNN + all	87.0	87.4	81.4	79.1	86.9	87.0

Table 3.32: Experimental results (accuracy and F1-score) after adding pause duration features for pitch accent detection in the FFNN and CNN models, using pauses. The *all* setting combines raw and normalized duration as well as pause duration.

feature combination	BURNC		BDC		DIRNDL	
	acc	F1	acc	F1	acc	F1
CNN using pauses	93.0	80.4	91.0	65.7	97.5	89.7
CNN + pause dur	91.7	73.8	90.9	62.7	97.5	89.7
CNN + all	92.5	78.8	92.1	70.7	97.5	89.7
FFNN using pauses	89.9	70.9	90.5	65.4	97.5	89.6
FFNN + pause dur	89.3	**72.5**	**93.5**	**78.1**	97.6	**90.3**
FFNN + all	91.1	75.3	94.0	80.6	97.7	90.3

Table 3.33: Experimental results (accuracy and F1-score) after adding pause duration features for phrase boundary detection in the FFNN and CNN models, using pauses. The *all* setting combines raw and normalized duration as well as pause duration.

duration. The results improved for pitch accent and phrase boundary detection using the FFNN, and were better than using raw duration only. One difference that this effect may be attributed to is the fact that more features were added (that is, 9 features in total), giving more weight to the temporal information relative to the acoustic information in the network. Again, this only leads to an improvement for the FFNN, and not on the CNN.

3.8.5 Discussion

The fact that raw duration was found to be a helpful feature for pitch accent detection while normalized duration was not can be explained by the correlation between pitch

accents and absolute word length, which has previously been discussed in the context of automatic detection by Batliner et al. (2001c) and Rosenberg (2009). Absolute duration can indicate how likely a word carries a prosodic event, since this correlates with word identity or part-of-speech (Batliner et al., 2001c; Nenkova et al., 2007). To show that this also holds for our datasets, we considered the average duration of words with and without prosodic event labels in each corpus (Table 3.34). The word durations in this table include possible pauses (as in the original setting), however, when leaving pauses out, the numbers are similar. This confirms that the words carrying prosodic events are generally longer than those without: the average difference in length between words with and without prosodic events is roughly 20ms (this is also in line with the findings reported by Rosenberg (2009)).

In our experiments, normalized duration was found to only significantly improve the performance of phrase boundary detection, and not pitch accent detection. This may be because duration in general can be considered more useful for phrase boundary detection due to pre-boundary lengthening, however these experiments alone cannot provide further support for this assumption. Furthermore, normalized duration is usually a measure of whether a word is prosodically lengthened compared to its "usual" acoustic form. In these experiments, we used only a rough computation, that is, the word duration divided by the number of characters in the word. It is therefore possible that this measure is too noisy to serve as a useful feature for pitch accent detection.

We observed the largest effect of using pause transcriptions for preprocessing on the BDC corpus, especially when combined with duration features in the FFNN. We considered specific corpus properties to gain some insight into these results. Table 3.35 shows the average length of pauses compared to the average length of words in the three datasets. BDC is the only corpus where the average pause is longer than the average word, which may be explained by the presence of spontaneous speech data in the corpus. We assume that this is why the models trained on this corpus are better when pauses are used (that is, treated as separate words). As an additional effect, this also makes the duration features more effective, since pauses do not increase the word length when included separately, and since the duration features also includes pause durations.

It appears that our experimental findings are in part in line with previous findings, but also display some differences. Related work had shown that temporal features are especially good predictors of phrase boundaries. Nöth et al. (2002) found that the majority of phrase boundaries can be predicted using duration features only, however, they performed detection at the syllable level. Soto et al. (2013) reported that boolean si-

	pitch accents	none	phrase boundaries	none
BURNC	0.46 (0.43)	0.24 (0.18)	0.52 (0.50)	0.30 (0.28)
BDC	0.42 (0.34)	0.22 (0.14)	0.51 (0.42)	0.27 (0.20)
DIRNDL	0.61 (0.55)	0.36 (0.24)	0.57 (0.53)	0.47 (0.38)

Table 3.34: Mean (median) length in seconds of words with and without positive event labels

	pauses	words
BURNC	0.16 (0.10)	0.32 (0.29)
BDC	0.45 (0.32)	0.24 (0.20)
DIRNDL	0.32 (0.18)	0.40 (0.35)

Table 3.35: Mean (median) length of pauses and words and in the three corpora

lence features were the single best predictors of phrase boundaries across several corpora. From our results, however, we cannot draw such strong conclusions. In fact, the temporal information added to the FFNN appeared to be relatively weak. Our experiments did not investigate the relative importance of the temporal features compared to the acoustic features in the FFNN, since the main goal was to investigate, and further motivate, the use of the CNN-based prosodic event detector. The question of the role of duration in the CNN is further addressed in Section 5.

3.9 Comparison to a recurrent neural network

The prosodic event detector developed for this thesis is based on a convolutional neural network, which leads to the question of how a recurrent neural network would perform on this task. Both are popular in speech processing, but are two very different learning methods: A CNN learns patterns irrespective of their position in the frame-based representation of the speech signal, while a recurrent network processes the input in a sequential manner, thereby learning to model the data over time. An recurrent neural network therefore inherently learns to model data in context, while a CNN focuses on more local two-dimensional patterns. The use of both models can be motivated for prosodic event detection: For example, determining whether or not a word is pitch accented involves finding prominent syllables, which have no fixed position in a word, and therefore a CNN may be a suitable method. The sequential property of speech, however, also makes a recurrent network an intuitive choice. Both types of models have previously

been used for prosodic event detection (see Section2.3). In this section, we compare the performance of the CNN to that of an RNN, namely a long short-term memory network (LSTM).

Long short-term memory network We chose an LSTM because of its superior performance over a simple recurrent network (we had tested both). In order to obtain two directly comparable models, we ran the LSTM on the identical input features. That is, the LSTM takes the same 3-word input matrix as the CNN, but the input is processed frame by frame. The model learns a representation across the entire input and performs the same classification at the end. We used one LSTM layer with 100 units and tan$h$ activation. The network was regularized and trained ($l2$; Adam) in the same manner as the CNN, but without dropout. We trained the model for only 10 epochs since the training time was significantly longer than for the CNN, although the number of parameters is comparable (43k; the CNN has around 44k parameters) and because we found that the LSTM did not require more epochs to match the performance levels of the CNN.

Within-corpus performance We first compared the two models in terms of pitch accent and phrase boundary detection performance on all three datasets used in the previous sections. Table 3.36 lists the accuracy, precision, recall and F1-score of the CNN for reference, using the updated settings (6 features, 20/50ms frames, dropout $p = 0.8$). These numbers differ slightly from previously reported results due to the random initialization and the fact that different 5-fold cross-validation splits were used. The same splits were used to obtain directly comparable results of the LSTM, given in Table 3.37. It is interesting to find that both the CNN and the LSTM yield quite similar detection performance. Only the results on BURNC are slightly higher in terms of F1-score when using the LSTM. Whether this is because this corpus contains repeated prosodic patterns over longer stretches of time that the sequential model can capture, would be a question that requires further investigation, and is not pursued further in this thesis.

Cross-corpus performance Since the within-corpus performance of the LSTM appears to be very similar to that of the CNN, we also tested if this holds for the cross-corpus case as well. We ran the models with the same settings as in the within-corpus case, and used 500 words from the respective target corpus for validation. Table 3.38 lists the results for the CNN and the results for the LSTM are in Table 3.39. As in the within-corpus

case, the numbers are generally comparable, but we do observe some differences: The LSTM models trained on BURNC and DIRNDL perform better than the CNN versions, while the LSTM trained on BDC is worse in most cases. Since we cannot ascertain from these mixed results whether this means that the LSTM is a superior model, we can only conclude, as in the previous section, that both models behave very similar.

	accuracy	precision	recall	F1
pitch accents				
BURNC	87.2	87.3	87.7	87.5
BDC	80.5	76.9	81.1	78.8
DIRNDL	86.5	85.0	89.1	87.0
phrase boundaries				
BURNC	92.3	88.3	69.3	77.5
BDC	90.1	78.9	53.7	63.8
DIRNDL	97.4	93.0	85.6	89.1

Table 3.36: Prosodic event detection using the CNN on all corpora. The models were trained for 20 epochs.

	accuracy	precision	recall	F1
pitch accents				
BURNC	88.0	87.8	89.1	88.4
BDC	79.9	76.0	81.1	78.4
DIRNDL	86.0	85.4	87.4	86.3
phrase boundaries				
BURNC	92.4	84.1	74.6	79.1
BDC	90.2	76.7	52.4	62.0
DIRNDL	97.5	92.1	87.5	89.7

Table 3.37: Prosodic event detection using the LSTM on all corpora. The models were trained for 10 epochs.

Discussion The most relevant question that arises from these results is the question why these two entirely different learning methods yield almost the same results. This may be caused by fact that both methods operate on identical input information, namely frame-based acoustic features. We added the position indicator features to the input of both models, although one may assume that the sequential model does not require these since it memorizes patterns over longer stretches of input. In Section 5, we analyze both

training \ test	BURNC	BDC	DIRNDL
pitch accents			
BURNC	–	73.5	83.2
BDC	85.8	–	83.5
DIRNDL	82.5	66.9	–
phrase boundaries			
BURNC	–	51.2	72.8
BDC	66.2	–	76.5
DIRNDL	58.8	42.6	–

Table 3.38: The cross-corpus detection results (F1-score) for the CNN. The models were trained for 20 epochs.

training \ test	BURNC	BDC	DIRNDL
pitch accents			
BURNC	–	75.2	84.2
BDC	84.5	–	83.2
DIRNDL	83.4	68.8	–
phrase boundaries			
BURNC	–	58.2	79.5
BDC	62.5	–	79.7
DIRNDL	64.8	55.2	–

Table 3.39: The cross-corpus detection results (F1-score) for the LSTM. The models were trained for 10 epochs.

models further and show that even the LSTM benefits from the position indicators. We discuss their role in marking word duration as the main reason for this.

In speech and language processing applications, it is common to model RNNs on sequences of words. This type of method was previously used by Rosenberg et al. (2015) for prosodic modeling. Instead of a frame-based representation of the words, however, they used word-level aggregations as input features. In our case, this could be seen as a combination of the approach taken in the previous section for the feed-forward neural network with aggregated features, and a sequential model. Although both the LSTM and the CNN are comparable in strengh, they may differ in terms of what representations they learn. As both network types learn representations from the same frame-based input, this is an interesting question that we investigate in Section 5.

While we believe that detecting prosodic events using convolutional neural networks is a method that has several advantages (little preprocessing, faster training than the LSTM), there are further questions that exceed the scope of this work: For example, the notion of combining a convolutional model (which learns features from frame-based input) with a sequential model (which processes longer word sequences) could be investigated, however, more sophisticated architectures require much more prosodically annotated training data, which is quite scarce. Since the CNN is efficient and fast to train, we conclude that it is a practical choice for automatic prosodic event detection, especially for use in a pipeline setup.

3.10 Conclusion

This section introduced an efficient method of prosodic event detection based on a convolutional neural network. The only preprocessing that this method requires is the extraction of frame-based acoustic descriptors from the speech signal, and word-level time alignment of the speech data. A simple position indicator feature marks each frame as either belonging to the current word or the two neighboring words in the three-word input matrix.

The model performs comparably to methods proposed in related work on radio news corpora in English (BURNC) and German (DIRNDL) as well as spontaneous speech (BDC) and non-native English (LeaP). The cross-speaker and cross-corpus generalization results are acceptable and comparable to other methods as well. We found that using a simple set of acoustic features extracted using a signal processing toolkit is sufficient to reach good performance levels.

Our experiments extending the CNN to include word-level information, such as word embeddings or word duration, does not yield notable benefits: The former causes the model to overfit to the lexical content of the training data, and the latter does not appear to add any additional information, indicating that the CNN learns this on its own.

In a final experiment, we compared the CNN to a long-short-term-memory network (LSTM) as a sequential model and obtained similar performance levels from both models.

The aim of developing this method was to obtain a simply method of detecting prosodic events that can be readily integrated in a speech understanding pipeline. We believe that this goal has been met, since this method is efficient to train, simple to implement, and yields good detection performance using frame-based acoustic features only. This method limits the amount of preprocessing steps required to apply this method to

new datasets and avoids the use of lexical features, which are very corpus-specific.

The experiments so far looked at different features in terms of how they affect the model's performance on pitch accent and phrase boundary detection. We address the question of what the neural network is learning in Section 5, where we continue to focus on the acoustic, duration and context information investigated in this section. In the next section, we discuss the use of automatic prosodic event detection methods for different downstream applications.

4 Application Examples

The focus of this thesis is not only to provide an efficient method of prosodic event detection, but to focus on the use of prosody for speech understanding pipelines in general. We reviewed several applications of automatic prosodic event detection in Section 2.2. In this section, we describe three additional scenarios in which pitch accent and phrase boundary labels can provide helpful information to downstream tasks.

The first application, which we discuss in Section 4.1, is **slot filling**. Motivated by the fact that the performance of even state-of-the-art methods drops on ASR output, we investigate the use of adding pitch accent information on an English slot filling benchmark dataset. The pitch accent detector used in these experiments is a method developed by Schweitzer (2010).

The second experiment in Section 4.2 was based on a pilot study by Rösiger and Riester (2015), which showed how prosodic information can help **coreference resolution** in German. In this study, the CNN-based prosodic event detector (introduced in Section 3) was used to provide automatically obtained prosodic labels. We analysed the effects of these labels in comparison to manually annotated labels on coreference resolution on the DIRNDL corpus.

In Section 4.3 we discuss an entirely different application of prosodic event detection, namely **corpus annotation**. We provided pitch accent and phrase boundary labels for a German interview corpus using the CNN-based prosodic event detector. Furthermore, we measured the agreement between our labels and those in another prosodic annotation layer that had been created using the method by Schweitzer (2010). We also discuss the utility of the CNN-based method for this type of application.

4.1 Pitch accent features for slot filling

Slot filling is a subtask of spoken language understanding (SLU) that aims at assigning a semantic label to words in a sentence that "fill" a semantic frame, or slot. This task was originally defined for research on spoken dialogue systems or information systems.

In this context, "slots" refer to words or expressions that provide key information such as location, date or time. The Airline Travel Information Systems (ATIS) corpus (Hemphill et al., 1990) was collected as a benchmark dataset for this task.

State-of-the-art methods yield around 95% F1-score on ATIS using deep neural network (DNN) architectures (Yao et al., 2013; Xu and Sarikaya, 2013; Yao et al., 2014; Mesnil et al., 2015; Vu et al., 2016b; Vu, 2016). The features used in these models are typically lexico-semantic representations in the form of word embeddings (Bengio et al., 2000). As slot filling operates on text only, these experiments use the text data provided in the benchmark dataset along with the slot annotations. Since SLU is designed to extract information from speech and thus involves the use of automatic speech recognition (ASR), the more realistic setting would be to apply and optimize these tasks on actual ASR output. Mesnil et al. (2015) reported that the performance of their RNN-based slot filling model drops to around 85% F1-score on recognized text, although their result on the reference dataset is around 95%. He and Young (2003) compared different SLU tasks on recognized and reference text and found that the performance of slot filling (referred to as semantic parsing) drops from around 90% to 89% F1-score (using a vector state model). For this reason, it may be helpful to include additional features that can be extracted from speech, such as prosodic information.

Prosody can be used as an additional knowledge source for slot filling since pitch accents can mark important parts of utterances. Intuitively, this may be explained as follows: During human-to-human discourse, there can be recognition errors due to, for example, environmental noise. A human listener may be able to locate important keywords not only by using the correctly understood context information, but also by prosodic cues. Slots refer to the key query terms, which can be considered as the most important information in this setting. For this reason, we expect the speakers to put special prosodic emphasis on these words in the form of pitch accents. For example, an utterance in the ATIS dataset like *List flights from Dallas to Houston* is expected to have pitch accents on *Dallas* and *Houston*, since these will constitute **new** as well as key, or even **focused**, information in this setting.

While the use of prosody for parsing has previously explored on the ATIS dataset (Veilleux and Ostendorf, 1993), there is no previous work on the use of prosody for slot filling. This section consists of experiments that were published as two conference papers (Stehwien and Vu, 2016, 2017b). In the first paper, we investigated the correlation between pitch accents and words that are annotated with slots in the ATIS corpus. Our goal was to determine whether pitch accents can help in localizing this type of

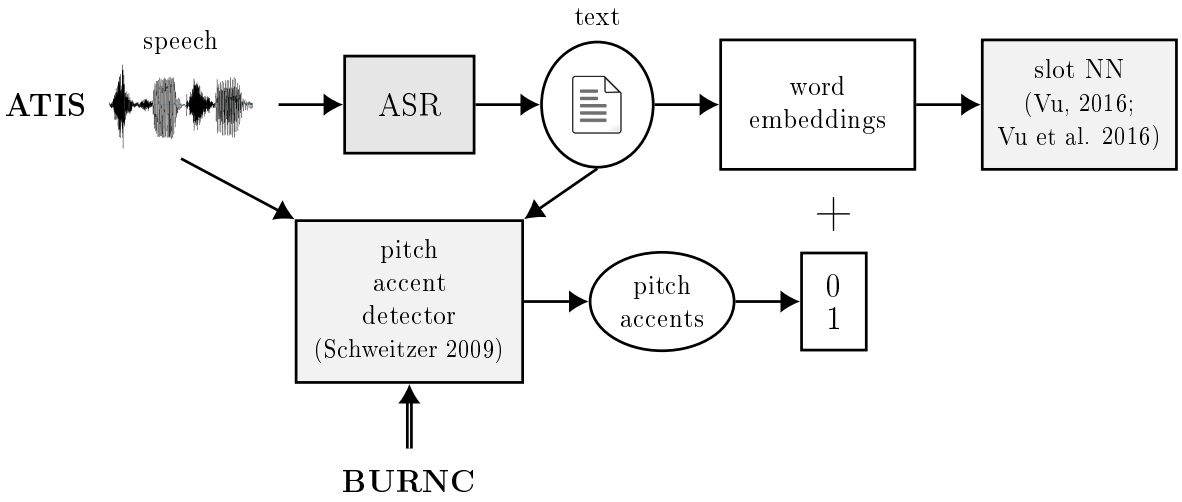


Figure 4.1: Overview of the experimental setup for slot filling. A pitch accent detector trained on BURNC is used to predict binary pitch accent labels that are added to the word embeddings as input features for a neural-network-based slot filling system.

information from raw speech data. Furthermore, we were interested in whether our findings hold on ASR output. We simulated a pipeline setup where we applied an automatic pitch accent detector to an automatically recognized version of ATIS. Since these experiments had been published before the CNN-based prosodic event detector had been developed for this thesis, the pitch accent detection method was based on a method developed by Schweitzer (2010). After we had found a high rate of co-occurrence between pitch accents an slots, our second publication presented a first simple, efficient step to extend two state-of-the-art DNN models for slot filling by adding pitch accent features. This information is included as binary pitch accent features alongside the traditionally used lexico-semantic word embeddings. Previous research has shown that convolutional neural network (CNN) models based on high-dimensional word embeddings can benefit from even a few linguistically informed features (Ebert et al., 2015). We compared the effect this extension has on a bidirectional recurrent neural network (RNN) (Vu et al., 2016b) and a bidirectional sequential CNN (Vu, 2016) applied to recognized text. An overview of our experimental setup in this section is shown in Figure 4.1.

4.1.1 The ATIS corpus

The Airline Travel Information Systems (ATIS) corpus (Hemphill et al., 1990) is a benchmark dataset for spoken language understanding tasks such as intent determination and slot filling. It contains both speech data and corresponding text files that are annotated with semantic labels. The speech files consist of single utterances by speakers requesting flight information from a dialogue system. The corpus was collected in a Wizard-of-Oz-setup[1] from the utterances of several speakers. A human translated the spoken queries into database queries (SQL) and sent the extracted flight information back to the users.

Key query terms are assigned slot labels referring to semantic roles such as *departure date, departure time, airline name.* The remaining words are assigned an empty slot ("O"). The following example sentence was taken from the ATIS corpus:

(1) SHOW *O* FLIGHTS *O* FROM *O* BURBANK *B-fromloc.city_name* TO *O* MILWAUKEE *B-toloc.city_name* FOR *O* TODAY *B-depart_date.today_relative*

The standard benchmark split, used by He and Young (2003) and in subsequent related work, consists of 4,978 utterances from the Class A training data of the ATIS-2 and ATIS-3 datasets. 893 utterances from ATIS-3 are used as test data. The experiments in this section do not use the same training dataset, although the slot filling experiments do use the same test data for comparability. ATIS contains several transcribed versions of the spoken utterances, namely exact transcriptions including any disfluencies uttered by the speaker, and two different levels of "cleaned" versions, namely a read version of the spontaneous utterance and a checked version. Since the slot filling task was not designed for use on speech data, the subset for the benchmark dataset consists of the cleaned and checked data. Therefore, the spoken utterances that correspond to each sentence do not match the slot filling text word for word. Any mismatch in word boundaries in the checked transcriptions and the corresponding audio may also affect the pitch accent detector, but we consider this effect to be small. This does, however, pose a problem for evaluation slot filling on ASR output, and we discuss a workaround approach in Section 4.1.6.

4.1.2 Pitch accent detection based on PaIntE features

The pitch accent detection method used for these experiments was based on a prosodic event detector developed by Schweitzer (2010). This method provides prosodic annota-

[1] The participants were not informed that the fully automatic setup was simulated.

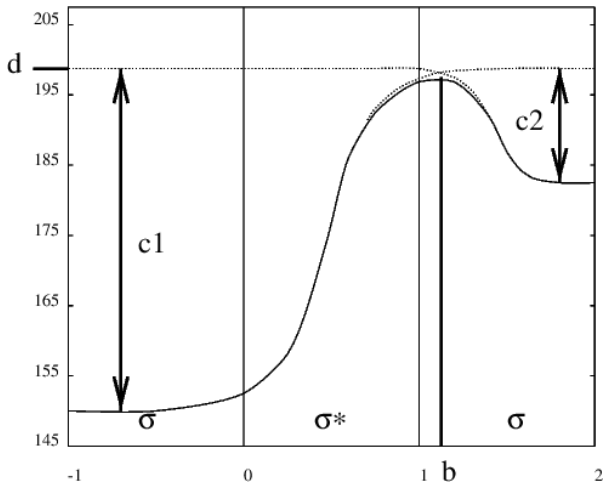


Figure 4.2: Parametrized Intonation Events, taken from Möhler and Conkie (Möhler and Conkie, 1998). The parameters a1 and a2 describe the steepness of the rising and falling sigmoid representing the F0 contour, d and b represent the height and temporal alignment of the peak, and c1 and c2 describe the amplitude of the rising and falling sigmoid.

tions at the syllable-level, representing each candidate syllable with a range of linguistically motivated features. One of the main contributions of this method was the use of PaIntE parameters, which are described in the following.

The Parametrization of Intonation Events (PaIntE) was developed by Möhler and Conkie (1998) and defines six parameters to describe the F0 contour on pitch-accented syllables. This parametrization makes it possible to describe an arbitrary number of curves; and this was the motivation for its initial use in for modeling prosody in text-to-speech synthesis. An illustration of PaIntE is given in Figure 4.2. The contour is fitted by two sigmoid functions that span a three-syllable window around the accented syllable (denoted by $\sigma*$). The six parameters describe the steepness of the rising (a1) and falling (a2) sigmoid, the height (d) and temporal alignment (b) of the peak, and the amplitude of the rising (c1) and falling (c2) sigmoid.

Prosodic event detection method Schweitzer (2010) used these parameters along with phone duration, phonological features and higher-level linguistic information to train a classifier to detect and label GToBI(S) pitch accent and boundary tone types at the syllable level. The PaIntE-parameters used in the feature set are the 6 parameters described above, which are extracted for the current syllable. Additional parameters are the maximum of c1 and c2, the size difference between the rise and the fall, and

the parameters c1, c2 and d of the two preceding and the two following syllables. All parameters were z-scored per speaker, except for the b parameter, since is not considered to be speaker-specific. Further features include syllable, phone and nucleus duration features for the current and surrounding syllables, the vowel identity in these syllables as well as higher-level linguistic features, such as part-of-speech tags and word identity of the current word and the surrounding context. The phonological structure of the data was extracted using an aligner (Rapp, 1995) and the Festival speech synthesis system (Black et al., 2010). The classifier is a random forest model trained using the WEKA machine learning toolkit (Hall et al., 2009).

This model was originally developed for German data (and used language-specific lexical features such as part-of-speech tags) and requires a substantial amount of pre-processing that was not readily applied to the datasets used in this work. For our purposes, we reduced the feature set to those that were most readily extracted from new, automatically recognized or multi-lingual data in order to simuluate a pipeline setting. Thus, we adapted the method to use purely acoustic features: 23 features were extracted at the syllable-level, including z-scores of the six PaIntE parameters and further amplitude descriptions (maximum of c1 and c2, difference between c1 and c2), as well as the duration of the syllable nucleus. The rest of the features comprise c1, c2, d and the nuclei durations for the two preceding and following syllables. This model predicts pitch accents at the syllable level, but we evaluated at the word level by considering only syllables with primary stress. After adapting the above method to better suit our purposes, we trained the pitch accent detector on English data. We report on the performance in the next section.

4.1.3 Performance of the pre-trained model

Since the ATIS corpus does not contain prosodic labels, we used a pre-trained version of the pitch accent detector to label the dataset. The prosodically annotated corpus we chose for pre-training the model was a subset of BURNC. For evaluation, we first tested the performance of the PaIntE-based pitch accent detector on BURNC, and then created predictions for ATIS.

Speaker-independent performance on BURNC We used a smaller subset of BURNC than in Section 3 since several files had to be discarded during the extensive feature extraction pipeline. The resulting subset consisted of 220 speech recordings by 5 speakers (3 female and 2 male), amounting to around 1 hour and 20 minutes of speech. Dataset

speaker	f1a	f2b	f3a	m1b	m2b
# files	42	100	16	30	32
# words	1,865	6,994	1,142	1,907	1,928
# accents	1,126	3,978	609	1,022	1,153
% accents	0.60	0.56	0.53	0.54	0.60

Table 4.1: Dataset statistics of the BURNC subset used to train the PaIntE-based pitch accent detector.

speaker	accuracy (%)
f1a	73.1
f2b	74.7
f3a	76.7
m1a	73.7
m2b	73.8

Table 4.2: Pitch accent detection accuracy per speaker on the BURNC subset. The evaluation is at the syllable level but considers stressed syllables only.

statistics per speaker are shown in Table 4.1. Table 4.2 shows the per-speaker accuracy of the model. The prediction and evaluation was carried out at the syllable level by the predictor, but took only stressed syllables into account, since we were interested in how the model performs at the word level. Ranging from around 73% to 77% accuracy, the model was found to be quite stable across the various speakers. With 5-fold leave-one-speaker-out cross-validation, we obtained 74.4% accuracy on average. Considering the fact that we built the model solely from F0 (i.e. PaIntE) and nuclei duration features without any further acoustic or lexical information, we judged the performance to be sufficient for our purposes. Table 4.3 shows the per-speaker confusion matrices for these results. This overview shows how many words were labeled either correctly or falsely as *ACCENT* and *NONE*. For all speakers, the *ACCENT* class is more frequently labeled as a false positive than the *NONE* class as a false negative. While this means that the model slightly overpredicts the *ACCENT* class, we do not consider this to be a disadvantage in this case, since pitch accents are used as additional cues for lot locations and not as a sole predictor of a semantic label.

Cross-corpus performance on ATIS After having evaluated the within-corpus performance, we trained the pitch accent detector on the BURNC subset and applied it to the ATIS corpus. Since the ATIS corpus does not contain prosodic labels, we estimated

Speaker	# Ref. \ # Pred.	ACCENT	NONE
f1a	ACCENT	804	199
	NONE	201	282
f2b	ACCENT	3348	400
	NONE	1075	967
f3a	ACCENT	399	49
	NONE	126	178
m1b	ACCENT	506	104
	NONE	159	230
m2b	ACCENT	633	124
	NONE	182	228

Table 4.3: Confusion matrices per speaker for syllable-level pitch accent detection.

# files	50
# words	514
# slots	201
# human-labeled accents	235
# words with predicted accents	234
agreement: # words	173
precision	58.6
recall	76.6
F1-score	66.4

Table 4.4: Performance of the pitch accent detector, evaluated using 50 manually labeled files of the ATIS subset.

the accuracy of the pre-trained pitch accent detector using a small amount of hand-labeled data. For this, we annotated 50 files with pitch accents. The annotation as well as the evaluation was done at the word level. Table 4.4 shows that in total around 230 words were found to carry pitch accents by both the automatic predictor and the human labeler. Specifically, they agree in 173 cases, which amounts to an accuracy of around 70%. This is similar to the within-corpus performance on BURNC, and shows that this detection method is reasonably accurate on this dataset. Although the performance is lower than state-of-the-art methods, we considered it to be sufficient for our experiments on the ATIS corpus. We argue that in order to keep the simulated pipeline experiments "realistic", a certain amount of labeling error should be taken into account.

Performance on automatically recognized speech The previous performance levels were obtained using the manual transcriptions of the speech data. Our pipeline experiment aimed to simulate a setup where we have no transcriptions available, and intend to detect pitch accents using the audio file only. Automatically recognizing the spoken utterances using ASR, however, introduces recognition errors and thus results in slightly different word boundaries. As our pitch accent detector relies on word boundary information, we also evaluated the pitch accent detector on an automatically recognized version of BURNC.

The detection of prosodic events using this method requires the speech files to be time-aligned with the respective phone-, syllable- and word-level transcriptions. An ASR system can provide this information along with automatic transcriptions. The acoustic model for our ASR system was trained on the training set of the Wall Street Journal (WSJ) corpus (Paul and Baker, 1992) using a neural network model setup (*nnet2*) in the Kaldi ASR toolkit (Povey et al., 2011)[2]. As a language model, we interpolated a WSJ Kneser-Ney 4-gram model with a bigram model trained on the aforementioned BURNC subset (we treated both as having equal weight by setting $\lambda = 0.5$) with which we obtained a perplexity[3] of 205 on the BURNC subset. The recognized data contained a word error rate (WER) of 27.4%[4]. Finally, we created phone, syllable and word alignments for the recognized transcription.

We predicted the locations of pitch accents using (1) the recognized output and (2) the reference transcriptions for each speaker individually, using the respective pretrained model trained data from all other speakers. Thus, these models were speaker-independent. Since we were interested in pitch accents that lie on certain words, and the pitch accent detector detects and evaluates at the syllable level, we compared the results at the word level with the reference accent labels[5]. Therefore, the results for the reference transcriptions are not completely identical to those shown in Table 4.2.

The results for the first scenario in which we used the reference transcription are shown in Table 4.5. We found that the F1-scores obtained on the male speakers (m1b, m2b) are considerably lower (62.5%) than those obtained on the female speakers (>70%). Table 4.6 contains the results for the same procedure using the recognized transcriptions. Here,

[2] www.kaldi-asr.org

[3] Perplexity refers to the branching factor of a language model

[4] Word error rate is defined the number of substitutions, insertions and deletions divided by the length of the sequence.

[5] Since the pitch accent detector makes a prediction for each stressed syllable, we found that in some cases, the predictor placed several accents on one word (e.g. words with several stressed syllables like *transportation*). This evaluation, however, only counts one accent per word.

we observed a much lower variance across speakers. While this is an interesting result, we do not have an immediate explanation for this. The averages of these measures over all speakers are listed in Table 4.7. The precision was higher when using the reference transcriptions, but the recall was lower. Considering that the average F1-score lies around 75% for both versions, we concluded that manual transcriptions are not necessary for localizing pitch accents in speech. This finding suggests that pitch accent detection can be readily integrated into a pipeline setup, and does not suffer considerably from ASR errors.

speaker	f1a	f2b	f3a	m1b	m2b
# predicted	1,005	4,459	525	665	815
# true positives	817	3,348	406	527	655
precision	81.3	75.1	77.3	79.2	79.2
recall	72.6	84.2	66.7	51.6	51.6
F1-Score	76.7	79.4	71.6	62.5	62.5

Table 4.5: Word-level accuracy of pitch accent detection per speaker on the BURNC subset using reference transcriptions.

speaker	f1a	f2b	f3a	m1b	m2b
# predicted	1,146	4,527	690	1,027	1,130
# true positives	855	3,195	468	749	867
precision	74.6	70.6	67.8	72.9	76.7
recall	75.9	80.3	76.8	73.3	75.2
F1-Score	75.3	75.1	72.1	73.1	76.0

Table 4.6: Word-level accuracy of pitch accent detection per speaker on the BURNC subset using recognized transcriptions.

	reference	automatic
# predicted	7,460	8,520
# true positives	5,753	6,134
precision (%)	77.0	72.0
recall (%)	72.9	77.8
F1-Score (%)	74.9	74.8

Table 4.7: Average word-level accuracy of pitch accent detection using automatic and reference transcriptions of the BURNC subset (220 files, 13836 words, 7888 accents).

4.1.4 Correlation of pitch accents with semantic slots in ATIS

Experimental setup Having tested the performance of the pre-trained pitch accent detector on ATIS and on an automatically recognized version of the BURNC subset, we applied it to an automatically recognized version of the ATIS corpus and analyzed the correlation of semantic slots with the predicted accents. For this study, we used a subset of the ATIS corpus (607 files of the ATIS-3 test set) that is annotated with semantic slots; and the rest of the ATIS dataset was used to train an ASR model to recognize the smaller subset. We used a Kaldi ASR model[6] trained on the ATIS-2 and ATIS-3 Class-A (context-independent) training data to recognize the test set (WER 11.7%) and time-align it to the audio files. We used the pitch accent detector trained on the entire BURNC subset to predict the locations of pitch accents on the smaller ATIS subset.

In order to measure the correlation of the predicted pitch accents with slots, we counted how many pitch accents were co-located on words annotated with semantic slots. We were interested in how many slots co-occur with predicted pitch accents and in what cases pitch accents occur on non-slot words. Table 4.8 provides an overview of the results.

# files	607
# words	6,099
# slots	2,452
# predicted accents	3,428
# pred. accents on slots	2,218
# pred. accents on non-slots	1,210
slots with pred. accent	90.5 %

Table 4.8: Frequency of predicted pitch accents in a subset of the ATIS3 test set

Results The pitch accent detector predicted around 3,400 accented words of which 2,218 were annotated with slots. This result shows that we can cover 90.5% of all the slots in the dataset using the pitch accent detector, which implies that most slots can be localized by finding pitch accents. Since we were also interested in what other information we can localize in this manner, we also considered the 1,210 pitch accented words that are not associated with slots. Table 4.9 contains a list of the non-slot words that most frequently received a predicted pitch accent. Some examples of words that do not have semantic labels but are frequently accented are *list, what, which, please, show*

[6] We used a triphone model with features transformed by latent discrimination analysis (LDA).

and *need*. These are question words and imperatives that show the speaker's intention, namely to request the program to provide information. Frequent non-slot words that are not often pitch accented are function words like *to, from, and, the*, which is typical of intonation patterns in English. The word *flight* and its plural form *flights* occur quite often in ATIS and are accented around 60% of the time. These are important content words, however since the semantic content of this corpus revolves around air travel information, it may often be regarded by the speakers as *given* information, which is usually deaccented. In sum, these results indicate that the localization of pitch accents can directly be used to find important information from speech data.

word	accented	frequency	accent ratio
FLIGHTS	179	313	57.2
LIST	137	157	87.3
FLIGHT	93	136	68.4
TO	87	504	17.3
ON	83	160	51.9
WHAT	79	87	90.8
I	78	106	73.6
FROM	78	448	17.4
ME	76	102	74.5
SHOW	63	83	75.9
NEED	39	50	78.0
ALL	37	48	77.1
WHICH	32	34	94.1
AND	31	73	42.5
PLEASE	31	33	93.9
THE	29	184	15.8

Table 4.9: Most frequent accented non-slot words in the ATIS subset and their accent ratio.

Coverage of slots using pitch accents in agreement with human labeling As an additional evaluation, we compared the correlation between slots and both the human-annotated and the automatically predicted pitch accents. Tables 4.10 and 4.11 list the results. Around 74% of slot words were judged by the human labeler to bear a pitch accent, while around 82% were predicted as accented by the automatic detector. These numbers show that in this specific subset, the predicted pitch accents correlate more with slots than those created by the human labeler. It appears that in this specific case, the detector can be considered more suitable for localizing slots.

# human-labeled accents	235
# accents on slots	149
# accents on non-slots	86
# slots with no accent	52
# slots with accent	74.1 %
accents on slots	64.7 %

Table 4.10: Correlation between slots and human-annotated pitch accents in 50 ATIS files.

# predicted accents	234
# accents on slots	164
# accents on non-slots	70
# slots with no accent	37
# alots with accent	81.6 %
accents on slots	53.4 %

Table 4.11: Correlation between slots and automatically predicted accents in 50 ATIS files.

Summary This study aimed to find evidence that prosody, specifically the presence of pitch accents, can provide useful information for SLU tasks. In a first experiment, we found that a PaIntE-based pitch accent detector trained on part of BURNC does not require pre-transcribed data at test time: the transcriptions can be obtained from ASR and still yield comparable results, which means we can readily include a pre-trained pitch accent detector in a pipeline system. The second part of this study examined the correlation of accents and semantic slots in the ATIS corpus. We showed that most of the words in the corpus that are labeled with slots also carry a pitch accent. This finding agrees with the expectation that words that are important and pertain to new information should be perceptually more prominent. Furthermore, we determined that many words that are pitch accented but are not associated with slots also convey substantial information about the speaker's objective, and can thus point towards the speaker's intention or the general topic. We conclude that the high correlation between pitch accents and semantic slots found in this study as well as the comparability of our results on ASR output motivates the use of prosody for slot filling.

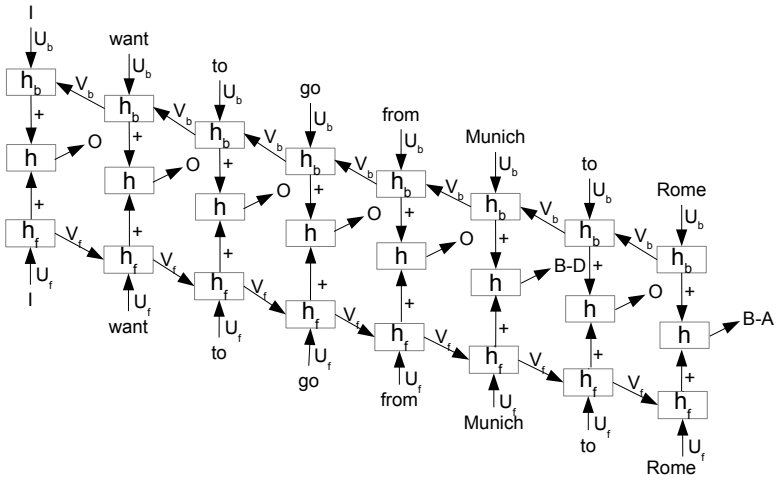


Figure 4.3: Bidirectional RNN for slot filling, taken from Vu et al. (2016b).

4.1.5 Pitch accent features for neural slot filling

Having shown that there is a notable correlation between pitch accents and semantic slots in the ATIS corpus, we integrated this information into neural slot filling models that are based on textual information in the form of word embeddings. Our aim was to investigate whether we can exploit the co-occurrence between pitch accents and slots to improve slot filling on automatically recognized text.

Neural slot filling models Two neural baseline slot filling systems were built based on a recurrent neural network (RNN) and a convolutional neural network (CNN) proposed by Vu et al. (2016b) and Vu (2016), respectively. We examined the effect that adding pitch accent extensions to the word embeddings have on both models. These two models yield similar performance levels, but differ in the way they process the information provided by the word embeddings. The first model (Vu et al., 2016b) is an Elman-type bidirectional RNN that yields an F1-score of 95.56% on the benchmark ATIS dataset. The features in this model are 100-dimensional word embeddings that are randomly initialized and jointly trained along with the RNN. The bidirectionality of this model refers to the combination of a forward and backward hidden layer using an addition operator in order to take past and future contexts (trigrams) into account. Figure 4.3 illustrates the architecture of this model.

The second model involves a bidirectional sequential CNN (Vu, 2016), as shown in Figure 4.4. It slightly outperforms the first model on the slot filling task with an F1-score of 95.61% on the ATIS benchmark. This method combines two CNNs that model

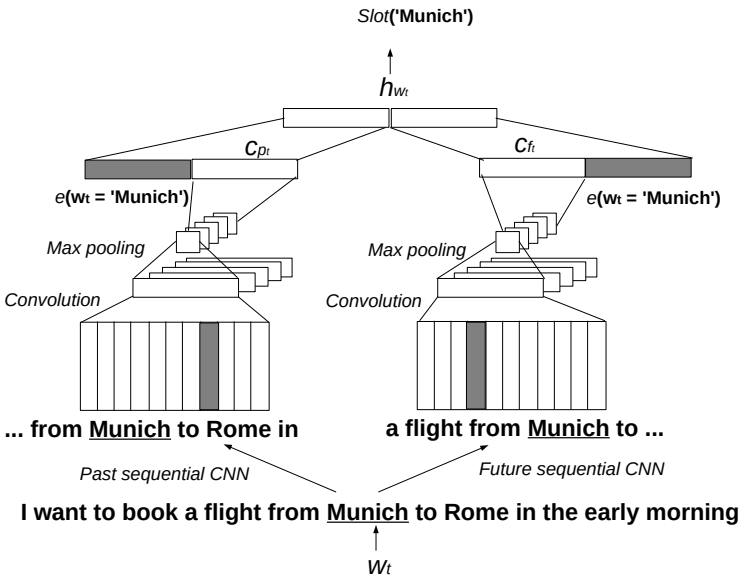


Figure 4.4: Bidirectional sequential CNN for slot filling, taken from Vu (2016).

the future and past contexts respectively, as well as an additional extended surrounding context. The output representation of the past and future contexts are concatenated to form the final feature representation. The context windows for both the forward-moving and backward-moving CNNs comprise 9 words in total, and the surrounding context consists of 3 words. The word embeddings in this model are 50-dimensional.

Objective Function Following Vu et al. (2016b) and Vu (2016), we use the ranking loss function proposed by Santos et al. (2015). Instead of using the softmax activation function, we train a matrix W^{class} whose columns contain vector representations of the different classes. Therefore, the score for each class c can be computed by using the product:

$$s_\theta(w_t)_c = h_{w_t}^T [W^{class}]_c \tag{4.1}$$

The ranking loss function Santos et al. (2015) maximizes the distance between the true label y^+ and the best competitive label c^- given a data point x. The objective function is

$$L = \log(1 + \exp(\gamma(m^+ - s_\theta(w_t)_{y^+}))) + \log(1 + \exp(\gamma(m^- + s_\theta(w_t)_{c^-}))) \tag{4.2}$$

with $s_\theta(w_t)_{y^+}$ and $s_\theta(w_t)_{c^-}$ as the scores for the classes y^+ and c^- respectively. The parameter γ controls the penalization of the prediction errors, and m^+ and m^- are

margins for the correct and incorrect classes. γ, m^+ and m^- are hyperparameters which can be tuned on the development set. For the empty class O (no slot), only the second summand of Equation 4.2 is calculated during training, that is, the model does not learn a pattern for class O but nevertheless increases its difference to the best competitive label. Furthermore, it implicitly solves the problem of unbalanced data, since the number of class O data points is much larger than other classes. During testing, the model will predict class O if the scores for all other classes are < 0.

Word embeddings with pitch accent features Instead of modifying the system architecture, we modified the purely text-based embedding vectors to include information on the presence or absence of a pitch accent on the respective word. This procedure constitutes a simple and efficient way to enhance word embeddings with pitch accent information for NLP tasks. The modified vector for a word w consists of the word embedding of w as well as a binary flag $\in (0, 1)$ indicating the absence or presence of a pitch accent on w:

$$embs(w) = [lexical\_embs(w), pitch\_accent\_flag(w)] \tag{4.3}$$

"1" refers to pitch-accented words and "0" to non-pitch-accented words. To extract this flag, we first recognized the data with an ASR system on the audio signal to obtain the word, syllable, and phone alignments. Afterwards, the pre-trained PaIntE-based pitch accent detector was applied to determine the binary label for each of the recognized words. The lexical word embeddings were randomly initialized and concatenated with the binary pitch accent flag to form the new vectors.

4.1.6 ASR version of the ATIS benchmark dataset

In these experiments, we used 4,924 utterances[7] and the corresponding .WAV files from the training dataset and used the 893 test utterances. We recognized the ATIS test set using a triphone model from the Kaldi toolkit trained on around 4,900 utterances from the ATIS training data with 7.06% WER.

[7] The reason for our smaller training set is the fact that the utterance IDs for the benchmark training dataset were not available to us and some sentences did not have a matching equivalent in the original corpus. This is necessary, however, to create pairs of audio files and the respective transcriptions for our experiments.

Transfer of slot annotations to ASR output Having obtained recognized text, we faced a problem arising from the fact that that the slot annotations were created for the original text, and we could not simply transfer the annotations to the ASR output for evaluation since the text differed slightly. This is because the text version that was annotated with slots was a cleaned version of the utterance transcriptions and therefore, the ASR output did not match the text directly. Furthermore, the ASR output also contained some recognition errors that resulted in mismatched text. In order to make the recognized output compatible with the slot filling system and comparable with the original test data, the following preprocessing steps were necessary. We time-aligned the words in the original text to the audio files using Kaldi, which automatically provided the time intervals for each slot. Then, using the time-alignments of the recognized text, we assigned each word in the recognized output a slot label according to the time intervals of the respective label and the reference word itself: if the reference and recognized words were the same, we assigned the recognized word the label of the reference word, on the condition that their time intervals matched within a threshold of 0.05 seconds. Slot-annotated words in the reference that did not have an exact match in the recognized output were assigned to the recognized word within a time interval of 0.02 seconds. Using this method, we automatically acquired the annotations for the recognized output that we needed in order to measure slot filling performance. Human-detectable inconsistencies could not be ruled out, however, because of recognition errors and falsely aligned words, even though these remained at an acceptable level. It should also be noted that in the "cleaned" transcriptions for the slot filling benchmark, several words from the original transcriptions had been replaced in order to better suit the semantic labeling task, e.g. dates and times like *TWO THIRTY P M* were replaced by *DIGITDIGITDIGIT PM* and the commonly occurring token *SAINT* in city names was replaced by *ST.*. Also, clitics like *I'D* were tokenized as *I 'D*. This issue had to be taken into account because the ASR system usually recognizes the full words, which thus needed to be converted afterwards. The ATIS slot filling training and test data as used by Mesnil et al. (2013)[8] already contains the IDs for each word for use as feature vectors. We reused the same word-to-ID and label-to-ID indexes to create the new recognized test set.

Correlation of pitch accents and slots on the ATIS test set Next, we ran the same analysis as in the previous study to estimate the correlation between pitch accents and

[8]The data was downloaded from `http://deeplearning.net/tutorial/rnnslu.html` as a Python pickle object.

	original transcriptions	recognized text
# files	893	893
# words	9551	9629
# slots	3663	3560
# predicted accents	5295	5169
# pred. accents on slots	3395	3308
# pred. accents on non-slots	1900	1861
slots with pred. accent	92.7%	92.9%

Table 4.12: Co-occurrences of pitch accents and slots in the original and recognized transcriptions of the ATIS test set.

slots, both in the benchmark test set of ATIS and the recognized version. We counted the co-occurrences in both the recognized and reference transcriptions. Table 4.12 shows the results of this analysis. Almost 93% of the words that are annotated with slot labels are also pitch-accented. This holds for both the original transcriptions as well as the ASR output. Thus, the co-occurrence rate of slots and pitch accents in the ATIS benchmark test set is similar to that in the ATIS subset found in the previous analysis.

4.1.7 Slot filling experiments

After obtaining a recognized version of the slot filling benchmark test set and confirming that the correlation between pitch accents and slots is as high as in the previously tested dataset, we used this data in our slot filling experiments. We compared the effect of adding pitch accent information to a text-based slot filling system on both reference and recognized text.

Experimental setup We time-aligned the full ATIS dataset at the phone, syllable and word level. We applied the pitch accent detector to obtain the time points of every predicted pitch accent in the training data, the original transcriptions of the test data and the recognized test data. This new information was included in the input data to the slot filling models as a binary feature vector for each word: if there is an accent within the time interval of a word, then the feature value was set to 1, if not, then the value was set to 0. We added this information to the baseline slot filling models using the method described in Section 4.1.5. The models were trained using stochastic gradient descent with up to 100 epochs and early stopping. We began training with an initial learning rate of 0.025 and halved the learning rate if the performance on the development set did not improve after 5 epochs.

Results We compared the performance of the RNN and CNN slot filling models described in Section 4.1.5 on both the original text and the recognized output. The F1-scores are given in Table 4.13. Our results on the manual transcriptions are slightly different compared to the results reported in Vu et al. (2016b) and Vu (2016), which can be explained by the fact that we used a smaller training set and used 10% of the training data as a development set for early stopping. As expected, results on the ASR output are much worse than on the manual transcriptions. This performance drop is a result of the recognition errors produced by the ASR system. The pitch accent extensions did not improve the F1-score when using manual transcriptions, but did not harm the slot prediction performance either. This indicates that context information as modeled by the sequential model is strong enough to predict the correct slots in this dataset. Adding the pitch accent extensions, however, slightly improved the F1-score of both the RNN and the CNN on ASR output. This implies that the added pitch accent information may help to balance out the effect of ASR errors in some cases. In the next section we discuss several reasons why slot filling performance drops on recognized output, and analyze in what cases the pitch accent extension may help improve the results.

	RNN	CNN
Transcriptions (lexical word embeddings)	94.97	95.25
+ pitch accent extensions	**94.98**	**95.25**
ASR output (lexical word embeddings)	89.55	89.13
+ pitch accent extensions	**90.04**	**89.57**

Table 4.13: F1-scores for slot filling on original and recognized text and after adding pitch accent features.

4.1.8 Analysis of slot filling results on ASR output

An interesting aspect of the ATIS data is the occurrence of *unknown* tokens, which also poses a challenge for slot filling, especially on recognized text. In the following we discuss the role of these tokens and how pitch accent information can help deal with them.

In the benchmark dataset, words that occur only once were replaced with the unknown token <*UNK*>. The unknown token is part of the official dataset and serves as a "wildcard", receiving its own representation via the word embeddings and, at the same time, can be assigned any label. After ASR, additional unknown tokens were added to the transcriptions. Some words were not correctly recognized by the system, that is either incorrectly recognized or not recognized at all, and thus are also marked as (<*UNK*>).

If an incorrectly recognized word in the ASR output did not exist in the original dataset, then it was also replaced by *<UNK>*, in line with the original benchmark.

Taking a closer look at our results, we found cases in which different slot labels were assigned to the unkown token, depending on whether the models were extended with pitch accents. Two example sentences are given in Table 4.14. In the first sentence, the unknown token was assigned a *location* slot label by the model trained with pitch accent features, while the baseline model assigned the empty slot. In the second sentence, the word "Toronto" was incorrectly recognized by the ASR system. The baseline model assigned the lexically appropriate label *round trip*, although it is contextually incorrect. In this case, the model that was extended with pitch accents simply "ignored" this part of the sentence. If no label were to be considered better than a false label, then this outcome would be more appropriate.

We evaluated the RNN results in order to gain some insight into whether the pitch accent information helps to label slots on unknown words, independent of the slot type. Specifically, we counted the instances in which the model correctly distinguished slots from the empty class. Of all unknown words with a reference slot, around 43% were correctly labeled in the baseline model, whereas around 51% were correctly assigned a slot when using pitch accent extensions. This indicates that pitch accent information was used by the model to localize a semantic slot, even though the predicted slot type was incorrect. Thus, unknown words may still originally carry acoustic information in the signal that is beneficial to this task. This information appeared to be captured by the proposed method.

For comparison, we also considered the overall impact of the unknown tokens. In order to estimate how well the model would perform without the weakness stemming from unknown words, we calculated the F1-score after all predicted labels on unknown words were replaced by the correct label. This upper bound for the RNN model on ASR output is 91.48% with pitch accent features and 91.1% without. Using the non-ASR text (unknown tokens were only used when words occur only once in the corpus) we estimated an upper bound of 96.01% if all unknown words were labeled correctly (the actual result was 94.97%). The absolute difference in the amount of unknown words is 69 in the original test data and 285 tokens in the recognized version.

reference text	I NEED THE FLIGHTS FROM **WASHINGTON** TO MONTREAL ON A SATURDAY
recognized text	I NEED THE FLIGHTS FROM <***UNK***> TO MONTREAL ON SATURDAY
reference slots	O O O O O **B-fromloc.city_name** O B-toloc.city_name O B-depart_date.day_name
with accents	O O O O O **B-fromloc.city_name** O B-toloc.city_name O B-depart_date.day_name
baseline	O O O O O **O** O B-toloc.city_name O B-depart_date.day_name
reference text	WHICH AIRLINES FLY BETWEEN **TORONTO** AND SAN DIEGO
recognized text	WHICH AIRLINES FLY BETWEEN **TO ROUND** <***UNK***> AND SAN DIEGO
reference slots	O O O O **O O O** O B-toloc.city_name I-toloc.city_name
with accents	O O O O **O O O** O B-toloc.city_name I-toloc.city_name
baseline	O O O O **B-fromloc.city_name B-round_trip I-round_trip** O B-toloc.city_name I-toloc.city_name

Table 4.14: Example utterances, recognized versions and their slot labels, predicted by the RNN model with and without pitch accent extensions.

# utterances	total	with WER >0	with <*UNK*>
fewer errors	25	14 (58%)	8 (32%)
more errors	153	116 (76%)	91 (59%)

Table 4.15: Number of utterances with fewer or more slot filling errors obtained using the RNN on ASR output (89.55%) compared to the results on original text (94.97%).

# utterances	total	with WER >0	with <*UNK*>
fewer errors	51	36 (70%)	29 (70%)
more errors	37	28 (75%)	26 (70%)

Table 4.16: Number of utterances with fewer or more slot filling errors obtained using the RNN with pitch accents (90.0%) compared to RNN without accents (89.55%).

Next, we investigated how the ASR output quality affects the slot filling results and how the pitch accent features help. We analyzed how WER affects the errors in slot filling by counting utterances in the test set on which the WER was larger than 0, i.e. the resulting text was different than the reference transcription. We also counted how many utterances contain slot filling errors by measuring the Levenshtein distance between the sequence of predicted and reference slot labels to obtain a "slot error rate". These numbers are given in Table 4.15. This table shows how many utterances were labeled with more or fewer slot errors in the ASR output versus the manual transcriptions. We observed that the WER correlates with slot errors around 76% of the time, which means that utterances with a higher WER also contained more slot filling errors in 76% of the cases. Also, almost 60% of the time, the slot filling result was worse when unknown words were present. Table 4.16 shows the same analysis after adding pitch accent features. These results show that the pitch accent features help to increase the number of utterances with fewer slot filling errors, and that around 70% of these errors can be attributed to recognition errors and the presence of unknown words. However, this table also shows that the pitch accents cannot help in every case, since a comparable percentage of utterances with more slot errors are also caused by these factors.

4.1.9 Discussion and conclusion

In this section, we addressed the notion of overcoming the performance drop of state-of-the-art slot filling methods on speech recognition output. Our method involved combin-

ing pitch accent features with word embeddings as a way of including acoustic-prosodic information in the downstream task. We tested this method on the ATIS benchmark corpus using two different neural slot filling models. In terms of quantitative results, small but positive effects were obtained on both models. The analysis showed that pitch accent features may be helpful in the case of incorrectly recognized or unknown words. However, there are some limitations to the conclusions that can be drawn from these experiments alone, which we discuss in the following.

We cannot observe a clear pattern to support the assumption that the presence or the absence of a pitch accent can help in identifying slots using these models. This leads to the question on why the prosodic information used in this work did not prove helpful. We believe that the main reason for this is that the neural slot filling models were developed specifically for the text-based benchmark ATIS dataset. They are also sequential models that rely heavily on the lexical context, which likely overrides any further information added to the system.

Since the ATIS corpus consists of human-to-machine speech recorded in a laboratory setting (Hemphill et al., 1990), its naturalness may be questioned. Tur et al. (2010) discussed limits to the utility of the ATIS corpus for research in spoken language understanding. They argued that, while state-of-the-art methods achieve a high accuracy on this dataset, this does not mean that the tasks are close to being solved. Instead, the limited and closed-domain nature of ATIS, as well as the fact that the utterances are not completely natural, accounts for this. After investigating unseen examples in the test corpus, they concluded that more data should help expand the possibilities of training more powerful models. They analyzed typical slot filling errors and categorized them as being caused by long distance dependencies, incorrectly assigned slot types, unseen word sequences due to the mismatch between training and test sets or simply annotation errors. At least for unseen word sequences that result in unknown words or recognition errors, our analysis has indicated that pitch accent features can be useful additional cues.

It should also be noted that the drop in measured performance may also in part stem from the workaround slot annotation method we used to transfer the reference slots to the ASR output (Section 4.1.6). Incorrectly recognized words may have received a slot label if the time interval was very similar to the original; or correctly recognized words may be labeled as empty if the aligner makes a sufficiently large error. The drawback of using this method is that false negative errors may occur when the model predicts a correct label when the reference label was actually false. We consider this effect to

be comparably small, but it is an important issue to consider when evaluating tasks on recognized output, for which no reference annotations exist.

Another aspect to consider is the performance of the pitch accent detector, and the fact that automatic labels will inherently contain some degree of error. The pitch accent detection accuracy of the model used in this work may simply be too low. The PaIntE-based method may not be suitable for use in pipeline setting, since it was originally designed to receive more higher-level linguistic, as well as lexical, information. Furthermore, it had been created and optimized for German; and we had reduced the feature set to purely acoustic features. In such a setting, the CNN-based prosodic event detector would be a better choice, since it performs well across speakers and languages, especially since no POS-tags are required. Since it is a word-based rather than a syllable-based method, it may also be less sensitive to alignment and recognition errors. Repeating these experiments with the CNN-based method, however, would likely not change the overall trends observed in this section. We expect that a stronger pitch accent detector would lead to small improvements, but only to the extent that would be possible by means of "perfect" prosodic labels. We did not test the effect of adding manually predicted pitch accents, which would provide an upper bound to the effect that these labels have on the slot filling performance.

In future experiments, it would make sense to expand the prosodic information added to the slot filling models. The binary pitch accent flag used in these experiments is quite simple. One possibility would be to adapt the embeddings to encode more prosodic information, similar to the method used by Wang et al. (2016). For example, neural network-based modeling of pitch accents yields different forms of feature (e.g. hidden) representations that could be used instead of a symbolic pitch accent representation. This approach, however, would call for more data and for manual prosodic labeling. Furthermore, the current trend of direct modeling on speech data or creating joint architectures may also be a promising direction.

Overall, there may be simply too many factors involved that override any effect these new features have. Our analysis provides a straightforward intuition of the benefits of prosodic information for this task, but it remains difficult to determine whether the proposed method, or the fact that the ATIS dataset is rather limited, is the reason why the performance increase is only slight. While further investigation is necessary to fully investigate the potential of prosodic information in SLU, this section presented promising first results on ATIS as the benchmark dataset for slot filling. Furthermore, we showed that our method does not require any kind of manual transcriptions during

test time. We conclude that using automatic pitch accent detection in spoken language understanding pipelines is a direction that still deserves attention in future research.

4.2 Prosodic features for coreference resolution

The next section presents another example of a downstream task that can benefit from prosodic information, namely coreference resolution. This is a typical task in NLP that usually does not include information from speech. We first introduce the task of coreference resolution, as described in the conference paper in which the experiments in this section were published (Rösiger et al., 2017) and list the contributions of this thesis.

4.2.1 Motivation

Noun phrase coreference resolution is the task of grouping noun phrases (NPs) together that refer to the same discourse entity in a text or dialogue. In Example (2), taken from Umbach (2002), the question for the coreference resolver, besides linking the anaphoric pronoun *he* back to *John*, is to decide whether *an old cottage* and *the shed* refer to the same entity. The noun phrases are marked with curly brackets and are also referred to as "mentions". They are marked with subscript numbers that refer to each entity (here: *John* and *he*).

(2) {John}$_1$ has {an old cottage}$_2$.
Last year {he}$_1$ reconstructed {the shed}$_?$.

Coreference resolution is inherently a text-based (NLP) task, although a few systems for pronoun resolution in transcripts of spoken text have been proposed (Strube and Müller, 2003; Tetreault and Allen, 2004). Differences between written and spoken text lead to a drop in performance when coreference resolution systems developed for written text are applied on spoken text (Amoia et al., 2012). These differences stem not only from potential recognition errors, as in the previously discussed case of slot filling, but also from the fact that there are differences between written and spoken language (including spontaneous speech). For this reason, it may help to use additional information available from the speech signal, for example prosody.

In English and German, coreferent items (i.e. entities that have already been introduced into the discourse) tend to be deaccented, as the speaker assumes the entity to be salient in the listener's discourse model (Terken and Hirschberg, 1994; Baumann and

Riester, 2013). Thus, their information status is **given**. The experiment in this section exploits this connection by providing prosodic information to the coreference resolver. Example (3), this time marked with prominence information, shows that prominence can help resolve cases in which the transcription is potentially ambiguous. The anaphor under consideration is typed in boldface; its antecedent is underlined. Accented words are capitalized.

(3) {John}$_1$ has {an old cottage}$_2$.

a. Last year {he}$_1$ reconstructed {the SHED}$_3$.

b. Last year {he}$_1$ RECONSTRUCTED **the shed**}$_2$.

The pitch accent on *shed* in (3-a) leads to the interpretation that *the shed* and *the cottage* refer to different entities, and the shed is a part of the cottage (i.e. they are in a bridging relation). In contrast, in (3-b), *the shed* is deaccented, which suggests that *the shed* and *the cottage* co-refer.

A pilot study by Rösiger and Riester (2015) had shown that enhancing the text-based feature set for a coreference resolver, consisting of e.g. automatic part-of-speech (POS) tags and syntactic information, with pitch accents and prosodic phrasing information helps to improve coreference resolution of German spoken text. The prosodic labels used in the experiments were annotated manually, which is not only expensive but not applicable in an automatic pipeline setup. In this experiment, we reproduced the main results from the pilot study by annotating the prosodic information automatically, thus omitting any manual annotations from the feature set. We show that adding prosodic information significantly helps in all of our experiments. Furthermore, using this method, we can outperform previously reported results on the same dataset.

The pipeline setup was simulated without the ASR component. Here, we worked with the manual transcriptions because of the difficulties in evaluation discussed previously: Using ASR output would involve transferring the gold standard coreference annotations to the recognized output, resulting in mismatched datasets and preprocessing issues, as encountered in Section 4.1. Therefore, we limited the task to using automatically derived features, including the prosodic labels, so that we could more readily and accurately evaluate the experiments against the reference annotations.

In this experiment, the prosodic information was obtained using the CNN-based prosodic event detector. The work was published as a collaborative conference paper (Rösiger et al., 2017), and the two first authors contributed equally to the experimental design and written paper. Ina Rösiger implemented the coreference resolution system

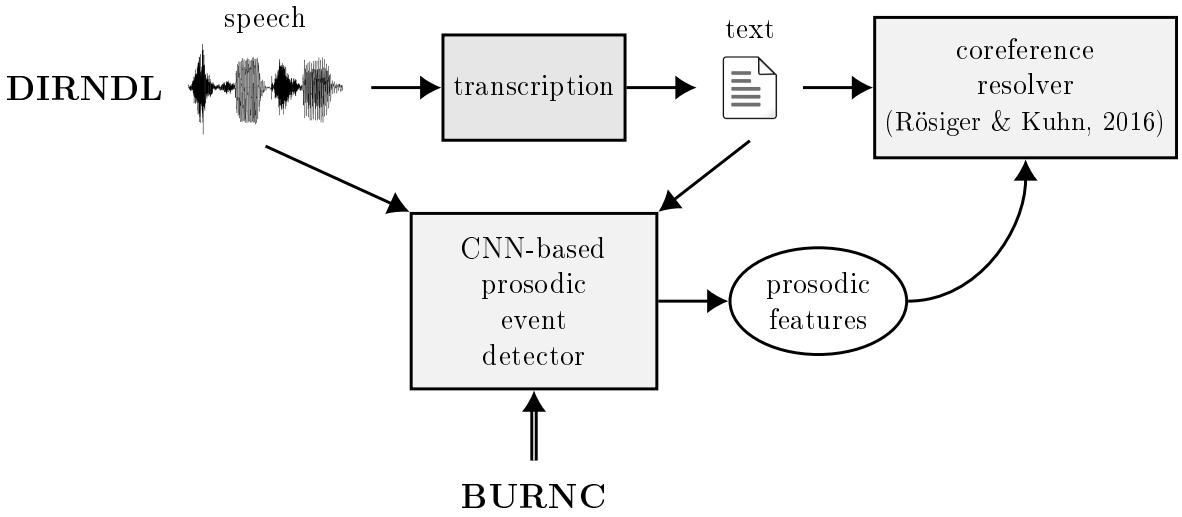


Figure 4.5: Overview of the experimental setup for coreference resolution with prosodic features on the DIRNDL corpus. The CNN-based prosodic event detector was pre-trained on BURNC.

(Section 4.2.2) and ran the experiments and analysis (Section 4.2.4). The contribution towards this thesis consisted of providing automatically obtained prosodic information (Section 4.2.3) and integrating it into the required format of the corpus used for the coreference resolver. An overview of the experimental setup for this section is shown in Figure 4.5.

4.2.2 Coreference resolution baseline

The baseline coreference resolution system consisted of a resolver for German that uses features derived from automatic annotations only.

Data To ensure comparability, we used the same dataset as in the pilot study, namely the anaphora subset of the DIRNDL corpus Björkelund et al. (2014), annotated with both manual coreference and prosody labels. We adopted the official train, test and development split[9] designed for research on coreference resolution. The recorded news broadcasts in the DIRNDL-anaphora corpus were spoken by 13 male and 7 female speakers, and amounts to roughly 5 hours of speech.

[9]http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/dirndl.en.html

IMS HotCoref DE The IMS HotCoref DE coreference resolver is a state-of-the-art tool for German (Rösiger and Kuhn, 2016). The learning method is data-driven; a structured perceptron models coreference within a document as a directed tree to create non-local features (that go beyond a pair of NPs) across the tree structure. The standard pre-computed features are text-based and consist mainly of string matching, parts of speech, constituent parses, morphological information and combinations thereof. The performance of the coreference system is measured using the *CoNLL* score, the standard measure to assess the quality of coreference resolution.

Coreference resolution using automatic preprocessing To simulate a completely automatic pipeline setup that is applicable to new texts, all annotations used to create the text-based features were automatically predicted using NLP tools. It is frequently observed that the performance drops when the feature set is derived in this manner, compared to using features based on manual annotations. For example, the performance of IMS HotCoref DE drops from 63.61 to 48.61 CoNLL score on a reference dataset consisting of German newspaper text. The system, pre-trained on the reference corpus and applied to DIRNDL, yields a CoNLL score of 37.04 using predicted annotations. This comparatively low score is in line with the observation that a system trained on written text performs worse when applied to spoken text. The drop in performance can also be explained by the slightly different domains of the two corpora, namely newspaper text and radio news. However, after training on both the training and development sets of DIRNDL, we achieved a score of 46.11. This served as a baseline in the following experiments.

4.2.3 Automatically predicted prosodic information

Pre-trained CNN-based prosodic event detector We predicted prosodic events for the whole DIRNDL subcorpus used for coreference resolution. The CNN-based prosodic event detector was used to automatically obtain binary pitch accent and phrase boundary labels. The model was pre-trained on a different dataset to simulate an actual application setting. We trained the prosodic event detector on the BURNC corpus and applied the model to the German DIRNDL corpus. Pre-training on this dataset was justified for the following reasons: First, we had shown in Section 3 that the cross-corpus performance on these two corpora are acceptable. This is made possible because the prosody of English and German is similar, and because both corpora consist of radio news. Furthermore, the acoustic correlates of prosodic events, as well as the connection between sentence prosody

and information status exploited in this paper, are similar in English and German. The prediction accuracy achieved in these experiments on the DIRNDL anaphora corpus is 81.9% for pitch accents and 85.5% for intonational phrase boundary tones. The per-class accuracy is 82.1% for pitch accents and 37.1% for phrase boundaries. Despite the fact that the quality of the automatic phrase boundary annotations is comparably low, we believe that, as a first step, their effectiveness can still be tested.

Prosodic features for coreference resolution Similar to the pilot study, we made use of both pitch accent and phrase boundary information as binary features. We did not predict the pitch accent type, e.g. fall (H*L) or rise (L*H), as this distinction was not helpful in the pilot study and is generally more difficult to model. The actual features used in the coreference feature set consisted of pitch accent features; and the phrase boundaries were used to derive nuclear accents. In this work, nuclear accents were defined as the last and thus perceptually most prominent accent in an intonation phrase. All other accents were defined as prenuclear. We tested whether the previously reported tendencies for manual labels also hold for automatic labels, namely:

1. Short NPs: Since long, complex NPs almost always have at least one pitch accent, the presence and the absence of a pitch accent is more helpful for shorter phrases.

2. Long NPs: For long, complex NPs, we considered nuclear accents as an indicator of the phrase's overall prominence. If the NP contains a nuclear accent, it is assumed to be less likely to take part in coreference chains.

We tested the following features that had proven beneficial in the pilot study. **Pitch accent presence** indicates the presence of a pitch accent, disregarding its type. If one accent is present in the NP, this boolean feature is assigned the value *true*, and *false* otherwise. **Nuclear accent presence** is a boolean feature comparable to pitch accent presence. It is assigned the value *true* if there is a nuclear accent present in the NP, and *false* otherwise. These features were derived for each NP and added as single binary features to the baseline feature set in the coreference resolver.

We defined *short NPs* to have a length of 3 words or shorter[10]. In this setup, we applied the prosodic features only to short NPs. In the *all NP* setting, the features were used for all NPs. The ratio of short vs. long NPs in DIRNDL is roughly 3:1. Note that we evaluated on the whole test set in both cases.

[10] In our experiments, this performed even better than length 4 or shorter as used in Rösiger and Riester (2015)

We compared how the performance of the coreference resolver is affected in three settings. The system was

1. trained and tested on manually obtained prosodic labels (short *gold*),

2. trained on manual prosodic labels, but tested on automatically obtained labels (this simulates an application scenario where a pre-trained model is applied to new texts (short *gold/auto*),

3. trained and tested on automatic prosodic labels (short *auto*).

4.2.4 Results

Table 4.17 shows the effect of the pitch accent presence feature on our data. All features performed significantly better than the baseline (according to the Wilcoxon signed rank test (Siegel and Castellan, 1988) at the 0.01 level. As expected, the numbers were higher when we limited this feature to short NPs. We believe that this can be explained as follows: on short NPs, a pitch accent makes it more likely for the NP to contain new information, whereas long NPs almost always have at least one pitch accent, regardless of its information status. Thus, the features contributes most when it is most meaningful. We achieved the highest performance with gold labels, followed by the *gold/auto* version with a score that is not significantly worse than the *gold* version. This is important for applications, as it suggests that the loss in performance is small when training on gold data and testing on predicted data. As expected, the version that was trained and tested on predicted data performed worse, but was still significantly better than the baseline. Hence, the prosodic information was helpful in all three settings. This also shows that the assumption on short NPs in the pilot study is also true for automatic labels.

Table 4.18 shows the effect of adding nuclear accent presence as a feature to the baseline. Again, the results were all significantly better than the baseline. The improvement is largest when we applied the feature to all NPs, that is, including long and more complex NPs. This is in line with the findings in the pilot study for long NPs. For short NPs, the nuclear accent feature does not receive the value *true* as often as the pitch accent presence feature does. Therefore, nuclear pitch accents do not provide sufficient information for a majority of the short NPs. For long NPs, however, the presence of a nuclear accent is more meaningful. The performance of the different systems using the nuclear accent feature follows the same pattern as when using pitch accent type: *gold* >

baseline	46.11	
+ pitch accent	short NPs	all NPs
+ presence gold	53.99	49.68
+ presence gold/auto	52.63	50.08
+ presence auto	49.13	49.01

Table 4.17: Effect of pitch accent presence on coreference resolution performance, comparing the application to only short NPs or to all NPs. The performance is improved in all cases, with a slight decrease in quality the more automatically obtained prosodic information is added.

baseline	46.11	
+ nuclear accent	short NPs	all NPs
+ presence gold	48.63	52.12
+ presence gold/auto	48.46	51.45
+ presence auto	48.01	50.64

Table 4.18: Effect of the nuclear accent presence feature on coreference resolution performance. This feature is more powerful when for use on NPs of all lengths.

gold/auto > *auto*. Again, the automatic prosodic information contributed to improving the system's performance.

The highest score when using automatic labels is 50.64, as compared to 53.99 with gold labels. At the time of publication, these were the best results reported on the DIRNDL anaphora dataset.

Examples In the following we discuss two examples from the DIRNDL dataset that provide some insight as to how the prosodic features helped to improve coreference resolution in our experiments. The first example is shown in Figure 4.6. The coreference chain marked in this example was not predicted by the baseline version. With prosodic information, however, the fact that the NP *"der Koalition"* is deaccented helped the resolver to recognize that this was given information: it refers to the recently introduced antecedent *"der Großen Koalition"*. This finding supports our assumption that the absence of pitch accents helps to resolve short NPs.

An additional effect of adding prosodic information concerns the length of antecedents (i.e. NPs mentioned before the candidate NP) determined by the resolver. In several cases, e.g. in example (4), the baseline system incorrectly chose an embedded NP (1A) as the antecedent for a pronoun. The system with access to prosodic information correctly chose the longer NP (1B). Our analysis suggests that this was a result of the pitch accent

EXPERTEN	{der Großen KOALITION}$_1$	haben	sich	auf [...]	ein	
Experts	*(of) the grand coalition*	*have*	*themselves*	*on*	*a*	

Niedriglohn	Konzept	VERSTÄNDIGT.	Die strittigen Themen [...]		sollten	bei
low wage	*concept*	*agreed.*	*The controversial topics*		*shall*	*at*

der	nächsten	Spitzenrunde	**{der Koalition}$_1$**	ANGESPROCHEN	werden.
the	*next*	*meeting*	*(of) the coalition*	*raised*	*be.*

EN: Experts within the the grand coalition have agreed on a strategy to address [problems associated with] low income. At the next meeting, ***the coalition*** *will talk about the controversial issues.*

Figure 4.6: Example from the DIRNDL dataset with English translation. The candidate NP (anaphor) of the coreference chain in question is marked in boldface, the antecedent is underlined. Pitch accented words are capitalized.

on the short NP (*Phelps*). The presence or absence of a pitch accent on the adjunct NP (*USA*) does not appear to have an impact.

(4) {{Michael PHELPS}$_{1A}$ aus den USA}$_{1B}$. **{Er}**$_1$ ...
Michael Phelps from the USA. He ...

4.2.5 Conclusion

In this collaborative experiment, we showed that using automatically obtained prosodic labels significantly improves the performance of a coreference resolver. We predicted these labels using the CNN-based prosodic event detector and used them as additional features in a coreference resolution system for German. Despite the fact that the quality of the predicted labels was slightly lower than the manual labels, we confirmed previous results of including manually annotated prosodic information. This encouraging result also confirms that not only is prosodic information helpful for coreference resolution, but that it also has a positive effect even when predicted by a system.

The implication for this thesis is that the CNN-based method, pre-trained on a different dataset (BURNC), can be readily integrated into a pipeline setting. Furthermore, it can provide useful information in the form of symbolic prosodic event labels for a traditionally text-based downstream task.

To pursue coreference resolution directly on speech, a future step would be to perform all necessary annotations on ASR output. This still poses a challenge due to word recognition errors and the fact that an ASR system does not, by default, provide punctuation,

which is important cue for text-based coreference resolution. As a first step, our results on German spoken text are promising and we expect them to be generalizable to other languages with similar prosody.

4.3 Prosodic event detection for corpus annotation

The third application of automatic prosodic event detection that we investigated is the annotation of large datasets. The motivation behind this use is the fact that manual prosodic labeling is expensive and requires trained experts. Another issue is the comparably low inter-annotator agreement (Pitrelli et al., 1994). Therefore, it is desirable to have reliable, automatic annotation methods to aid linguistic and speech processing research on a large scale. While automatic labeling will unavoidably produce errors, the gain in efficiency makes it a realistic alternative, especially when considering the fact that the level of agreement between human labelers has been compared to the agreement with an automatic labeler (Elvira-García et al., 2016).

The CNN-based prosodic event detector, although not initially designed for this purpose, was used to create prosodic event labels for a German radio interview corpus (GRAIN). This resource, which was annotated using several automatic tools, was introduced in a collaborative conference paper (Schweitzer et al., 2018). The first prosodic annotation layer was provided by the PaIntE-based method by Schweitzer (2010). Our contribution consisted of creating an additional prosodic annotation layer consisting of a binary pitch accent label for each word using the CNN-based method, thereby providing additional information to increase the confidence of the prosodic annotations. Furthermore, we hand-labeled a small sample of the corpus in order to evaluate and compare the prosodic annotations in the corpus. In the publication describing the official first release, phrase boundary annotations derived from the CNN-method were listed as annotation layers under development. They have been since provided but are not included in the initial version of the corpus. These annotations were obtained in the same manner as the pitch accents, however the annotation quality has not been estimated.

This section first describes the GRAIN corpus and the prosodic annotation layers, and then reports on the measured agreement between these layers. Since the prosodic labeling methods used for the different layers were both used in this thesis, this section also serves as a relevant comparison between these two methods. A discussion on the utility of the CNN-based method for corpus annotation concludes this section. An overview of the experimental pipeline setup for the annotation and analysis described in

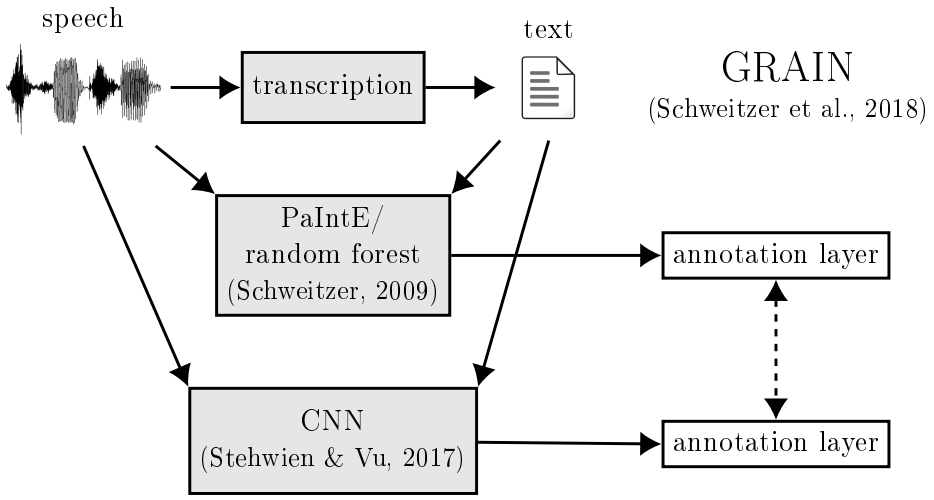


Figure 4.7: Overview of the prosodic annotation pipeline for the GRAIN corpus. The CNN-based prosodic event detector was pre-trained on German speech data and the resulting annotations were compared to those obtained by a PaIntE-based prosodic event detector.

this section is illustrated in Figure 4.7.

4.3.1 The GRAIN silver-standard corpus

The GRAIN corpus is a collection of German radio interviews that have been both manually and automatically annotated using state-of-the-art tools for speech and text processing. The interview data, although belonging to the domain of spontaneous speech, is comparably clean since it consists of interviews with one professional and one non-professional, but experienced, public speaker. The data consists of interviews of around 10 minutes each, downloaded along with the corresponding edited transcriptions from the radio station's website. The first official release of the corpus[11] consists of 140 interviews of about 23 hours of audio and 221,000 word tokens.

The GRAIN corpus attains to two state-of-the-art concepts for linguistic corpora: the idea of a *non-static* resource as well as a *silver standard* corpus. Non-static in this context refers to the goal of a continuously growing resource, extended as the radio station releases more interviews. The term *silver standard* sets the corpus apart from the typical *gold standard*, in that the annotations were not created manually, but by applying automatic text- and speech processing tools. Thus, the idea of silver standard data is to overcome the lack of human-annotated data required for research in speech

[11] https://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/grain.html

and language processing. The goal is to combine several tools and annotation layers pertaining to the same linguistic annotations in order to improve the quality of the resource. Incorporating several annotation layers from different sources is intended to make the annotations more precise, since the overlap that different tools create should, ideally, increase the confidence of the correctness of the annotation. The confidence estimations are provided as an additional layer and can be used for visualization and search purposes, for example for finding areas with especially high or low confidence.

20 interviews were chosen as the gold standard (i.e. annotated manually) and the rest comprise the "silver standard" subset of the corpus. The annotation layers included in the release include segmentation at the word- syllable- and phone level, as well as sentence segmentation. The data is tokenized, part-of-speech tagged, and parsed for dependency and constituency by several tools. In the following, we describe the two annotation layers for prosodic events.

4.3.2 Prosodic annotation layers

The prosodic annotations come from two very different tools that were both pre-trained on German radio news data. The PaIntE-based predictor, described in Section 4.1.2 takes several fine-grained and linguistically motivated features, and annotates GToBI(S) prosodic event types at the syllable level. This model was pre-trained on German read speech (Schweitzer, 2010).

The CNN-based prosodic event detector operates at the word level only, predicting only the binary presence or absence of a pitch accent, but therefore requiring much less preprocessing. The CNN-based pitch accent detector was pre-trained on the German radio news corpus DIRNDL. The corpus was converted to 16kHz .WAV format from the original .mp3 format and the 6 acoustic features (20ms/50ms frames) were extracted and z-scored. The network was trained for 30 epochs with a dropout rate of $p = 0.8$. The remaining settings were the same as in the baseline method described in Section 3.2. The model was applied to the time-aligned GRAIN corpus to obtain word-level annotations.

Even though no type of pitch accent was assigned, this layer makes it possible to apply the silver standard idea to prosodic annotations: The placement of the automatically predicted pitch accent labels derived from PaIntE features can be compared and combined with this layer for confidence estimations, which we show in the next section.

4.3.3 Agreement on pitch accent placement

Since GRAIN contains no manually annotated intonation events, the prediction accuracy was estimated on a small amount of data labeled by a human expert. This analysis estimates the agreement of word-level pitch accent placement between the two annotation layers (PaIntE-based and CNN-based) and compared to human annotation. For this purpose, we hand-annotated one interview file containing 1938 words and 509 reference pitch accents.

Table 4.19 shows the performance of the two tools measured against the reference annotations. In terms of accuracy, both tools provided annotations of similar quality, namely around 80%. This confirms that both pre-trained models labeled the pitch accents with acceptable quality. The main difference between the two methods is that the CNN-based tool yields a higher recall, while the PaIntE-based annotations have a higher precision. This is likely a result of the fact that the PaIntE-based method uses more linguistic information for making predictions.

Additional numbers not shown in the table are the following: Both automatic tools show an agreement of 77.2% in terms of overall accuracy on both classes, and all three sources (CNN-based, PaIntE-based, and manual) agree on 53.6% of pitch accents. This means that slightly over half of the pitch accents in the dataset can be correctly localized, with a high confidence, using the overlapping predictions that both tools produce. Summing the correct predictions of either tool, that is, taking into account predictions with a lower confidence, results in a 88.4% coverage of the manually annotated pitch accents.

Table 4.20 compares the precision on either label classes *accent* and *none* obtained using both methods and when counting only the labels that the two tools agree on. These results show that in cases in which both annotations agree with respect to the presence or absence of a pitch accent, the precision increases considerably for both classes (up 10% for the accent class) compared to when used alone. These results are in line with the silver standard idea in that the confidence of the annotations increases (especially in terms of precision) when both annotation layers are taken together.

4.3.4 Discussion

The interview domain is challenging for prosodic annotation since it not only consists of spontaneous speech, but of dialogue recordings that involve turn-taking. This does not necessarily pose a disadvantage to the CNN model, as it was shown to generalize

	PaIntE-based	CNN-based
# accents	494	629
accuracy	80.9%	80.0%
precision	67.2	62.2
recall	65.2	76.8
F1-score	66.2	68.7

Table 4.19: Agreement with human labeling of word-level pitch accent placement: The PaIntE-based and CNN-based automatic annotations are compared on one example file containing 1778 words and 509 pitch accents.

class	PaIntE-based	CNN-based	both
accents	67.2	62.2	76.0
none	86.2	89.7	94.2

Table 4.20: Precision for both label classes *accent* and *none* obtained using either pitch accent labeling method and when taking only the labels into account on which both tools agree.

well across speakers in Section 3.4. The fact that the speaker turns were not annotated, however, affects the acoustic context information collected for each word. For the sake of efficiency, we did not separate speaker turns during preprocessing, and the results show that this was also not needed; the method still performed comparably well. This could even be regarded as an advantage of using a comparably local context of only 3 words, as the CNN-based method does, as opposed to using entire sentences, which would first require the data to be segmented according to speaker turns. We also found that normalizing the features per utterance still performed better than using non-normalized features, even though each utterance contained speech of two different speakers. This was an unexpected finding, and would require further investigation beyond the scope of this work.

While the evaluation in this section can only provide an estimation of prosodic event detection performance (especially when using only one human labeler), the reported numbers do show that both tools, using entirely different methods of prosodic modeling, complement each other. It also demonstrates the idea of a "silver standard" which was implemented in GRAIN: combining two layers of automatic annotation on the same annotation level yields a better annotation quality than just one level alone. A more fine-grained annotation, as provided by the PaIntE-based method, can be considered more appropriate for linguistic resources, but is costly to implement and use. Combined

with the more efficient neural-network-based method, which provides a more coarse labeling, yields predictions with a higher confidence. This implies that not only can the CNN-based prosodic event detector be used to provide labels for understanding tasks, but that it can also contribute to creating corpus resources that can be used in the development of speech and language applications.

4.4 Conclusion

In this section, we considered three different examples of applications in which automatic prosodic event detection can provide useful information. Two typically text-based tasks, slot filling and coreference resolution, were augmented with automatically obtained prosodic labels, which yielded promising results. The role of prosody for both tasks can be traced back to its function in marking *information structure*: The coreference resolution experiment exploits the relationship between prosodic prominence and the distinction between given and new information, and the correlation between pitch accents and semantic slots is connected to focus. We also presented our contribution to the GRAIN silver standard corpus in which we provided prosodic event labels to increase the confidence of automatic annotation layers. In all three scenarios, the prosodic event detection models were pre-trained on different corpora. The nevertheless acceptable performance levels support their use as standalone modules in pipeline systems.

All scenarios described in this section use different experimental pipeline setups, combining neural with non-neural methods and consisting of several preprocessing and modeling steps. A substantial amount of preprocessing was needed for each step, because the datasets that were used for the originally text-based downstream tasks (ATIS slot filling benchmark; DIRNDL anaphora corpus) were not initially designed for prosodic analysis. For this reason, a simple method such as our CNN-based prosodic event detector greatly reduces the effort required to obtain prosodic information.

The slot filling experiments did not involve the CNN-based method, but employed a method by Schweitzer (2010) that makes use of more higher-level, linguistically motivated features, such as PaIntE parameters. The experiments demonstrate the difficulty of using methods that require extensive preprocessing in a pipeline setup. For example, in order to use the PaIntE-based method on BURNC and ATIS, we reduced the feature space to only acoustic features, since part-of-speech tags are language- and sometimes even corpus-specific, and since they first need to be acquired by means of an automatic tagger. The CNN-based method, since it uses simple frame-based features, is much more

readily applied to new datasets.

Both slot filling and coreference resolution benefit from the knowledge of prosodic events at the word-level. In this case, the efficiency and simplicity of the CNN-based method makes it a suitable choice in an assumed pipeline setting. For the corpus annotation task, the binary word-level prosodic event labels may be considered too coarse, as many applications that aim to answer linguistic research questions require a more fine-grained labeling. Thus, the PaIntE-based annotation method may be a more appropriate tool. However, the results have shown that the CNN-based method can be useful for creating a the silver standard, and possibly also for providing more coarse-grained labels before a more detailed analysis can be applied.

In sum, we have provided further evidence that using automatic methods of prosodic event detection can help improve downstream tasks for speech understanding. The challenges and necessary steps to set up these experiments has further motivated our notion of an efficient prosodic event detector that is readily inserted into a speech processing pipeline. Since the CNN-based prosodic event detector has shown to provide useful information while attaining to this notion, we conclude that we have met our goal of developing an efficient and flexible method.

One advantage of the approach taken in this work, namely a symbolic modeling approach in a pipeline setting, is that it allows us to analyze the trained prosodic event detection models. We thus turn our attention to the analysis of the neural network in the next section.

5 Analysis of learned feature representations

The previous two sections focused, respectively, on the use of convolutional neural networks for prosodic event detection, and on their application for automatically obtaining prosodic labels as features for speech and language understanding tasks. In this section, we focus on the analysis of the trained models.

The largest difference of this method over typical machine learning methods of prosodic event detection is the fact that the neural network is able to learn higher-level feature representations from lower-level input. Since one of the main motivations is to replace human feature engineering, the question arises as to what exactly the learned feature representation consists of. Therefore, the analysis in this thesis focuses on the learned feature representation.

We first describe our method of analysis in Section 5.1. This procedure is a simple and task-agnostic way of estimating what a neural network has learned without the need to implement costly methods of analyzing the entire network.

The quantitative analyses of the CNN performance described in Section 3 led to questions on what acoustic, temporal and context information the model can learn on its own. We show that the CNN can learn duration for pitch accent detection in Section 5.2. Section 5.3 focuses on the importance of individual acoustic descriptors for pitch accent and phrase boundary detection. Finally, in Section 5.4, we take a closer look at the amount of acoustic and temporal context information that is learned by the CNN and compare it to the LSTM introduced in Section 3, which, as a sequential model, is specifically designed to take context into account.

5.1 Linear regression analysis of neural network output

Our method of analyzing what the neural network has learned involves using linear regression to determine what information is latent in the output feature representation of the CNN. We first defined a set of candidate features that we hypothesize could be encoded in, or correlated with, the CNN output. These features consist of the aggregated features describing information that the network may have learned from the frame-based acoustic input. We then extracted the output feature representation of the CNN, that is, the output of the max pooling layer as shown in Figure 5.1. This yields an n-dimensional vector for each data point, which we refer to as the *CNN features*. We trained a simple linear regression model (*lm*) in R (R Core Team, 2013) with the n CNN features as predictors and one candidate feature as the dependent variable. In this work, we chose different representations of aggregated acoustic information across each input word in the context matrix (w_{prev}, w_{cur}, w_{next}) as well as the duration of each word (depicted in the top part of Figure 5.1). We measured the goodness of fit using the adjusted R^2 measure, which, taking all predictors into account, indicates how much of the variance of the dependent variable can be explained by the predictors. If these features can be predicted or at least approximated using the linear models trained on the CNN output, we can conclude that the CNN has learned this information, and that it is latent in the learned feature representation.

5.2 Duration as a learned feature in the CNN

In Section 3, we compared the CNN to a model that does not have access to this information, namely a simple feed-forward neural network (FFNN) with acoustic features manually computed for entire words. The results showed that the FFNN can benefit from adding temporal information manually, while the CNN does not. Judging from these results, we assumed that the CNN can learn from the implicit access to temporal information provided by the position indicators. In this section, we use the linear model analysis to verify whether word duration is latent in the CNN output representation.

We tested two assumptions on various prosodic detection tasks on German radio news speech: First, since the duration of the input words or syllables is provided implicitly as the length of the frame sequence, we assume that it is likely that the CNN is making use of this information. Second, we expect that this depends on the method of pooling

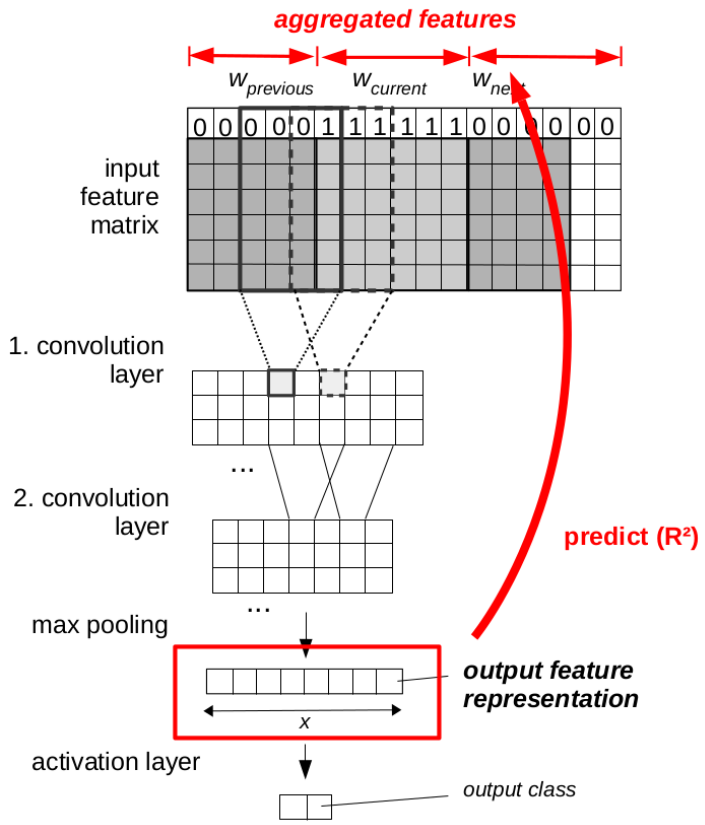


Figure 5.1: Analysis of neural network output using linear regression: A linear model is trained on the n output features of the neural network to predict individual candidate features that represent the input information, e.g. aggregations of the frame-based acoustic features across the input words. The neural network is said to have learned certain information if it can be approximated by the linear model according to the adjusted R^2 measure.

and padding that we implemented in the prosodic event detector.

We compare not only word-based pitch accent detection, but also similar, syllable-based tasks that we assume to have different correlations with the duration of the segment in question: pitch accent classification and lexical stress detection. The latter is performed using the same method since the automatic detection strategies for pitch accents and lexical stress are often comparable (Shahin et al., 2016; Tepperman and Narayanan, 2005).

5.2.1 Experimental setup

All experiments were conducted on the DIRNDL corpus. We used the word- and syllable-level annotations as separate datasets for different tasks: Pitch accent detection distinguishes between the *accent* and *none* classes on either words or syllables. At the syllable level, we also performed lexical stress detection (primary stress) and classification of the following pitch accent types: H* (high pitch accent), H*L (falling pitch accent) and L*H (rising pitch accent). For the type classification tasks, we left out all syllables belonging to the negative class since we were interested in the distinction of given accents. Table 5.1 lists the number of words, syllables and the relevant prosodic events in the dataset.

words	35347	syllables	76396
accented	18137	stressed	35572
		accented	18958
		H*	16510
		H*L	6115
		L*H	7815

Table 5.1: Overview of annotations used in the DIRNDL dataset

The data was divided into training, development and test splits with 10-fold cross validation. 500 held-out words were used as the development set and the test sets consisted of 1000 words.

We used applied the same method developed for word-based pitch accent detection to syllable-based lexical stress detection, which only involves using different output labels and a context window of input syllables instead of words. The input to the CNN consisted of the 6 acoustic features (20ms/50ms frames) normalized per utterance and position indicator features. The output of the CNN is a feature vector whose dimensionality n depends on the method of pooling applied after the second convolution layer. In the next section we compared the default method of using 1-max pooling with the position indicators to 3-max pooling *without* position indicators. We used a dropout rate of $p = 0.8$ in all settings.

5.2.2 Comparison of pooling methods

1-max pooling with position indicators 1-max pooling is a simple selection of the maximum value in the 2-dimensional pooling window which spans the length of each

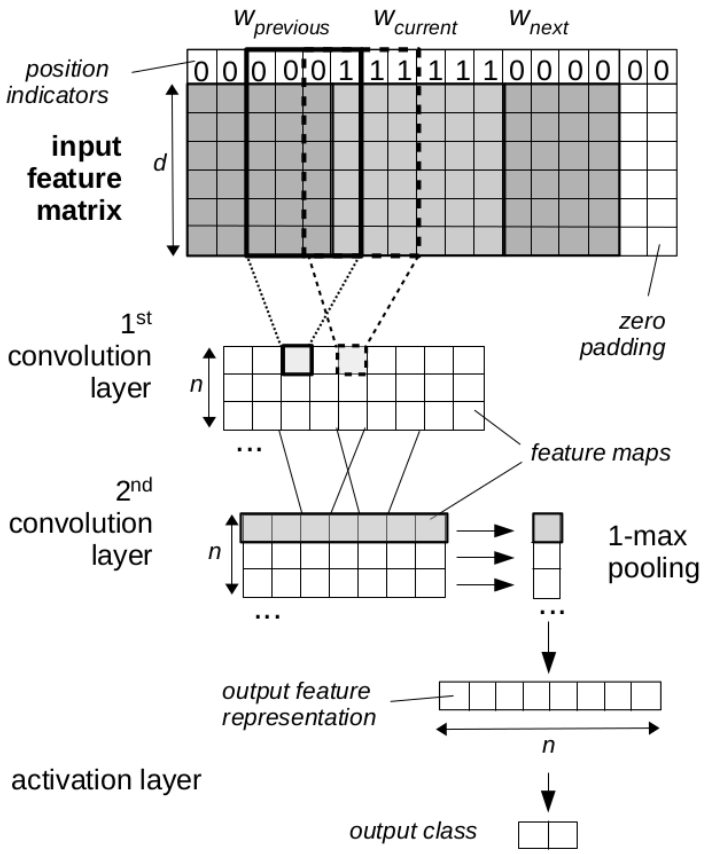


Figure 5.2: CNN for binary classification (2 output units) using 1-max pooling. The input is a 3-word context window of acoustic and position indicator features.

feature map (see Figure 5.2). The output of the pooling layer is a feature matrix of the shape $n \times 1$, which corresponds to the number of kernels in the last convolution layer. In Section 3.4 had shown that the pitch accent detector relies on a position indicator feature that distinguishes frames that pertain to the current word from those of the preceding and following context words. This can be explained as follows. Since the input words have different lengths, the input is padded with zeros so that each input matrix has the same size. The padding was concatenated to the end of the matrix. Therefore, there is no fixed position that corresponds to each individual input word. The position indicators help the model select features that are most important for representing the current word. They were added as an additional feature to the acoustic input and consist of ones for $w_{current}$ and zeros for the neighboring words. In the first convolution layer, we set the kernel size to span the complete feature dimension $(d+1)$ so that the position indicators are constantly taken into account.

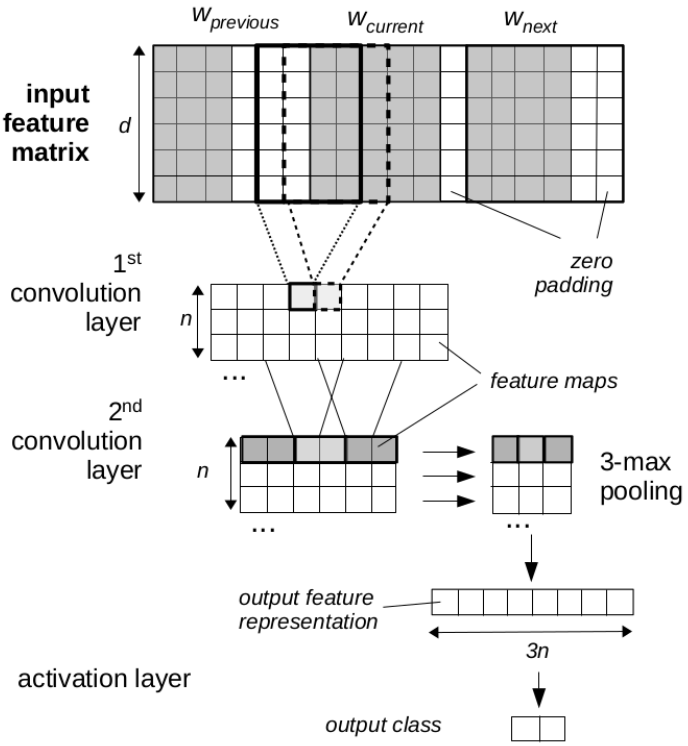


Figure 5.3: CNN for binary classification and 3-max pooling. Each input word is padded separately.

3-max pooling We compared the above method to an alternative that we refer to as 3-max pooling. In this setting, we zero padded each word in the input matrix separately so that each word has the same size (see Figure 5.3). Instead of pooling over each entire feature map, we defined three equal sized pooling windows for which one maximum value is selected. This method ensures that the learned feature representation contains information on all three words. Therefore, the position indicator features should not be required. The resulting flattened feature vector has the length $3n$. In these experiments, we used 30 feature maps, that is the output vector has a length of 90 and is thus comparable in size to the 1-max pooling output.

In sum, we obtained two different pooling methods and two different ways of padding the input. In the following experiments, we tested various combinations with and without the position indicator features.

Results Table 5.2 shows the F1-score for word-based pitch accent detection. As shown in Section 3.4, 1-max pooling performs best using position indicators. The model does not perform well if they are left out, however, the performance can be increased if

the padding is placed in between the words. This is likely explained by the fact that the padded areas give an indication of word boundaries. In contrast, the method of using 3-max pooling, as we had assumed, makes the position indicators unnecessary since each word has a fixed position in the input matrix and a corresponding pooling value. Overall, although 1-max pooling combined with the position indicators yields slightly better results, the performance of both methods is comparable. One possible disadvantage of pooling over each word separately is that the zero padding breaks up the input signal. However, this still may not explain the differences observed in these experiments. Instead, we believe that a more straightforward explanation is that the length n of the output feature representation affects the performance: Using $n = 100$ for 3-max pooling (that is, the final feature vectors has a length of 300), for example, increases the F1-score by roughly 1 percentage point.

setting		+ pos.-ind.	- pos.-ind.
1-max pooling	padding at end	**86.8**	63.4
	padding between words	86.2	70.4
3-max pooling	padding between words	85.8	**86.0**

Table 5.2: Performance (F1-score) of word-based pitch accent detection using 1-max pooling ($n = 100$) and 3-max pooling ($n = 30$)

Comparison of task performance Next we applied the two best settings, that is, 1-max pooling with position indicators and 3-max pooling without, on all word and syllable-level tasks. The results are listed in Table 5.3. The numbers for word-based pitch accent detection are the same as in Table 5.2 and included for comparison. Across all tasks, 1-max pooling performs slightly better than 3-max pooling, namely around 1-2 percentage points. The only exception is syllable-based pitch accent detection (2), where this difference is larger.

Syllable-level pitch accent detection is more difficult than word-based pitch accent detection for various reasons. First, the classes are less balanced (see Table 5.1), which makes it harder to model the minority class. Second, since the words are simply replaced by syllables in this case, much less context is provided (as discussed in (Rosenberg and Hirschberg, 2009)). Another reason, assuming that the model can learn duration, may be that the model cannot make use of any correlations to word length that could be attributed to word identity or part of speech.

Lexical stress detection (3) is an easier task, which is attributed, in part, to the fact

that the classes are more balanced. Pitch accent detection is facilitated when only stressed syllables are considered (4). In this case, the model does not have to learn to distinguish unstressed and therefore unaccented syllables, which greatly reduces the number of negative examples in the data. For the classification of pitch accent types, we restrict the number of datapoints further and consider only syllables which carry a positive label. The three-way classification (5) is considerably more difficult than the binary distinction in (6), not only because of the multi-class learning problem but also since the H*-class makes up around half of the labels in (5) and thus constitutes a proportionally large majority class.

It should be noted that since the model was applied "out of the box" to pitch accent type classification and syllable-level stress detection, we did not expect the performance to be very high. Instead, we used this to draw a comparison between the two pooling method on each task, and to obtain reference values for the next analysis.

Task	1-max	3-max	*maj. class*
word-based			
(1) PA detection	86.8	86.0	*51.3*
syllable-based			
(2) PA detection	60.0	55.0	*24.8*
(3) stress detection	69.0	67.0	*46.6*
(4) PA detection stressed-only	75.0	73.0	*53.3*
(5) H*/H*L/L*H	66.0	65.0	*54.2*
(6) H*L/L*H	72.0	70.0	*56.1*

Table 5.3: F1-scores and majority class sizes for various pitch accent (PA) and lexical stress detection tasks at the word and syllable level. The table compares 1-max pooling with position indicators to 3-max pooling without position indicators.

5.2.3 Evidence of duration in CNN output representations

After testing the performance of both pooling methods on the various tasks, we applied the linear regression analysis to determine whether the trained models learned to encode word or syllable duration. The CNN features consist of an n-dimensional vector for each data point. In this case, we obtain a vector of length $n = 100$ for the 1-max setting, and $3n = 90$ for 3-max pooling. We used this data to train a linear regression model to predict the duration of each of the three input words (w_{prev}, w_{cur}, w_{next}) and measured the goodness of fit using the adjusted R^2. Using this method, we compared models

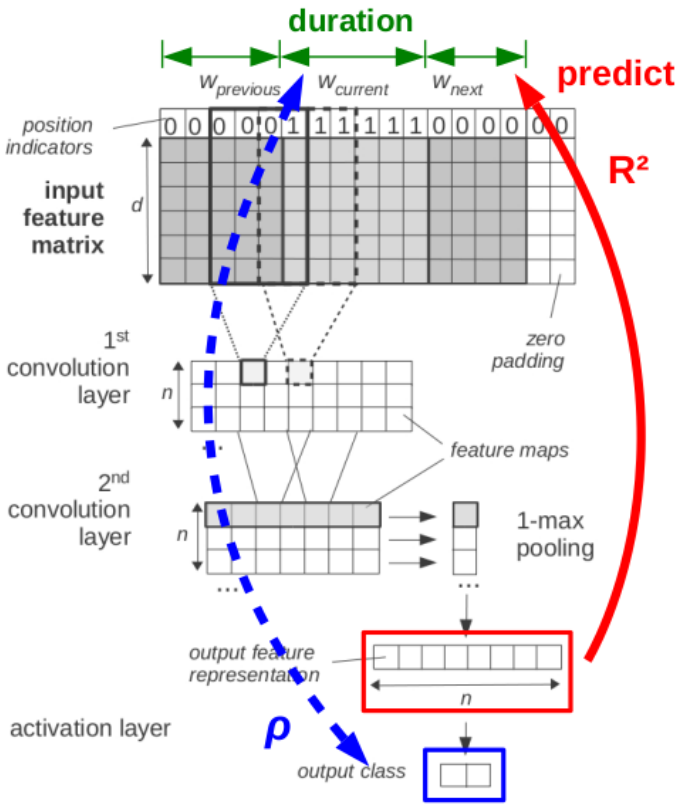


Figure 5.4: Linear model analysis for duration

trained using 1-max pooling and position indicators and 3-max pooling without position indicators, assuming that the duration of the three input words (or syllables) was learned differently. We expected to find differences across the various tasks listed in Table 5.3, where duration may be more or less important. For comparison, we also measured the Spearman correlation (using R) between the target label, that is, the respective pitch accent or stress label, and the duration of w_{cur} (see Figure 5.4).

The results of the analysis for each task are shown in Table 5.4. An important finding is that for word-based pitch accent detection (1), the linear model yields a moderately good fit (0.64). Compared to this, the R^2 for the duration of the current syllable is lower ($<$0.50 in tasks 2-4), but still high enough to be considered an approximate fit. Based on this result, we conclude that the CNN can learn duration, even if it is not directly included as a feature.

To show that this effect is caused by the position indicators, we added results for 1-max pooling without them for comparison. The R^2 for w_{cur}, in this case, is much lower. This may lead to the assumption that the position indicators simply make the

CNN "ignore" the context, however, our previous experiments in Section 3.4 showed that this performed better than when not using any context at all. We investigate this idea further in Section 5.4.

Across tasks, the correlation (ρ) between the duration of w_{cur} and the target label appears to explain how well the former is predicted by the linear models. This confirms that duration is an important feature that the CNN can learn on its own, even though it is only implicitly included in the input. The comparison of correlations also shows that duration is more indicative of pitch accents for words than it is for syllables. This is likely because syllables display a much lower variation in length than words. Nevertheless, the correlation in row (3) reflects the fact that there is a difference in length between accented and non-accented syllables, even when considering only stressed ones. Interestingly, duration appears to be less important for detecting lexical stress (2), although the lengthening of segments and syllables is one of the main correlates of stress. Finally, we did not measure a correlation between the syllable duration and target label for the tasks that classify given pitch accent types (5-6). Thus, the CNN features do not appear to encode much of this information and yield a low R^2 for predicting syllable duration.

Next, we compared how well duration was learned when using 1-max and 3-max pooling. In addition to the similar performance levels, both pooling methods lead to a similar R^2 on w_{cur}. The only notable difference is the fit on w_{prev} and w_{next}. When using 1-max pooling, the CNN can not be said to learn the duration of the two context words, since the R^2 is very low. For 3-max pooling, however, these numbers are increased. In this case, the CNN features show evidence of containing duration information for all three input words, but similar to 1-max pooling, the duration of w_{cur} yields the best fit. This is an interesting result, since there were no position indicators used in this setting. Thus, the CNN appears to have automatically learned features pertaining mainly to the "correct" word or syllable.

5.2.4 Summary

These experiments showed that the feature representations learned by a CNN trained to detecting pitch accents or lexical stress on the DIRNDL corpus contain latent representations of word and syllable duration. The observation that the CNN does not benefit from adding duration as a feature, along with the findings from the linear regression analysis, indicates that the CNN can learn word duration on its own, even though it is only implicitly included in the frame-based input features. This also holds for syllable-

task/setting		1-max pooling	3-max pooling
measure	ρ	R^2	R^2
duration	w_{cur}	w_{prev} / w_{cur} / w_{next}	w_{prev} / w_{cur} / w_{next}
word-based			
(1) PA detection	0.70	0.11 / 0.64 / 0.09	0.48 / 0.61 / 0.41
- pos.-ind.		0.06 / 0.14 / 0.06	
syllable-based			
(2) PA detection	0.34	0.06 / 0.42 / 0.06	0.16 / 0.40 / 0.18
(3) stress detection	0.22	0.11 / 0.31 / 0.07	0.31 / 0.38 / 0.32
(4) PA detection str.-only	0.36	0.09 / 0.40 / 0.06	0.24 / 0.38 / 0.21
(5) H*/H*L/L*H	-0.04	0.05 / 0.16 / 0.12	0.21 / 0.19 / 0.15
(6) H*L/L*H	-0.04	0.03 / 0.14 / 0.07	0.09 / 0.17 / 0.08

Table 5.4: R^2 of predicting word duration using the output of CNN models trained on various tasks. The table compares 1-max pooling with position indicators ($n = 100$) and 3-max pooling without position indicators ($n = 30$). ρ refers to the Spearman correlation between the duration of w_{cur} and the target label (pitch accent or stress).

based pitch accent and lexical stress detection as long as it helps to solve the respective task. We compared two pooling methods and showed that pooling over each of the three input words or syllables, as opposed to simple 1-max pooling, increases the amount of context information captured by the CNN. Most information that is learned, however, pertains to the current word or syllable.

5.3 Analysis of acoustic feature representations

Next we applied our method of analysis in order to explore the acoustic information learned the CNN models. In Section 3.6, we tested the acoustic features individually, showing that some features tend to be more useful than others for detecting pitch accents and phrase boundaries. However, the question of their relative importance in the CNN when used together as a full feature set is not straightforwardly answered. In this section, we take a closer look at the latent acoustic representations in the CNN output. We also extend the linear regression analysis in the "reverse" direction, as described in the following.

5.3.1 Two-way linear regression analysis

The analysis consists of two parts: In the first part, which corresponds to the method used in the previous section, we trained linear models on the CNN to predict one aggregated feature each as the dependent variable, and measured the goodness of fit. The aggregated features were not scaled (and the CNN features did not require scaling).

The second part is the "reverse" analysis. In this case, we trained the linear models on a set of aggregated features (that is, they are used as independent variables) and predicted individual CNN features. This yields 100 linear models in total. Each model is evaluated using adjusted R^2. The models were trained without interactions and the aggregated features were scaled beforehand. The aim of this analysis was to assess the significance of each aggregated feature as a predictor of each CNN feature. By counting the frequency with which each predictor is significant ($p < 0.01$) in the 100 models, we obtained an estimation of how well it is correlated with the information contained in the CNN feature vector.

This method requires the specification of features to be predicted by a linear regression model trained on the CNN output representations. We defined a set of temporal and acoustic features as "candidates" to be tested by predicting them using the method described above. It is important to note that this is merely an approximation of what information is potentially encoded in the CNN output representation, since we cannot know for sure what the model is learning. Furthermore, the network is designed to learn representations that are not straightforwardly interpreted. The candidate features chosen for our analysis included the 120 aggregated acoustic features described in Section 3.8.2: minimum, maximum, mean and standard deviation of each acoustic descriptor across each word, as well as delta features that represent the difference of each aggregated feature between the current word and the two context words. Since we had already shown that duration is latent in the CNN output, we also added the raw duration for each word in the input window (w_{prev}, w_{cur}, w_{next}) for comparison.

A first runthrough of this analysis did not show any F0 features to be significant predictors, which is why we added 7 additional, more fine-grained F0 features to the candidate feature set. We assumed that simply computing e.g. the maximum or mean across the whole word may not reflect what the CNN is learning. Three features described the steepest intra-word slope for each input word, which is the largest difference across each 6-frame interval in a word. This was motivated by the kernel size in the first convolution layer of the CNN, which consisted of 6 frames. Four features described the steepest inter-word slope across each word boundary: two rise features, that is the

difference between the minimum F0 value in one word and the maximum F0 value in the next word, and two fall features, that is the difference between the maximum F0 value in one word and the minimum F0 value in the next word.

Thus, we obtained a final set of 130 aggregated features. The position indicator feature, which is always part of the CNN input, was also not part of the manual feature set. We assumed that its contribution is quite large given the effect it has on the model performance (Stehwien and Vu, 2017b). However, since its role is clear and its importance already shown, we did not include it in this analysis.

5.3.2 Experimental results

We carried out the analysis in both directions (predicting aggregated features as well as CNN features) on the CNN output representation obtained for all tasks described in this paper, that is the pitch accent and phrase boundary detection models for all three datasets. Since including pause information appeared to make a slight difference in performance on the BDC dataset, we also compared the two preprocessing settings in this section.

We first describe the results of using CNN features to predict aggregated features. Table 5.5 shows which aggregated features were predicted best, and how well the duration of each input word was predicted according to R^2. Two main observations can be made for all datasets: For the pitch accent models, the best features were mostly the voicing probability of the current word w_{cur}. We believe that this feature is important since it indicates voiced regions of speech, which are a requirement for the presence of pitch accents. It may also help distinguish words from pauses. For phrase boundaries, energy features pertaining to the following word w_{next} were predicted best. This is also in line with our expectations, since Section 3.6 had shown that energy was one of the best features for phrase boundary detection. Furthermore, since phrase boundaries are located on the last word of an intonation phrase, it is helpful to consider the following acoustic context.

The duration of the current word could be approximated with an R^2 between 0.2 and 0.4 in the versions without pauses. As in the above section, this finding supports our conclusion that the CNN has learned this information implicitly from the frame-based input. We even found the model fit for the duration of w_{cur} to be stronger when pause transcriptions were used (up to almost 0.7 R^2, shown in bold). In three out of the six cases in which we use pause transcriptions, the duration of the current word was the best predicted feature (for both pitch accents and phrase boundary models on BDC,

	best fit		dur w_{prev}	dur w_{cur}	dur w_{next}
	R^2	feature	R^2	R^2	R^2
pitch accents					
BURNC	0.52	max voicing w_{cur}	0.13	0.42	0.08
using pauses	0.52	max voicing w_{cur}	0.13	**0.52**	0.11
BDC	0.52	min voicing w_{cur}	0.06	0.31	0.13
using pauses	0.67	dur w_{cur}	0.05	**0.67**	0.13
DIRNDL	0.48	mean htnr w_{cur}	0.14	0.21	0.16
using pauses	0.63	dur w_{cur}	0.11	**0.63**	0.10
phrase boundaries					
BURNC	0.38	max energy w_{next}	0.10	0.34	0.21
using pauses	0.49	mean htnr w_{cur}	0.08	**0.41**	0.30
BDC	0.40	max energy w_{next}	0.21	0.27	0.30
using pauses	0.65	dur w_{cur}	0.15	**0.65**	0.19
DIRNDL	0.46	min energy w_{next}	0.23	0.07	0.23
using pauses	0.68	mean energy w_{next}	0.08	**0.10**	0.09

Table 5.5: Results of linear models trained on CNN output features to predict manual features. The table lists the features that yielded the best fit as well as the results for predicting duration.

and pitch accents on DIRNDL). As in the above analysis, we assume this was because the word boundaries were more accurate in this setting. As expected, the duration of the two context words could not be modeled well (indicated by the lower numbers in the columns for w_{prev} and w_{next}), however, the duration of w_{next} was predicted better for the phrase boundary models than for pitch accent models. It appears that not only the following acoustic context, but also the duration of the following word is especially relevant for phrase boundaries. We investigate the role of context information further in Section 5.4.

Figure 5.5 shows the results for the second analysis, in which we used the manual features to predict the CNN features. The y-axis refers to the maximum percentage of the 100 CNN features that are correlated with a single aggregated feature. We considered only the acoustic feature categories in the visualization, which means that we do not show the aggregation type (e.g. *max, min, mean, stddev*) and word position (i.e. *prev, current, next*) of the respective "best feature". This means that, for example, if the feature *min energy* w_{cur} was a significant predictor for 60 out of the 100 CNN features, and this was the maximum rate for all energy features, then this is the maximum percentage of CNN features that are correlated with a single aggregated feature of this category. The *deltas* category refers to any of the delta features.

The longer bars in the diagrams for pitch accents are caused by the fact that the linear models showed an overall higher correlation between the CNN features and the aggregated features. This is because the CNN pitch accent models had a higher performance than the phrase boundary models, which we attribute to the better balance of positive and negative instances in the data. The stronger pitch accent models therefore have learned more useful acoustic information.

The duration features are among the most frequent significant predictors for the CNN features, as shown by the longer bars in both diagrams. The best duration feature is w_{cur} (not shown); it is correlated with over 90% of the CNN output features in the pitch accent models, and slightly less frequently for the phrase boundary models. The latter finding may be attributed to the less robust phrase boundary models.

In terms of acoustic features, we frequently found voicing probability, energy and harmonics-to-noise-ratio to be significant for pitch accents, while energy and loudness were strongest for phrase boundaries. Especially on BURNC, the deltas feature category displays a high correlation for pitch accents. What is not shown in the figures is that this mostly pertains to the voicing probability features. These observed differences between the pitch accent and phrase boundary models is in line with what would be expected of them to have learned: For detecting pitch accents, relevant voiced regions need to be identified, since pitch accents require the presence of voicing. Furthermore, a drop in energy and loudness marks phrase boundaries and co-occurring silences.

For the settings in which pause transcriptions are used, the numbers for duration and delta features increased (except for the DIRNDL phrase boundary results). We interpret this as an effect of the more precise word boundaries that result from separating pauses in the input matrix. Furthermore, many acoustic features appear less frequently when pauses are used for the pitch accent detection models. This may be explained by the fact that delta and duration features were more informative in this setting, or by the fact that the acoustic information was not required for separating speech from non-speech segments. However, we cannot verify this assumption from these results alone.

What is notable is that the importance of many features depends on the corpus. For example, voicing probability and delta features are strongest in the pitch accent models trained on BURNC. However, since we only used one dataset per domain (radio news vs. spontaneous speech and English vs. German), we cannot ascertain whether these differences are corpus-, domain- or language-specific.

Two additional observations are not shown in this figure: First, although F0 was not significant in the majority of linear models, almost all best-fitting models did include

F0 features as significant predictors. This suggests that F0 is still be an informative feature despite the fact that it is one of the weakest. The finding that the prosodic event detector performs best when using the full feature set supports this assumption.

Second, the most frequent delta features for phrase boundaries were those computed as the difference between w_{cur} and w_{next}. This is the corresponding observation to that made in the first analysis, indicating that the CNN has learned to place more focus on the end of the current word and the beginning of the next, which is the region where intonational phrase boundaries occur.

5.3.3 Summary

In this section, we analyzed the acoustic information learned by the CNN-based prosodic event detector for pitch accents and phrase boundaries. We used a two-way linear regression analysis to determine how well a set of candidate features correlated with the latent information in the CNN output. While these results show many corpus-specific differences, i.e. speaking style (read vs. spontaneous) and language (English vs. German), we do observe some patterns: Voicing probability and harmonics-to-noise-ratio could be approximated as independent variables in the first analysis, and appeared quite frequently as significant predictors in the second analysis. We assume that these features may be used to find relevant voiced regions in the speech signal.

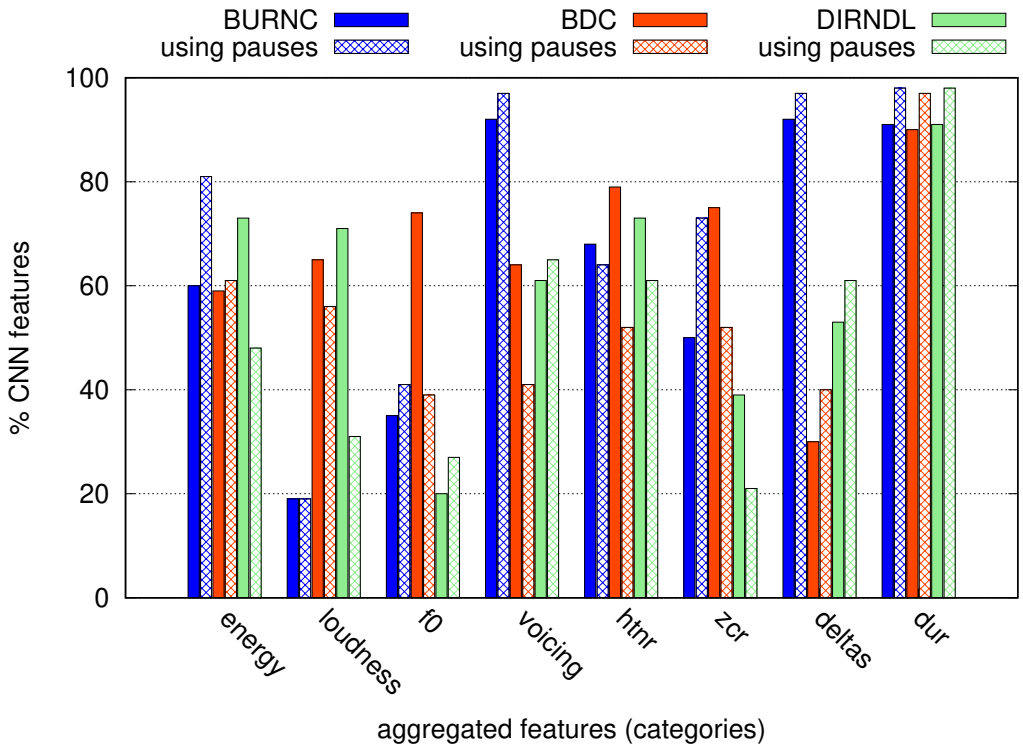
Our analysis shows some large differences between the three tested corpora. Especially Figure 5.5 shows that the importance of individual features is corpus-specific to a certain extent, which explains why models trained on one corpus do not necessarily generalize well to another. To what extent this is genre- or language-specific or how this relates to the lexical content of the corpora would be an interesting question to address in future work.

Some results on the DIRNDL corpus could not be explained: Our analysis showed that the CNN output of the DIRNDL phrase boundary models did not appear to encode duration. Similarly, duration features were not useful for improving phrase boundary detection on this dataset using the FFNN. We assume that these findings are connected, however further investigation is necessary to ascertain the reason for these results.

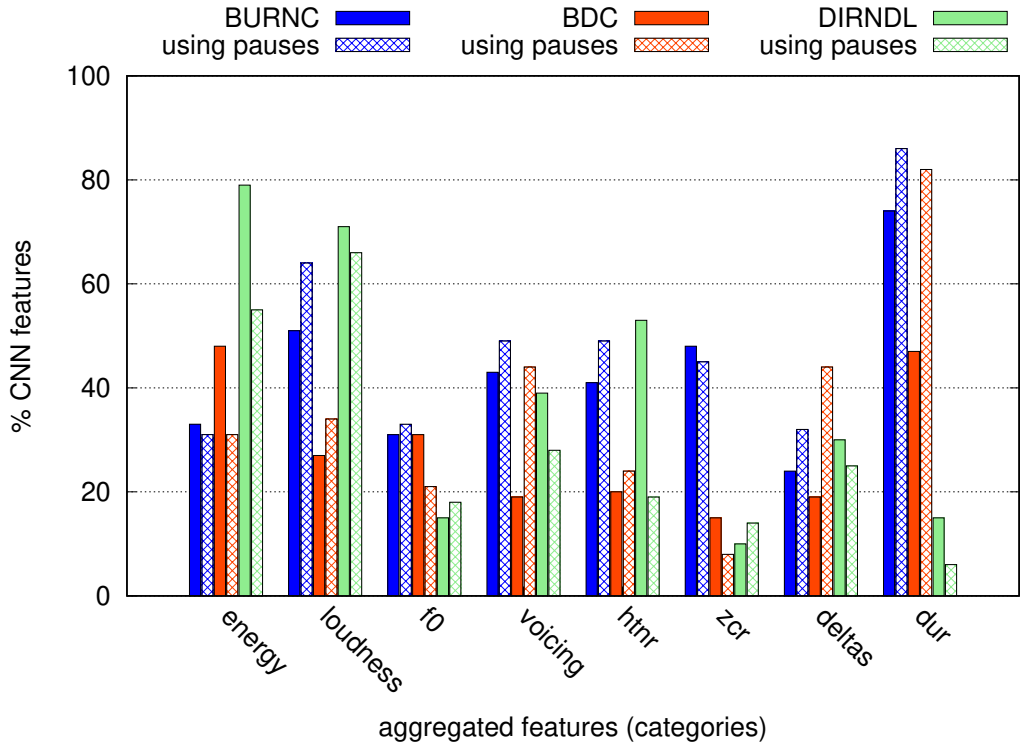
Some of the findings in this section reflect the observations made in Section 3: Energy and loudness were found to be strong features across corpora, especially for phrase boundaries. Finally, despite the fact that F0 is one of the primary correlates of prosody, it is one of the weakest features in the prosodic event detector. Not only is this finding is in line with previous research, but it is an observation that has not previously been

made in neural-network-based prosodic models.

The CNN appears to have learned most information for the current word, in line with our findings in Section 5.2. In the next section, we continue our analysis with the question of how much context information the model has learned.



(a) Linear model results for pitch accent detection



(b) Linear model results for phrase boundary detection

Figure 5.5: Best aggregated features in the linear models trained to predict single CNN features. The bars shows the maximum frequency with which an aggregated feature in the listed feature categories was found to be a significant predictor in a set of 100 linear models, pertaining to the 100 CNN features.

5.4 The role of context information in convolutional and sequential models

In previous sections, we discussed the role of context information in combination with the position indicators in terms of performance (Section 3.4) and showed how this is connected to the method of pooling and padding (Section 5.2). We found that the method of padding the input matrix and then performing 1-max pooling makes it necessary to add position indicators that mark which frames pertain to the context or current word, and that therefore, the CNN output encodes primarily the duration of the current word. The latter finding also holds when using 3-max pooling without position indicators if the input words are each padded separately and thus have fixed positions in the input matrix. This, along with the observations made in Section 5.3, shows that the CNN learns most information for the current word.

In this section, we extend our investigation to how much acoustic context information is learned when using the default method of 1-max pooling with position indicators. In Section 3.9 we compared the CNN to an LSTM that operates on the same input matrix with position indicators, and found that both models yield comparable performance. We do assume, however, that there may be differences in how both models learn context information, since the LSTM is designed to model sequences, while the CNN looks at more local patterns. In the following experiments, we compared how duration and the included acoustic context information is learned in both models.

We also analyzed the effect of this is learned when different input information is provided, namely with or without context frames and the position indicators. All experiments were conducted on the BURNC dataset with 5-fold cross-validation, since we assume that the investigated effects are not corpus-specific. We used the CNN model with the same settings as in the previous section and treated pauses as separate words, since this was found to produce better results on the BDC corpus while not harming the results on the other two corpora.

5.4.1 Forward and backward recurrent networks

In addition to the standard *forward* LSTM used in Section 3.9, we also implemented a *backward* LSTM, that is, one that processes the input sequence in reverse order. Since recurrent models are known to retain the most information for the most recent time steps, we placed the zero padding so that it was seen first: For the forward LSTM, the

zero padding was added to the front of the input matrix, and for the backward LSTM, the padding was added to the end.

All other settings were identical to the settings in Section 3.9, except for the number of epochs: We observed that especially the LSTM models for phrase boundary detection benefitted from longer training times. Therefore, we trained both the LSTM and CNN models for 30 epochs in the experiments that focus on evaluating the performance in Section 5.4.2. The subsequent analyses were conducted on the models trained for Section 3.9, since the overall trends were not affected by the performance differences that resulted from longer training times.

The motivation for analyzing forward and backward LSTMs separately was to show that both models display a different way of modeling context information, which we discuss in sections 5.4.3 and 5.4.4. For the sake of completeness, we also tested the performance of a bidirectional LSTM for pitch accent and phrase boundary detection using the default method of adding context and position indicators. Since the bidirectional model contains twice as many parameters as the unidirectional models, we trained it for 50 epochs. This network type yields an F1-score of 89.1 for pitch accent detection and 83.4 for phrase boundary detection. Since it does not outperform the unidirectional models (*bw*-LSTM for pitch accents and *fw*-LSTM for phrase boundaries, see Table 5.6) and at the same time requires more training time, we conclude that more complex models may not be necessary for the size of datasets used in our experiments.

5.4.2 Effect of context information and position indicators

We first compare the performance of all 3 network types when using context information with and without the position indicators. Table 5.6 shows the F1 score of both the CNN and the forward and backward LSTMs for pitch accent and phrase boundary detection. The results show the effect of adding context information with and without the position indicators (*context* and *context + posind*) on the detection performance.

Without context information, all models perform quite well (around 85% for pitch accent detection), but when adding "raw" context information, the results are quite poor. As we have previously shown for the CNN (Section 3.4), these results confirm that the position indicators are also necessary for the LSTM since both methods operate on the same frame-based input. It appears that even though the issue of pooling does not apply for the LSTM, it still has difficulty learning what frames pertain to the current word to be labeled, which is a result of the padding method described in Section 5.2.

An interesting result is that the backward LSTM does not perform well with context

on phrase boundary detection, even with the position indicators. We assume that this is because important cues to phrase boundaries are found in the context after the current word, which was also shown in the analysis in Section 5.3. Since this variant processes the input in reverse order, it is more likely to "forget" this information. The forward LSTM detects prosodic events slightly better than the CNN with the drawback of required more training time. In sum, both methods are roughly comparable in terms of performance, but the CNN is easier and faster to train.

These results once again highlight the importance of using balanced training data for training robust models, an issue discussed in Section 3.4. Less than 20% of the words are positive instances of the phrase boundary class, and therefore, the F1 score is lower than for pitch accents, where the classes are balanced. What this table does not show is that phrase boundary detection has a higher precision, while pitch accent detection has a higher recall. This is an effect that both CNN and LSTM have in common, and implies that, for this task, the data can have a larger impact on the performance than the actual network architecture.

	CNN	*fw*-LSTM	*bw*-LSTM
pitch accents			
no context	85.4	85.8	86.3
context	62.6	73.3	72.4
context + posind	87.1	88.9	89.1
phrase boundaries			
no context	74.8	80.7	78.7
context	51.8	68.0	55.2
context + posind	80.8	83.8	79.7

Table 5.6: F1 score for prosodic event detection using the CNN and the forward (fw) and backward (bw) LSTM with different context information (on BURNC). The models were trained for 30 epochs.

5.4.3 Analysis of learned word duration

In the next analysis we investigated how much information on the duration of the input words is latent in the output representation of the three network types. As in Section 5.2, we trained linear models to predict the duration of each of the three input words (w_{prev}, w_{cur}, w_{next}) using the NN features from the prosodic event detection models tested in this section: We compared the CNN to both the forward and backward LSTMs, trained

	w_{prev}	w_{cur}	w_{next}
CNN			
no context	*0.05*	**0.64**	*0.02*
context	*0.07*	0.17	0.12
context + posind	0.15	**0.53**	0.12
forward LSTM			
no context	*0.05*	**0.78**	*0.04*
context	*0.07*	0.25	0.18
context + posind	*0.06*	**0.66**	**0.37**
backward LSTM			
no context	*0.05*	**0.75**	*0.03*
context	0.20	0.24	0.14
context + posind	**0.41**	**0.65**	*0.04*

Table 5.7: Comparison R^2 for predicting word duration using NN output features for pitch accent detection. Numbers shown in italics are too low to reflect approximation, numbers shown in bold reflect a moderate to strong approximation.

for both pitch accent and phrase boundary detection and using the different context settings tested in Section 5.4.2. The results for the pitch accent detection models (R^2 of predicting word duration) are shown in Table 5.7 and the phrase boundary results are given in Table 5.8.

All network types appear to have learned the duration of the current word w_{cur} when no context is added to the input, as shown by the good model fit (>0.6 R^2). This can be explained by the fact that the network does not need to distinguish parts of the matrix that represent w_{cur} and is therefore better at estimating its duration. In contrast, when context is added without the position indicators, then the fit is not good enough to conclude that the model has succeeded in learning the word duration. This effect is the same across all three neural network types and shows that the position indicators are crucial for the learning of duration. Even without the position indicators, however, it should be noted that the R^2 is still higher for the current word than for the context words, which indicates that this information is at least weakly encoded in the learned features.

The difference between the CNN and the two LSTMs is in how well the context word duration was learned: The CNN features appear to primarily encode the duration of the current word, as the results for the context words is still rather low. The LSTM features, however, appear to additionally include the duration of the word that occurred

	w_{prev}	w_{cur}	w_{next}
CNN			
no context	*0.02*	**0.52**	*0.05*
context	*0.08*	0.15	0.10
context + posind	0.12	**0.42**	0.31
forward LSTM			
no context	*0.04*	**0.71**	*0.07*
context	*0.06*	0.18	0.12
context + posind	*0.01*	**0.58**	**0.52**
backward LSTM			
no context	*0.03*	**0.72**	*0.04*
context	*0.06*	*0.02*	*0.002*
context + posind	**0.45**	**0.42**	*0.09*

Table 5.8: Comparison R^2 for predicting word duration using NN output features for phrase boundary detection. Numbers shown in italics are too low to reflect approximation, numbers shown in bold reflect a moderate to strong approximation.

last in the input sequence. Therefore, the duration of the w_{next} is learned well by the forward LSTM, and the duration of the w_{prev} by the backward LSTM. An interesting observation is that this also applies for the backward LSTM trained for phrase boundary detection, although the performance of the LSTM was found to be poor (Table 5.6).

We measured the correlation (Spearman's ρ) between the duration of each input word and the target prosodic event label (Table 5.9) to assess the importance for prosodic event prediction. There is a clear correlation between the current word duration and the target label, but not for the duration of the context words (except a slight negative correlation with the next word for phrase boundaries). Combined with the detection performance in Table 5.6, we can conclude that the current word duration is very important for the accurate detection of pitch accents and phrase boundaries, and the position indicators help with this in both the CNN and the LSTM.

	w_{prev}	w_{cur}	w_{next}
pitch accents	-0.15	0.64	-0.13
phrase boundaries	0.07	0.49	-0.28

Table 5.9: Spearman correlation between word duration and the target prosodic event label.

5.4.4 Analysis of acoustic context information

The role of context information on word duration was made clear in the previous analysis as well as in Section 5.2. We now investigate how much of the acoustic context is learned in the CNN and the LSTM. As in our analysis in Section 5.3, we computed the 72 aggregated acoustic features (without F0 and delta features) as described in Section 3.8.2 across each word in the input. We predicted each using the linear regression models trained on the CNN and LSTM features for each context setting (with and without context and position indicators). Since we were interested only in what acoustic feature categories were learned, we averaged the R^2 values across the aggregation types. That is, we obtained an average value for each acoustic descriptor over the minimum, maximum, mean and standard deviation values. The results of this analysis, which we show only for the pitch accent models[1], are visualized in Figures 5.6 to 5.7.

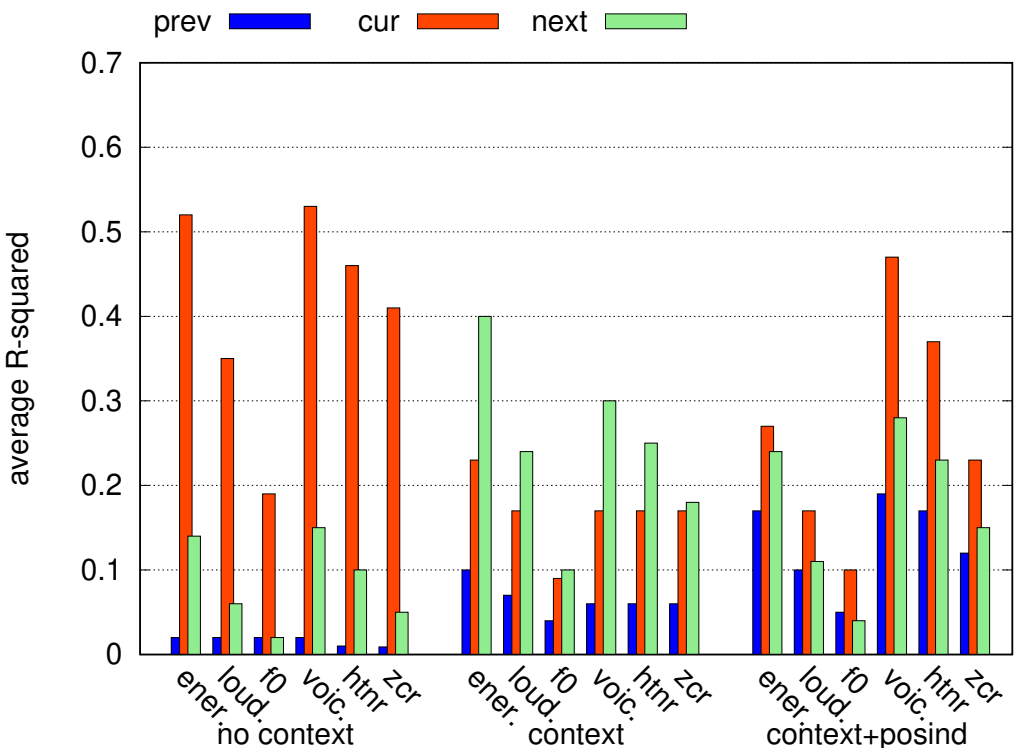


Figure 5.6: Results for predicting acoustic features using linear models trained on CNN output features (obtained for pitch accent detection). The y-axis represents the average R^2 over the min./max./mean/std.-dev. of each acoustic feature category and each input word (*prev, cur, next*).

[1]We chose not to include the results for the phrase boundary detection since the general trends are similar. We include phrase boundaries in the analysis in Section 5.4.5

Figure 5.6 shows that, for the CNN, most acoustic information was learned from the current word in the *no context* and *context+posind* settings. Without the position indicators (*context*), a lot of the acoustic information is learned from w_{next}. An interesting observation is that information pertaining to the next word appears have been modeled better than information pertaining to the previous word. This may be explained by the fact that the zero padding was added to the end of the input matrix: since the kernels are shifted all the way to the end of the matrix with a degree of overlap, information at the end of the context window is captured "more often" than at the beginning.

Figures 5.7a and 5.7b show the results for the forward and backward LSTM, respectively. Similar to the results for predicting word duration, these results illustrate how the direction of the LSTM influences where most of the learned acoustic information is derived from, namely w_{next} for the forward LSTM and w_{prev} for the backward LSTM. The forward LSTM learns hardly any information pertaining to the previous word, while the backward LSTM learns very little pertaining to the next word.

What is not visible here is the fact that in the *no context* and *context+posind* settings, the current word duration is the best predicted feature of all tested manual features. This goes for both the CNN and the LSTM, although the best acoustic features learned by the LSTM correspond to the most recently seen context word. It appears that while the position indicators are important for duration learning, they do not direct the LSTM towards the current word as strongly as in the case of the CNN.

As for the individual acoustic features, the findings for the LSTM are similar to the CNN results discussed in sections 3.6 and 5.3 in that energy and voicing probability appear to have been learned best, while F0 does not appear to be an important feature. In fact, the relative importance of the different acoustic feature categories are identical in all three network types (that is, the relative length of the bars display the same pattern in all three figures). This shows that the CNN and LSTMs modeled the same acoustic information for the same task and data, and that the primary difference is in how these architectures process and model the input sequence.

In Section 5.2, we had mentioned that while the role of duration appears to be quite large, it is not likely that the position indicators make the model “ignore” the context information. These results, however, show that this is not the case: the network does learn some context information regardless of whether the position indicators are added or not. Despite the fact that the 1-max pooling method “forces” the model to choose one neuron for each feature map, the information it encodes does not pertain only to the current word. This shows that the context is necessary for modeling prosodic events,

which is in line with previous findings as well as the theoretic linguistic background reviewed in Section 2.

5.4.5 Comparison of predicted features

In Section 5.3, we observed that the duration features appeared to be more important than the acoustic features, which was especially visible in Figure 5.5. Although the above analysis has shown that the models learn acoustic context, the duration information still may outweigh it. Therefore, we listed the top 5 manual features that were predicted with the highest R^2 in the linear model analysis in Tables 5.10 for pitch accent detection and 5.11 for phrase boundary detection in all 3 context settings tested above. The results clearly show the importance of the position indicators for learning the current word duration. Table 5.10 also shows that more learned information pertains to the current word duration if the position indicators are included or if no context is added at all. The bottom part of the table reveals the main difference in the effect that the position indicators have on the CNN and the two LSTMs: In the CNN, all top features represent the current word, while in the LSTM, this holds only for the current word duration. The acoustic features are learned mainly for the most recently seen word. For the phrase boundary models, the overall effects are similar, but the results do display some notable differences: Duration is not listed in the top 5 features when using context with the position indicators. It is still predicted fairly well, for example, R^2 for the current duration in the CNN is 0.47. Furthermore, while the CNN models most acoustic information for the current word when the position indicators are added to the pitch accent models, the best features in the phrase boundary models pertain to the next word. This is an interesting result that indicates that the CNN focuses more on the following context to model phrase boundaries. The LSTM does not appear to have this flexibility and instead learns most acoustic information from the most recently seen word.

It should be noted that the individual R^2 scores and the ranking of the manual acoustic features are not reproducible, since the exact representations vary randomly to a certain extent. We are only able to show general trends, which is why we had averaged over several acoustic feature scores in the previous analyses.

From these results the conclusion that duration appears to outweigh the acoustic information in most cases still holds. This can be explained by the fact that duration is highly correlated feature, while the manual acoustic features computed via aggregation over the whole word are merely approximations. We discuss this further in Section 5.5.

CNN			LSTM-fw			LSTM-bw		
R^2	word	feature	R^2	word	feature	R^2	word	feature
no context								
0.64	cur	dur	0.78	cur	energy-max	0.75	cur	dur
0.59	cur	voice-min	0.78	cur	dur	0.72	cur	htnr-mn
0.58	cur	voice-max	0.77	cur	htnr-mn	0.66	cur	energy-max
0.57	cur	htnr-mn	0.76	cur	loud-mn	0.65	cur	energy-mn
0.57	cur	energy-mn	0.65	cur	voice-std	0.65	cur	loud-mn
context								
0.45	next	energy-mn	0.63	next	energy-max	0.54	prev	energy-min
0.43	next	energy-max	0.62	next	loud-mn	0.53	prev	energy-mn
0.43	next	energy-min	0.58	next	energy-mn	0.52	prev	energy-max
0.43	next	loudn-mn	0.55	next	energy-min	0.50	prev	loud-mn
0.34	next	htnr-mn	0.49	next	energy-std	0.41	prev	energy-std
context + posind								
0.53	cur	dur	0.66	cur	dur	0.65	cur	dur
0.53	cur	voice-max	0.63	next	energy-max	0.60	prev	energy-mn
0.51	cur	htnr-mn	0.62	next	loud-mn	0.60	prev	energy-min
0.51	cur	voice-min	0.59	next	energy-mn	0.58	prev	htnr-mn
0.44	cur	voice-mn	0.55	next	energy-min	0.57	prev	energy-max

Table 5.10: Top 5 best predicted manual features (R^2) depending on included context information in the CNN and forward and backward LSTMs for pitch accent detection. The 4 aggregation types are minimum (min), maximum (max), mean (mn) and standard deviation (std).

5.4.6 Summary

We compared the output representations learned by the CNN-based prosodic event detector to those learned by an LSTM. The two network types were found to yield similar performance in Section 3.9 although they processes the frame-based input in a different manner. The learned acoustic features and word duration, as well as the beneficial effect of the position indicator, were found to be the same in all three models. A likely explanation for this observation is that they take the same frame-based input. The duration of the current word was learned best when the position indicators were provided, and the most important acoustic features are energy and voicing probability, while F0 was not found to be learned well. The main difference between these network types is in how much context information is encoded in the output feature representation: While the position indicators ensure that both models learn a significant amount of acoustic and duration information for the current word, most of the information in the LSTM output

CNN			LSTM-fw			LSTM-bw		
R^2	word	feature	R^2	word	feature	R^2	word	feature
no context								
0.52	cur	dur	0.71	cur	dur	0.72	cur	dur
0.49	cur	energy-mx	0.64	cur	energy-max	0.61	cur	loud-mn
0.46	cur	loud-max	0.63	cur	loud-mn	0.59	cur	energy-max
0.42	cur	energy-std	0.55	cur	energy-mn	0.51	cur	energy-mn
0.36	cur	energy-mn	0.53	cur	energy-min	0.48	cur	energy-min
context								
0.47	next	energy-max	0.64	next	energy-max	0.59	prev	loud-mn
0.45	next	energy-mn	0.63	next	loud-mn	0.58	prev	energy-max
0.45	next	loud-mn	0.57	next	energy-mn	0.57	prev	energy-mn
0.42	next	energy-min	0.53	next	energy-min	0.58	prev	energy-min
0.39	next	htnr-mn	0.52	next	energy-std	0.53	prev	energy-std
context + posind								
0.52	next	energy-mn	0.68	next	energy-max	0.59	prev	energy-max
0.52	next	voice-min	0.66	next	loud-mn	0.58	prev	energy-mn
0.51	next	energy-mn	0.66	next	htnr-mn	0.58	prev	energy-min
0.50	next	voice-mn	0.64	next	energy-mn	0.57	prev	loud-mn
0.50	next	htnr-mn	0.63	next	energy-min	0.53	prev	htnr-mn

Table 5.11: Top 5 best predicted features (R^2) depending on included context information in the NN and forward and backward LSTMs for phrase boundary detection

pertains to the most recently seen word in the input matrix. An interesting effect of this direction-based learning in the LSTM was the fact that a backwards-processing model does not perform well on phrase boundary detection. The results also confirmed that although the position indicators enable the CNN to focus more on the current word, they do not prevent the model from learning relevant context information.

5.5 Discussion

The method of analyzing the output of a neural network using linear regression provides a rough estimation about what information is latent in the CNN output features. It can be used to test hypotheses by measuring correlations between manually defined candidate features and the learned output. For example, the fact that most information pertains to the current word can be considered a straightforward finding, since this is the word to be labeled.

The fact that F0 was not consistently among the most important features (that is, weakly encoded in the learned features representation but not as strong as e.g. energy and voicing probability) reflects the previous findings discussed in the literature (Batliner et al., 2001a), namely that while F0 is an important correlate of prosody (especially prosodic event types, i.e. tonal shapes), it is often found to be a weak feature in automatic prosodic event detection. In Section 3.6, we had observed that while F0 was one of the strongest features when used individually, the performance did not suffer when it was left out of the whole feature set. Taken together, these observations suggest that F0 is simply not as important as other correlates for the task of detecting the location of pitch accents. Whether or not F0 would appear as one of the top features in this type of analysis when the models were trained for pitch accent type classification instead is a question that requires further investigation.

Although this analysis supported several of our previous assumptions, the best fit of the linear models did not exceed 0.6 R^2. We believe that this can be explained by the fact that the CNN learns a significant amount of information from the acoustic input features that is not expressed by the manually aggregated features (i.e. simple mean, minimum, maximum and standard deviation of the acoustic input features across each word.) The candidate features chosen for this analysis were simple aggregations that each describe individual acoustic descriptors over a single word. This is also why the CNN or even LSTM which learns from frame-based features achieves a better performance than the FFNN from Section 3.8.2.

Even the more fine-grained F0 features that we computed for the analysis in Section 5.3 were not predicted well by the linear models. Although this is likely caused by the issue mentioned above that F0 is generally not a strong feature for automatic prosodic event detection, it also shows that it is difficult to manually define what the neural network could have learned. For this reason, the caveat of using this type of analysis is that it is a top-down approach, in that one first must specify information to look for in the learned feature representation. We do not believe that this means that it is an unsuitable method of analysis. The fact that many results were in line with our expectations also supports our position that this is a suitable and readily implemented method of analyzing what a neural network has learned.

In future work, the linear regression analysis could be extended test more complex features. The choice of candidate features to predict could be guided by knowledge about the network architecture as well as statistical properties of the data. One possibility would be to compute features that take interactions between the 6 acoustic descriptors

into account, since the CNN kernels span the entire feature set in the first layer and thus learn all features simultaneously. Rosenberg et al. (2015) give an example of how a simple interaction features between pitch and energy can be computed by multiplying the values.

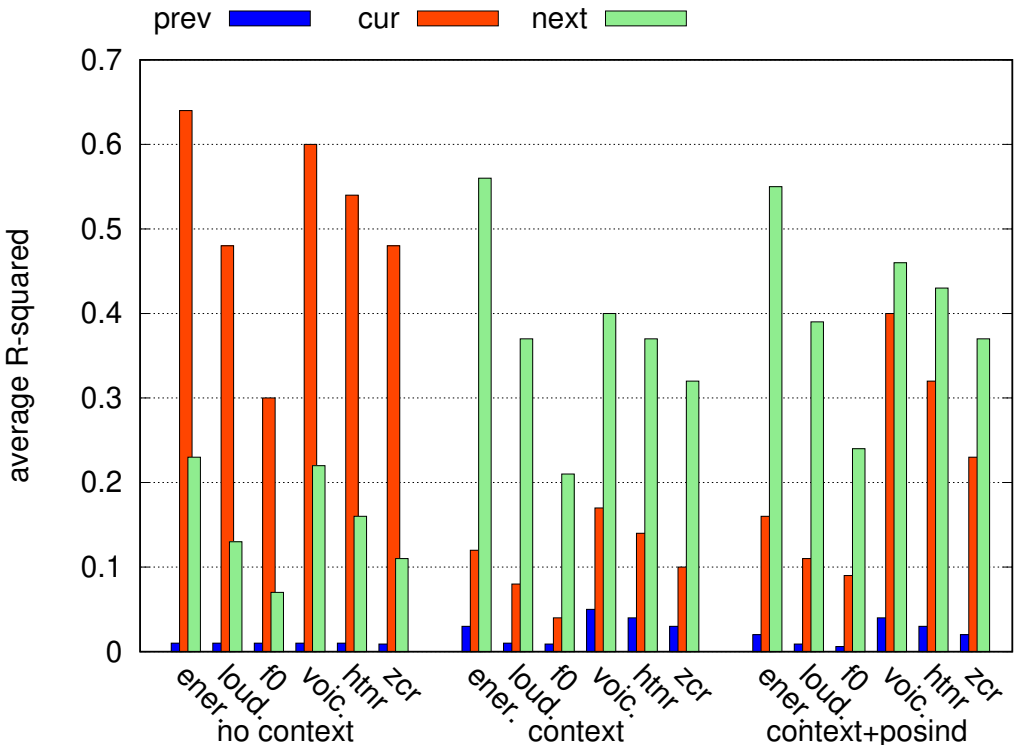
5.6 Conclusion

Our goal in this section was to gain an understanding of what the neural network is learning. We addressed the question of what information is latent in the learned feature representation using a simple method: By training linear regression models on either the CNN output to predict a set of candidate aggregated features and vice versa, we were able to investigate how well the aggregated features are correlated with the information encoded in the CNN output. We first showed that the CNN can learn word duration on its own from the frame-based features and position indicators. Our analysis on the acoustic information in the CNN revealed that while there are differences across corpora, certain acoustic features, such as energy and voicing probability, are more important for detecting pitch accents and phrase boundaries, while other features, such as F0, are not as important. This is in line with previous research on prosodic event detection. So far, this is the first study to confirm that these observations are in accordance with what information is latent in feature representations learned by neural network models of prosodic events.

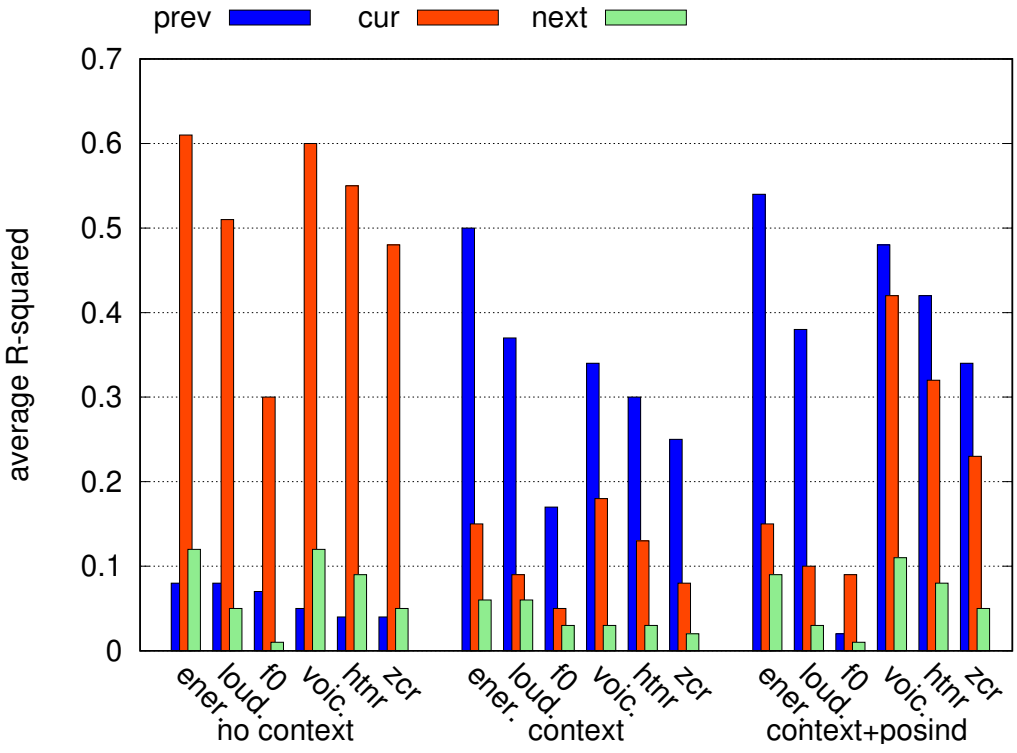
We then compared how the CNN learns context information and also performed the same analysis on a forward and a backward sequential model (LSTM) for comparison. The largest difference between the two network types is the amount of learned context information. We found that the LSTM captures more context information than the CNN. In the LSTM, this also depends on the direction in which the input sequence is processed: information that is most recently seen is learned best. Both networks appear to learn the duration of the current word, which is facilitated by the addition of the position indicators.

Although this analysis can only provide an estimation of what type of information is latent in the high-level feature representations, it is a task-agnostic and readily implemented method of interpreting what neural networks have learned. We believe that we have only scratched the surface of explaining what type of information the model could be learning. How exactly the CNN models pitch accents and phrase boundaries, and what the high-level features consist of exactly, remains a topic for future exploration.

In this section, we have described an important first step to our understanding of how convolutional neural networks model prosodic events.



(a) Forward LSTM



(b) Backward LSTM

Figure 5.7: Averages of R^2 for predicting acoustic features using a linear model trained on forward and backward LSTM output features (for pitch accent detection).

6 Conclusion

The work in this thesis was motivated by the fact that the prosody of languages such as English and German contributes to the interpretation and disambiguation of speech, but that exploiting this connection for speech technology is generally regarded as challenging. As of yet, no standard method of extracting and including prosodic information in automatic speech understanding exists, despite a significant amount of previous research indicating that it is a useful knowledge source in addition to text-based information. Standard automatic speech recognition (ASR) systems do not recognize prosody by default. Therefore, in a speech understanding pipeline, any prosodic information present in the speech signal that can help solve downstream tasks is lost. The work described in this thesis aims to overcome this challenge in three main parts: First, by presenting an efficient method of prosodic event detection that can be readily inserted into a pipeline; second, by demonstrating how the model can be analyzed and third, by investigating potential applications. Our approach to prosodic event detection was to use state-of-the-art neural network models, specifically convolutional neural networks (CNNs). We presented three different applications of automatic prosodic event detection: to provide features for slot filling and coreference resolution and for silver-standard corpus annotation. Since the question of explainability is currently an important issue in neural network research, we placed our final focus on analyzing what the network has learned.

The following summary describes are summarized the most important findings pertaining to each of the three goals in this thesis: modeling, applications and analysis.

6.1 Key Findings

We showed that **neural networks are an efficient method of performing automatic prosodic event detection**. The proposed method of detecting pitch accents and intonational phrase boundaries at the word level uses a small set of frame-based acoustic descriptors that represent a 3-word context window, and a position indicator feature. The performance rivals comparable methods on the Boston Radio News Corpus,

reaching 86.8% pitch accent detection accuracy for 10-fold cross-validation and 84.3% for leave-one-speaker-out cross-validation. The accuracy for phrase boundary detection, obtained using the same evaluation, is 92.5% (79.3% F1-score) and 91.9%, respectively. Furthermore, the model generalizes well from English to German data. Adding lexical information in the form of word embeddings can, however, harm the generalization performance, since these features are more corpus-specific. Temporal information such as pauses and word duration do not have a notable effect on the results. We considered this finding as evidence that the CNN can learn this information on its own; and the results of our analysis provided further support for this assumption.

Despite the fact that neural networks often rely on large amounts of training data, the CNN in our work does not require more hand-annotated data than related methods. This method can potentially be used on raw speech if the word boundary information is obtained using ASR. We conclude that it can be readily integrated into a pipeline setting, does not require much effort to be adapted and used as a pre-trained model on new datasets. This is an important prerequisite for the second goal, which was to test several applications.

We showed that **pre-trained prosodic event detectors, implemented as separate modules in speech understanding pipelines, can help downstream tasks**. The first set of experiments was motivated by the connection between pitch accents and salient or focused words. We found that the benchmark dataset for slot filling, the American English ATIS corpus, displays a high correlation between pitch accent words and **semantic slots**. This finding holds even when both the transcriptions and the prosodic annotations are obtained automatically, suggesting that a system can exploit this correlation despite recognition errors. In state-of-the-art slot filling systems, the lexical information is modeled too strongly for the model to benefit from the knowledge of pitch accent locations. When applied to automatically recognized text, however, this information may potentially help overcome the performance drop by pointing to key information in the text.

The next experiment was a result of collaborative work on the use of prosodic event labels for **coreference resolution**. Coreferent noun phrases tend to get deacccented in speech, since they comprise given information. Here, we showed that including features based on the presence of pitch accents and intonational phrase boundaries, predicted using the CNN-based method, significantly improves the performance of coreference resolution on German data. This is an important finding that indicates that manually annotated prosodic labels are not required for this task, although the automatic labels

were of lower quality since the predictor was pre-trained on a different corpus.

Since gold-standard prosodic annotations are expensive to create and tend to show a low inter-annotator agreement, the CNN-based prosodic event detector was tested for its use in creating **silver-standard annotations**. The CNN-based detector was pre-trained on a German dataset and applied to the GRAIN corpus (Schweitzer et al., 2018) to obtain word-level binary labels of prosodic events. Compared to a more linguistically-informed method that provided a finer-grained labeling of event types on syllables, our method yields a lower precision, but a higher recall. Taken together, the confidence of automatic prosodic annotations is increased.

Our final contribution was the analysis of the CNN-based model. We presented a method of **analyzing the learned output representation of the neural network**, and showed that it contains latent representations of important correlates. This method consists of a linear regression analysis between the features learned by the neural network and a set of manually computed features, in which one set of features was used to predict individual features in the other set and vice versa. This type of analysis has not been previously used for prosodic modeling.

We were able to confirm our assumption that the CNN-based prosodic event detector learns information on the word duration, although the input to the model consists only of frame-based acoustic features and therefore does not explicitly include this information. The position indicator feature was found to play a key role in how the model learns word duration as well as the surrounding context information. The acoustic features that the neural network output was found to encode suggest that the CNN can learn to localize voiced regions in speech that are relevant for detecting prosodic events. We also observed that energy is an important feature for prosodic event detection, whereas F0 was only weakly encoded. This is in line with findings previously reported in the literature on the importance of individual features for this task.

A recurrent neural network (RNN) that takes the same frame-based input as the CNN was found to yield similar performance levels and to learn similar acoustic and temporal features. The results of our analysis on the learned features from the RNN shows that the two models differ with respect to the amount of context information they contain: Specifically, the direction in which the RNN processes the input information affects what input words are modeled best. Both the CNN and the RNN learn most information pertaining to the current word, thereby making use of the position indicator features.

6.2 Outlook

For this work, we only assumed one type of experimental setup, namely a speech understanding pipeline that involves a separate module to predict symbolic prosodic events. As mentioned in Section 1, there are alternative approaches that this thesis does not explore, such as the use of abstract prosodic representations or more direct modeling approaches. While we do not argue for or against either method, the advantage of assuming a pipeline setup is the flexibility of using separate modules individually. For example, the PaIntE-based prosodic event detector used for the slot filling methods in section 4.1 could be replaced with the CNN-based method. In future work, both coud be systematically combined: The simpler and faster CNN-based method can be used to first perform a coarse-grained detection on words before a more fine-grained analysis is applied, which would increase the efficiency and precision of prosodic labeling. Rosenberg's AuToBI tool (Rosenberg, 2010a) follows a similar idea.

Our experiments, along with the work of Rösiger and Riester (2015), showed that binary word-level prosodic events already suffice to improve automatic coreference resolution. Based on the work by Baumann and Riester (2013), we can assume that even more fine-graind information such as pitch accent types may be even more useful cues to meaning such as information status. However, since the model was optimized for word-level prosodic event detection on radio news data, the prosodic event detector did not achieve the same performance on tasks other than word-level detection and across datasets. The results on the Boston Directions Corpus were comparably low, as were the results for syllable-level tasks when the model was applied "out-of-the-box". Furthermore, the classification of different ToBI types remains challenging. We believe that further optimization effort along with larger training datasets can help adapt neural-network-based prosodic event detection for these purposes.

One advantage of using neural networks is that they can be used in all suggested setups, not only as shown in this thesis but also for direct modeling or for creating abstract representations. To address the question of whether symbolic events are more suitable than abstract representations, a possible future direction would be to use the CNN-based model to create prosodic representations that are based on the neural network output or extracted from hidden layers. Furthermore, the modular nature of neural network architectures could be exploited to create more complex models. For example, given enough annotated data, the context information taken in to account may be increased by pairing the CNN with a sequential model in a hybrid architecture or by using a

sequential CNN similar to the slot-filling model used in section 4.1.

Since one aim of this work was to develop a prosodic event detector that requires very little preprocessing, the question arises as to whether an even more low-level representation of the speech signal would be desirable as input. The "end-to-end" approach of modeling speech processing tasks often aims to find patterns in raw speech data, without the need for further intermediate signal processing steps which may cause errors or information loss. However, this often requires deeper network architectures and therefore larger annotated training corpora, which are generally sparse and expensive to create. Furthermore, we found that extracting acoustic descriptors from the speech signal is unproblematic in terms of speed and cost.

The neural network analysis was restricted to the output feature representation, which means that we did not investigate the hidden layers, relevant input regions or other mechanisms that may help explain the model. The advantage of the method used in this thesis is that it is task-agnostic and can be employed to investigate further questions: Since lexico-syntactic information provides helpful cues for detecting prosodic events, the method based on linear regression may be extended to explore the lexical information included in the model, similar to the embeddings analysis by Adi et al. (2017). For example, the question of what semantic or syntactic information is latent in the output feature representation or how the acoustic and lexical cues interact in the model could be investigated. We expect that this method can also be used to analyze the hidden layers.

In this thesis, we focused only on the prosody of English and German. The CNN-based could also be readily extended for application on other languages. For example, the automatic prediction of ToBI labels has been explored for Spanish (Elvira-García et al., 2016), and Levow (2005) used a single method for predicting both pitch accents in English and tones in Mandarin Chinese.

The range of applications of the CNN-based prosodic event detector is not limited to the examples investigated in this thesis. We focused on the use of prosodic event detection to predict features for slot filling, coreference resolution and to provide corpus annotations. The prosodic information that is useful for slot filling and coreference resolution mainly comprises the presence of pitch accents and their role in the marking of salient information and information status. We expect that the CNN-based method can also be helpful for other downstream tasks that are based on other linguistic roles of prosody. For example, syntactic parsing or sentence segmentation can benefit from information on phrase boundaries.

As technological advancements in the field of speech and language processing enable spoken language applications to become increasingly mainstream, one of the main goals is to create natural-sounding speech and to imitate human verbal behavior in a realistic manner. Therefore, research that not only **connects spoken and written language**, but also takes **prosody** into account, is expected to become more relevant. The success of state-of-the-art applications is based on deep neural networks which are, in turn, powerful models that display remarkable learning capabilities given sufficient data, and with minimal linguistic knowledge. While these developments also have the potential to drive application-based research and linguistic research further apart, recent studies have argued that both fields can still benefit from each other (Linzen, 2018). Especially for research on prosody, which faces the challenge of multiple perceptual cues, low inter-annotator agreement and a relative lack of consensus on certain aspects (Wightman, 2002; Taylor, 2009, pp. 111–112,139–142), this notion is promising. We therefore believe that work that also **bridges the gap between application-based research and linguistics** deserves more attention. This thesis has considered neural-network-based modeling of prosody primarily from the perspective of the former. In the future, we hope that the scope of this work can be extended to support and help connect both fields.

Bibliography

Adi, Y., Kermany, E., Belinkov, Y., and Goldberg, Y. (2017). Analysis of sentence embedding models using prediction tasks in natural language processing. *IBM Journal of Research and Development*, 61(4/5):3–1.

Amoia, M., Kunz, K., and Lapshinova-Koltunski, E. (2012). Coreference in spoken vs. written texts: a corpus-based analysis. In *Proceedings of the 8th Conference on Language Resources and Evaluation*, pages 158–164.

Ananthakrishnan, S. and Narayanan, S. (2006). Combining acoustic, lexical and syntactic evidence for automatic unsupervised prosody labeling. In *Proceedings of Interspeech*, pages 297–300.

Ananthakrishnan, S. and Narayanan, S. (2007). Improved speech recognition using acoustic and lexical correlates of pitch accent in a n-best rescoring framework. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 873–876.

Ananthakrishnan, S. and Narayanan, S. S. (2008). Automatic prosodic event detection using acoustic, lexical and syntactic evidence. *IEEE Transactions on Audio, Speech and Language Processing*, 16(1):216–228.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):e0130140.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *ICLR 2015, arXiv:1409.0473*.

Batliner, A., Buckow, J., Huber, R., Warnke, V., Nöth, E., and Niemann, H. (2001a). Boiling down prosody for the classification of boundaries and accents in german and english. In *Proceedings of Eurospeech*, pages 2781–2784.

Batliner, A., Möbius, B., Möhler, G., Schweitzer, A., and Nöth, E. (2001b). Prosodic models, automatic speech understanding, and speech synthesis: towards the common ground. In *Proceedings of the European Conference on Speech Communication and Technology (Aalborg, Denmark)*, pages 2285–2288.

Batliner, A., Nöth, E., Buckow, J., Huber, R., Warnke, V., and Niemann, H. (2001c). Duration features in prosodic classification: Why normalization comes second, and what they really encode. In *Proceedings of the ISCA Tutorial and Research Workshop on Speech Recognition and Understanding*, pages 23–28.

Baumann, S. and Riester, A. (2013). Coreference, lexical givenness and prosody in German. *Lingua*, 136:16–37.

Becker, S., Ackermann, M., Lapuschkin, S., Müller, K.-R., and Samek, W. (2018). Interpreting and explaining deep neural networks from classification of audio signals. *arXiv preprint: arXiv:1807.03418*.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. In *Proceedings of NIPS*, pages 932–938.

Björkelund, A., Eckart, K., Riester, A., Schauffler, N., and Schweitzer, K. (2014). The extended DIRNDL corpus as a resource for automatic coreference and bridging resolution. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 3222–3228.

Black, A. W., Taylor, P., and Daley, P. (2010). The festival text-to-speech synthesis system. Centre for speech technology research, University of Edinburgh.

Boersma, P. (2001). Praat, a system for doing phonetics by computer. 5:9/10:341–345.

Chen, K., Hasegawa-Johnson, M., and Cohen, A. (2004). An automatic prosody labeling system using ann-based syntactic-prosodic model and gmm-based acoustic-prosodic model. In *Proceedings of ICASSP*, pages 509–512.

Cho, E., Niehues, J., Kilgour, K., and Waibel, A. (2015). Punctuation insertion for real-time spoken language tranlsation. In *Proceedings of the 11th International Workshop on Spoken Language Translation*.

Chollet, F. et al. (2015). Keras. `https://keras.io`.

Clark, J. and Yallop, C. (1995). *Introduction to Phonetics and Phonology*. Blackwell Publishers.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2461–2505.

Dieleman, S., Schlüter, J., Raffel, C., Olson, E., Sønderby, S. K., Nouri, D., et al. (2015). Lasagne: First release.

Domínguez, M., Farrús, M., and Wanner, L. (2016). Combining acoustic and linguistic features in phrase-oriented prosody prediction. In *Proceedings of Speech Prosody*, pages 796–800.

Ebert, S., Vu, N. T., and Schütze, H. (2015). A linguistically informed convolutional neural network. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 109–114.

Eckart, K., Riester, A., and Schweitzer, K. (2012). A discourse information radio news database for linguistic analysis. In Chiarcos, C., Nordhoff, S., and Hellmann, S., editors, *Linked Data in Linguistics. Representing and Connecting Language Data and Language Metadata*, pages 65–75. Springer, Heidelberg.

Elman, J. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Elvira-García, W., Roseano, P., Fernández-Planas, A. M., and Martínez-Geldrán, E. (2016). A tool for automatic transcription of intonation: Eti_ToBI, a ToBI transcriber for spanish and catalan. *Language Resources and Evaluation*, 50(4):767–792.

Ettinger, A., Elgohary, A., and Philip, R. (2016). Probing for semantic evidence of composition by means of simple classification tasks. In *Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NLP*, pages 134–139.

Eyben, F., Weninger, F., Groß, F., and Schuller, B. (2013). Recent developments in opensmile, the Munich open-source multimedia feature extractor. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 835–838.

Fant, G. (1960). *Acoustic Theory of Speech Production*. Walter De Gruyter.

Fernandez, R., Rosenberg, A., Sorin, A., Ramabhadran, B., and Hoory, R. (2017). Voice-transformation-based data augmentation for prosodic classification. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5530–5534.

Ferrer, L., Shriberg, E., and Stolcke, A. (2003). A prosody-based approach to end-of-utterance detection that does not require speech recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 608–611.

Féry, C. (1993). *The meaning of German intonational patters*. Max Niemeyer Verlag Tübingen.

Féry, C. (2017). *Intonation and Prosodic Structure*. Key Topics in Phonology. Cambridge University Press.

Godfrey, J. J., Holliman, E. C., and McDaniel, J. (1992). Switchboard: telephone speech corpus for research and development. In *[Proceedings of the 1992 IEEE International*

Conference on Acoustics, Speech, and Signal Processing, volume 1, pages 517–520.

Goldberg, Y. (2017). *Neural Network Methods for Natural Language Processing*. Morgan & Claypool publishers.

Goldwater, S., Jurafsky, D., and Manning, C. D. (2010). Which words are hard to recognize? prosodic, lexical and disfluency factors that increase speech recognition error rates. *Speech Communication*, 52:181–200.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Grice, M., Baumann, S., and Benzmüller, R. (2005). German intonation in autosegmental-metrical phonology. In Jun, S.-A., editor, *Prosodic Typology: The Phonology of Intonation and Phrasing*. Oxford University Press.

Haas, J., Kießling, A., Nöth, E., Niemann, H., and Batliner, A. (1995). Contrastive accents. how to get them and what do they look like? In *Proceedings of the International Conference on Phonetic Sciences*, pages 4:276–4:279.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18.

Hasegawa-Johnson, M., Chen, K., Cole, J., Borys, S., Kim, S.-S., Cohen, A., Zhang, T., Choi, J.-Y., Kim, H., Yoon, T., and Chavarria, S. (2005). Simultaneous recognition of words and prosody in the boston university radio speech corpus. *Speech Communication*, 46:418–439.

He, Y. and Young, S. (2003). A data-driven spoken language understanding system. In *Proceedings of the IEEE ASRU Workshop*, pages 583–588.

Hemphill, C. T., Godfrey, J. J., and Doddington, G. R. (1990). The ATIS Spoken Language Systems Pilot Corpus. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 96–101. Morgan Kaufmann.

Hess, W., Batliner, A., Kießling, A., Komple, R., Nöth, E., Petzold, A., Reyelt, M., and Strom, V. (1996). Prosodic modules for speech recognition and understanding in VERBMOBIL. Technical report, Universitäten Bonn, Braunschweig, Erlangen-Nürnberg, München.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Jaitly, A. M. N., Senior, A., Nguyen, V. V. P., Sainath, T. N., and Kingsbury, B. (2012). Deep neural networks for acoustic modelling in speech recognition: the shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hirschberg, J. (1993). Pitch accent in context: Predicting intonational prominence from

text. *Artificial Intelligence*, 63:305–340.

Hirschberg, J. (2002). Communication and prosody: Functional aspects of prosody. *Speech Communication*, 36:31–43.

Hirschberg, J., Litman, D., and Swerts, M. (2004). Prosodic and other cues to speech recognition failures. *Speech Communication*, 43:155–175.

Hirschberg, J. and Nakatani, C. (1998). Acoustic indicators of topic segmentation. In *Proceeding of the 5th International Conference on Spoken Language Processing*. 0976.

Hirschberg, J. and Nakatani, C. H. (1996). A prosodic analysis of discourse segments in direction-giving monologues. In *Proceedings of ACL*, pages 286–293.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Huang, J.-T., Hasegawa-Johnson, M., and Shih, C. (2008). Unsupervised prosodic break detection in mandarin speech. In *Proceedings of the 4th International Conference on Speech Prosody*, pages 165–168.

Jeon, J. H. and Liu, Y. (2009). Automatic prosodic events detection using syllable-based acoustic and syntactic features. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4565–4568.

Jeon, J. H. and Liu, Y. (2012). Automatic prosodic event detection using a novel labeling and selection method in co-training. *Speech Communication*, 54:445–458.

Jeon, J. H., Wang, W., and Liu, Y. (2011). N-best rescoring based on pitch accent patterns. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 732–741.

Kahn, J. G., Lease, M., Charniak, E., Johnson, M., and Ostendorf, M. (2005). Effective use of prosody in parsing conversational speech. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 233–240.

Kakouros, S., Suni, A., Simko, J., and Vainio, M. (2019). Prosodic representations of prominence classification neural networks and autoencoders using bottleneck features. In *Proceedings of Interspeech*, pages 1946–1950.

Katerenchuck, D. and Rosenberg, A. (2014). Improving named entity recognition with prosodic features. In *Proceedings of Interspeech*, pages 293–297.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. *ICLR 2015, arXiv:1412.6980*.

Klatt, D. H. (1976). Linguistic uses of segmental duration in english: acoustic and perceptual evidence. *Journal of the Acoustical Society of America*, 59:1208–1221.

Kompe, R. (1997). *Prosody in Speech Understanding Systems*. Lecture Notes in Artificial Intelligence. Springer-Verlag New York, Inc.

Krug, A. and Stober, S. (2018). Introspection for convolutional automatic speech recognition. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 187–199.

Kumar, V., Sridhar, R., Bangalore, S., and Narayanan, S. S. (2008). Exploiting acoustic and syntactic features for automatic prosody labeling in a maximum entropy framework. *IEEE Transactions on Audio, Speech & Language Processing*, 16(4):797–811.

Ladd, D. R. (1996). *Intonational phonology*. Cambridge Studies in Linguistics. Cambridge University Press.

Levow, G.-A. (2005). Context in multi-lingual tone and pitch accent recognition. In *Proceedings of Interspeech*, pages 1809–1812.

Levow, G.-A. (2006). Unsupervised learning of tone and pitch accent. In *Proceedings of the 3rd International Conference on Speech Prosody*. 224.

Levow, G.-A. (2009). Investigating pitch accent recognition in non-native speech. In *Proceedings of ACL-IJCNLP*, pages 269–272.

Li, K., Mao, S., Li, X., Wu, Z., and Meng, H. (2018). Automatic lexical stress and pitch accent detection for l2 english speech using multi-distribution deep neural networks. *Speech Communication*, 96:28–36.

Linzen, T. (2018). What can linguistics and deep learning contribute to each other? response to pater. *Language*, 95(1):e98–e108.

Linzen, T., Chrupała, G., Belinkov, Y., and Hupkes, D., editors (2019). *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Florence, Italy. Association for Computational Linguistics.

Lipton, Z. C. (2016). The mythos of model interpretability. In *ICML Workshop on Human Interpretability in Machine Learning*, pages 96–100.

Mairesse, F., Polifroni, J., and Fabbrizio, G. D. (2012). Can prosody inform sentiment analysis? experiments on short spoken reviews. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5093–5096.

Margolis, A., Ostendorf, M., and Livescu, K. (2010). Cross-genre training for automatic prosody classification. In *Proceedings of Speech Prosody*. 113.

Mayer, J. (1995). Transcription of German intonation. The Stuttgart system.

McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., and Sonderegger, M. (2017). Montreal forced aligner: trainable text-speech alignment using Kaldi. In *Proceedings of Interspeech*, pages 498–502.

Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., and Zweig, G. (2015). Using recurrent neural networks for slot filling in spoken language understanding. *IEEE Transactions on Audio, Speech and Language Processing*, 23:530–539.

Mesnil, G., He, X., Deng, L., and Bengio, Y. (2013). Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of Interspeech*, pages 3771–3775.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In *ICLR Workshop, arXiv:1301.3781*.

Mikolov, T., Yih, W., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *HLT-NAACL*.

Milde, J.-T. and Gut, U. (2002). A prosodic corpus of non-native speech. In *Proceedings of Speech Prosody*, pages 503–506.

Möhler, G. and Conkie, A. (1998). Parametric modeling of intonation using vector quantization. In *Third ESCA/COCOSDA Workshop on Speech Synthesis (SSW3)*, pages 311–316.

Möhler, G. and Conkie, A. (1998). Parametric modeling of intonation using vector quantization. In *Proceedings of the 3rd ESCA Workshop on Speech Synthesis*, pages 311–316.

Montavon, G., Samek, W., and Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15.

Morency, L.-P., Mihalcea, R., and Doshi, P. (2011). Towards multimodal sentiment analysis: Harvesting opinions from the web. In *Proceedings of the 13th International Conference on Multimodal Interfaces*, pages 169–176.

Nagamine, T., Seltzer, M., and Mesgarani, N. (2016). On the role of nonlinear transformations in deep neural network acoustic models. In *Proceedings of Interspeech*, pages 803–807.

Nenkova, A., Brenier, J., Kothari, A., Calhoun, S., Whitton, L., Beaver, D., and Jurafsky, D. (2007). To memorize or to predict: Prominence labeling in conversational speech. In *in Proceedings of NAACL-HLT*, pages 9–16.

Nicol, J. L. (1996). What can prosody tell a parser? *Journal of Psycholinguistic Research*, 25(2):179–192.

Nöth, E. (1991). *Prosodische Information in der automatischen Spracherkennung*. Niemeyer Tübingen.

Nöth, E., Batliner, A., Warnke, V., Haas, J., Boros, M., Buckow, J., Huber, R., Gallwitz,

F., Nutt, M., and Niemann, H. (2002). On the use of prosody in automatic dialogue understanding. *Speech Communication*, 36:45–62.

Ostendorf, M., Price, P., and Shattuck-Hufnagel, S. (1995). The Boston University Radio News Corpus. Technical Report ECS-95-001, Boston University.

Ostendorf, M., Shafram, I., and Bates, R. (2003). Prosody models for conversational speech recognition. In *Proceedings of the 2nd Plenary Meeting and Symposium on Prosody and Speech Processing*, pages 147–154.

Pan, S. and McKeown, K. R. (1999). Word informativeness and automatic pitch accent modeling. In *In Proceedings of EMNLP/VLC*, pages 148–157.

Paul, D. B. and Baker, J. M. (1992). The design for the Wall Street Journal-based CSR corpus. In *Proceedings of the DARPA Speech and Natural Language Workshop*.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Peters, J. (2014). *Intonation*. Universitätsverlag Winter Heidelberg.

Pierrehumbert, J. (1980). *The Phonology and Phonetics of English Intonation*. PhD thesis, Massachusetts Institute of Technology, Dept. of Linguistics and Philosophy.

Pierrehumbert, J. B. and Hirschberg, J. (1990). The meaning of intonational contours in the interpretation of discourse. In P. Cohen, J. Morgan, M. P., editor, *Intentions in Communications*, pages 271–311. MIT Press.

Pitrelli, J., Beckman, M., and Hirschberg, J. (1994). Evaluation of prosodic transcription labeling reliability in the ToBI framework. In *Proceedings of the 3rd Workshop on Spoken Language Processing*, pages 123–126.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. (2011). The Kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*.

Price, P., Ostendorf, M., Shattuck-Hufnagel, S., and Fong, C. (1991). The use of prosody in syntactic disambiguation. In *Proceedings of the Workshop on Speech and Natural Language*, pages 372–377.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rapp, S. (1995). Automatic phonemic transcription and linguistic annotation from known text with hidden markov models – an aligner for german. In *Proceedings of ELSNET Goes East and IMACS Workshop "Integration of Language and Speech in*

Academia and Industry, Moscow, Russia.

Ren, K., Kim, S.-S., Hasegawa-Johnson, M., and Cole, J. (2004). Speaker-independent automatic detection of pitch accent. In *ISCA International Conference on Speech Prosody*, pages 521–524.

Rendel, A., Fernandez, R., Hoory, R., and Ramabhadran, B. (2016). Using continuous lexical embeddings to improve symbolic-prosody prediction in a text-to-speech front-end. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5655–5659.

Riester, A. and Piontek, J. (2015). Anarchy in the NP. when new nouns get deaccented and given nouns don't. *Lingua*, 165(B):230–253.

Rosenberg, A. (2009). *Automatic Detection and Classification of Prosodic Events*. PhD thesis, Columbia University.

Rosenberg, A. (2010a). Autobi - a tool for automatic ToBI annotation. In *Proceedings of Interspeech*, pages 146–149.

Rosenberg, A. (2010b). Classification of prosodic events using quantized contour modeling. In *Proceedings of NAACL-HLT*, pages 721–724.

Rosenberg, A., Cooper, E., Levitan, R., and Hirschberg, J. (2012). Cross-language prominence detection. In *Proceedings of Speech Prosody*, pages 278–281.

Rosenberg, A., Fernandez, R., and Ramabhadran, B. (2015). Modeling phrasing and prominence using deep recurrent learning. In *Proceedings of Interspeech*, pages 3066–3070.

Rosenberg, A. and Hirschberg, J. (2007). Detecting pitch accent using pitch-corrected energy-based predictors. In *Proceedings of Interspeech*, pages 2777–2780.

Rosenberg, A. and Hirschberg, J. (2009). Detecting pitch accents at the word, syllable and vowel level. In *Proceedings of NAACL-HLT*, pages 81–84.

Rösiger, I. and Kuhn, J. (2016). IMS HotCoref DE: a data-driven co-reference resolver for German. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*, pages 155–160.

Rösiger, I. and Riester, A. (2015). Using prosodic annotations to improve coreference resolution of spoken text. In *Proceedings of ACL-IJCNLP*, pages 83–88.

Rösiger, I., Stehwien, S., Riester, A., and Vu, N. T. (2017). Improving coreference resolution with automatically predicted prosodic information. In *Proceedings of the 1st Workshop on Speech-centric natural language processing*, pages 78–83.

Santos, C. N. D., Xiang, B., and Zhou, B. (2015). Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the*

Association for Computational Linguistics, pages 626–634.

Schiel, F. (1999). Automatic phonetic transcription of non-prompted speech. In *Proceedings of the International Conference on Phonetic Sciences (ICPhS)*, pages 607–610.

Schweitzer, A. (2010). *Production and Perception of Prosodic Events-Evidence from Corpus-based Experiments*. PhD thesis, Universität Stuttgart.

Schweitzer, A. and Möbius, B. (2009). Experiments on automatic prosodic labeling. In *Proceedings of Interspeech*, pages 2515–2518.

Schweitzer, K., Eckart, K., Gärtner, M., Falenska, A., Riester, A., Rsiger, I., Schweitzer, A., Stehwien, S., and Kuhn, J. (2018). German Radio Interviews: the GRAIN release of the SFB732 Silver Standard Collection. In *Proceedings of the 11th Workshop on Language Resources and Evaluation*, pages 2887–2895.

Schweitzer, K., Walsh, M., Möbius, B., and Schütze, H. (2010). Frequency of occurrence effects on pitch accent realisation. In *Proceedings of Interspeech*, pages 138–141.

Selkirk, E. (1995). Sentence prosody: Intonation, stress and phrasing. In *The handbook of phonological theory 1*, pages 550–569.

Shafran, I., Ostendorf, M., and Wright, R. (2001). Prosody and phonetic variability: Lessons learned from acoustic model clustering. In *Proceedings of the ISCA Workshop on Prosody in Speech Recognition and Understanding*, pages 127–131.

Shahin, M., Epps, J., and Ahmed, B. (2016). Automatic classification of lexical stress in English and Arabic languages using deep learning. In *Proceedings of Interspeech*, pages 175–179.

Shattuck-Hufnagel, S. and Turk, A. E. (1996). A prosody tutorial for investigators of auditory sentence processing. *Journal of Psycholinguistic Research*, 25(2):193–247.

Shi, X., Padhi, I., and Kevin, K. (2016). Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirial Methods in Natural Language Processing*, pages 1526–1534.

Shriberg, E. (2005). Spontaneous speech: How people really talk and why engineers should care. In *Proceedings of the 9th European Conference on Speech Communication and Technology*, pages 1781–1784.

Shriberg, E., Bates, R., and Stolcke, A. (1997). A prosody-only decision-tree model for disfluency detection. In *Eurospeech*, pages 2382–2386.

Shriberg, E. and Stolcke, A. (2004). Prosody modeling for automatic speech recognition and understanding. In *Mathematical Foundations of Speech and Language Processing*, pages 105–114. Springer.

Shriberg, E., Stolcke, A., Hakkani-Tür, D., and Tür, G. (2000). Prosody-based automatic

segmentation of speech into sentence and topics. *Speech Communication*, 32:127–154.

Siegel, S. and Castellan, N. J. J. (1988). *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, Berkeley, CA, 2nd edition.

Silverman, K., Beckman, M., Pitrelli, J., Ostendorf, M., Wightman, C., Price, P., Pierrehumbert, J. B., and Hirschberg, J. (1992). ToBI: A standard for labelling English prosody. In *Proceedings of the 2nd International Conference on Spoken Language Processing*, pages 867–870.

Sluijter, A. and van Heuven, V. (1996). Acoustic correlates of linguistic stress and accent in dutch and american english. In *Proceedings of ICSLP 1996*, pages 385–388, Philadelphia, USA.

Soto, V., Cooper, E., Rosenberg, A., and Hirschberg, J. (2013). Cross-language phrase boundary detection. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8460–8464.

Sridhar, V. K. R., Nenkova, A., Narayanan, S., and Jurafsky, D. (2008). Detecting prominence in conversational speech: pitch accent, givenness and focus. In *Proceedings of Speech Prosody*, pages 453–456.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Stehwien, S., Schweitzer, A., and Vu, N. T. (2019). Convolutional neural networks can learn duration for detecting pitch accents and lexical stress. In *Proceedings of the 30th Conference on Electronic Speech Signal Processing (ESSV)*, pages 17–24.

Stehwien, S., Schweitzer, A., and Vu, N. T. (in press). Acoustic and temporal representations in convolutional neural network models of prosodic events. *Speech Communication*, submitted 2019, accepted 2020, tentatively published 2021.

Stehwien, S. and Vu, N. T. (2016). Exploring the correlation of pitch accents and semantic slots for spoken language understanding. In *Proceedings of Interspeech*, pages 730–735.

Stehwien, S. and Vu, N. T. (2017a). First step towards enhancing word embeddings with pitch accent features for DNN-based slot filling on recognized text. In *Proceedings of the 28th Conference on Electronic Speech Signal Processing (ESSV)*, pages 194–201.

Stehwien, S. and Vu, N. T. (2017b). Prosodic event recognition using convolutional neural networks with context information. In *Proceedings of Interspeech*, pages 2326–2330.

Stehwien, S., Vu, N. T., and Schweitzer, A. (2018). Effects of word embeddings on

neural-network-based pitch accent detection. In *Proceedings of Speech Prosody*, pages 719–723.

Stolcke, A., Shriberg, E., Bates, R., Ostendorf, M., Hakkani, D., Plauche, M., Tür, G., and Lu, Y. (1998). Automatic detection of sentence boundaries and disfluencies based on recognized words. In *Proceeding of the 5th International Conference on Spoken Language Processing*. 0059.

Strom, V. (1995). Detection of accents, phrase boundaries and sentence modality in german with prosodic features. In *Proceedings of EUROSPEECH*, volume 3, pages 2039–2041.

Strom, V. and Widera, C. (1996). What's in the "pure" prosody? In *Proceedings of the 4th International Conference on Spoken Language Processing*, pages 1497–1500.

Strube, M. and Müller, C. (2003). A machine learning approach to pronoun resolution in spoken dialogue. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 168–175.

Sun, X. (2002). Pitch accent prediction using ensemble machine learning. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 953–956.

Tamburini, F. (2003). Prosodic prominence detection in speech. In *ISSPA2003*.

Taylor, P. (1995). Using neural networks to locate pitch accents. In *Proceedings of the 4th European Conference on Speech Communication and Technology*, pages 1245–1348.

Taylor, P. (2009). *Text-to-speech Synthesis*. Cambridge University Press.

Tepperman, J. and Narayanan, S. (2005). Automatic syllable stress detection using prosodic features for pronunciation evaluation of language learners. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 937–940.

Terken, J. and Hirschberg, J. (1994). Deaccentuation of words representing 'given' information: Effects of persistence of grammatical function and surface position. *Language and Speech*, 37(2):125–145.

Tetreault, J. and Allen, J. (2004). Dialogue structure and pronoun resolution. In *Proceedings of the 5th Discourse Anaphora and Anaphor Resolution Colloquium*.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint: arXiv:1605.02688*.

Tran, T., Toshniwal, S., Bansal, M., Gimpel, K., Livescu, K., and Ostendorf, M. (2018). Parsing speech: A neural approach to integrating lexical and acoustic-prosodic information. In *Proceedings of NAACL-HLT*, pages 69–81.

Tur, G., Hakkani-Tur, D., and Heck, L. (2010). What's left to be understood in ATIS? In *Proceedings of the IEEE Spoken Language Technology Workshop*, pages 19–24.

Umbach, C. (2002). (De)accenting definite descriptions. *Theoretical Linguistics*, 2/3:251–280.

Vassière, J. (1988). The use of prosodic parameters in automatic speech recognition. In *Recent advances in speech understanding and dialog systems*, pages 71–99.

Veilleux, N. and Ostendorf, M. (1993). Prosody/parse scoring and its application in ATIS. In *In Proceedings of the ARPA Workshop on Human Language Technology*, pages 335–340.

Veilleux, N., Shattuck-Hufnagel, S., , and Brugos, A. (2006). 6.911 Transcribing prosodic structure of spoken utterances with ToBI. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA.

Vicsi, K. and Szaszák, G. (2010). Using prosody to improve automatic speech recognition. *Speech Communication*, 52:413–426.

Vu, N. T. (2016). Sequential convolutional neural networks for slot filling in spoken language understanding. In *Proceedings of Interspeech*, pages 3250–3254.

Vu, N. T., Adel, H., Gupta, P., and Schütze, H. (2016a). Combining recurrent and convolutional neural networks for relation classification. In *Proceedings of NAACL-HLT*, pages 534–539.

Vu, N. T., Gupta, P., Adel, H., and Schütze, H. (2016b). Bi-directional recurrent neural network with ranking loss for spoken language understanding. In *Proceedings of the 41st IEEE International Conference on Acoustics, Speech and Signal Processing*.

Wahlster, W., editor (2000). *Verbmobil: Foundations of speech to speech translation*. Springer.

Waibel, A. (1988). *Prosody and Speech Recognition*. Morgan Kaufmann.

Wang, P., Qian, Y., Soong, F., He, L., and Zhao, H. (2015). Word embeddings for recurrent neural network based TTS synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4879–4883.

Wang, S., Qian, Y., and Yu, K. (2017). What does the speaker embedding encode? In *Proceedings of Interspeech*, pages 1497–1501.

Wang, X., Takaki, S., and Yamagishi, J. (2016). Enhance the word vector with prosodic information for the recurrent neural network based TTS system. In *Proceedings of Interspeech*, pages 2856–2860.

Wightman, C. and Ostendorf, M. (1994). Automatic labeling of prosodic patterns. *IEEE Transactions on Speech and Audio Processing*, 2(4):469–481.

Wightman, C. W. (2002). ToBI or not ToBI? In *Proceedings of Speech Prosody*, pages 25–29.

Wightman, C. W., Shattuck-Hufnagel, S., Ostendorf, M., and Price, P. J. (1992). Segmental durations in the vicinity of prosodic phrase boundaries. *Journal of the Acoustical Society of America*, 91(3):1707–1717.

Xu, P. and Sarikaya, R. (2013). Convolutional neural network based triangular CRF for joint intent detection and slot filling. In *IEEE Workshop for Automatic Speech Recognition and Understanding*, pages 78–83.

Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G., and Shi, Y. (2014). Spoken language understanding using long short-term memory neural networks. In *IEEE Spoken Language Technology Workshop*, pages 189–194.

Yao, K., Zweig, G., Hwang, M., Shi, Y., and Yu, D. (2013). Recurrent neural networks for language understanding. In *Proceedings of Interspeech*, pages 2524–2528.

Yuan, J., Brenier, J. M., and Jurafsky, D. (2005). Pitch accent prediction: Effects of genre and speaker. In *Proceedings of Interspeech*, pages 1409–1412.

Zhang, D. and Wang, D. (2015). Relation classification via recurrent neural network. *arXiv preprint arXiv:508.01006v1*.

Zhang, Q. and Zhu, S.-C. (2018). Visual interpretability for deep learning: A survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39.