Prediction and Incorporation of Phrase Break Models in Text to Speech Systems

Thesis submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Engineering

by

Anandaswarup Vadapalli
201050048
anandaswarup.vadapalli@research.iiit.ac.in



International Institute of Information Technology
Hyderabad - 500 032, INDIA
March 2025

Copyright © Anandaswarup Vadapalli, 2025
All Rights Reserved

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Prediction and Incorporation of Phrase Break Models in Text to Speech Systems" by Anandaswarup Vadapalli, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. Kishore Prahallad

Acknowledgments

I would like to thank my advisor Dr. Kishore Prahallad for his guidance and encouragement throughout my PhD. I would also like to thank all the members of the Speech Processing Lab at IIIT Hyderabad (faculty, staff and students) both past and present for all their support and the fun times that we had.

I would also like to thank the faculty and administration of IIIT Hyderabad for creating world class infrastructure for research and for all the support provided to research students.

Thanks also to all my colleagues at MSIT, IIIT Hyderabad for all their support and the fun times. Thanks to all the MSIT students who participated in the perceptual listening tests.

Last but not the least, I would like to thank my parents (Late) Prof. V Guruprasada Rau and Mrs. Lakshmi Rau, my in-laws Mr. Narayana Swamy Vuppuluri and Mrs. Vijayalakshmi Vuppuluri, my wife Manasa and my darling daughters Drishti and Darshana for all their love and support throughout.

Abstract

Spoken utterances have an inherent structure in the sense that some words seem to group naturally together and some words seem to have a notable break or disjuncture between them. This can be described in terms of prosodic phrasing, meaning that a spoken utterance has a prosodic phrase structure, similar to how a written utterance has a syntactic phrase structure.

Phrase breaks in natural speech are important; they are physiologically essential, help emphasize content, and improve the intelligibility of speech. The process of inserting phrase breaks in an utterance is called phrasing. In the context of speech synthesis, phrasing is a crucial step. It breaks long utterances into meaningful units of information and improves the intelligibilty of the synthesized speech.

Traditional methods of phrase break prediction have used discrete linguistic resources like part-of-speech (POS) sequence information for modeling these breaks. These methods cannot be used for languages where the necessary linguistic resources are not readily available, resulting in unsupervised methods of inducing word representations which can be used as surrogates for POS tags in phrase break prediction. However, these methods are not suitable in the context of Indian languages, which are aggutinative in nature resulting in an increase in vocabulary size. This thesis presents the use of word terminal syllables (last syllable of the word) to model phrase breaks. We demonstrate the correlation between these terminal syllables and acoustic breaks found in the speech signal. We utilize these terminal syllables in building models for phrase break prediction from text in six Indian languages and demonstrate by means of objective and subjective measures that these models perform as well as traditional models using POS sequence information.

The discrete linguistic representations (like POS tags, induced POS tags, word-terminal syllables) used by traditional methods of phrase break prediction require a hard classification of words into a set of predefined discrete classes, which raises issues when there is ambiguity in the linguistic representation of a word. Moreover, such a representation does not capture the co-occurrence statistics of the words, i. e., they do not take into account the distributional behaviour of words. Both these issues can be addressed by the use of continuous dimensional representations of words also known as word embeddings. This thesis presents a neural network architecture to induce task specific word embeddings for phrase break prediction. We show that our proposed architecture combines feature induction and phrase break prediction in a single framework and thus avoids the two stage process required while using features derived from Latent Semantic Analysis (LSA). We train our model on audiobook data and show that

these task specific word features are better word representations as compared to those derived using LSA, for phrase break prediction.

With the advent of deep learning, there have been attempts to apply deep neural networks (DNNs) to phrase break prediction. While deep neural networks are able to effectively capture dependencies across features, they lack the ability to capture long-term relations that are spread over time. On the other hand, recurrent neural networks (RNNs) are able to capture long-term temporal relations and thus are better suited for tasks where sequences have to be modeled. This thesis presents the use of RNNs for phrase break prediction, and shows by means of experimental results that they perform better than DNNs.

Traditional techniques for learning word embddings learn static or context-free word embeddings, meaning that for each word in the vocabulary, one fixed embedding is learnt for all possible contexts in which that particular word appears in the text corpus. While static embeddings have proved to be effective in NLP applications, they do not take into account the fact that words have different meanings in different contexts. To address this issue, context-sensitive word representations, also called dynamic or contextual word embeddings were developed. The most popular technique for learning dynamic word embeddings is BERT (Bidirectional Encoder Representations from Transformers). This thesis presents work on phrase break prediction using bidirectional encoder representations learnt from fine-tuning a pretrained BERT model, with an additional token classification layer, on phrase break prediction. We show that representations using this model outperform task-specific static word embeddings learnt using a BLSTM token classification model trained from scratch.

Finally, this thesis presents work which attempts to answer the following questions: (a) Is there any utility in incorporating an explicit external phrasing model in an End-to-End TTS system? and (b) How do you evaluate the effectiveness of a phrasing model in an End-to-End TTS system? We perform subjective listening tests and show that incorporating explicit phrasing models in an End-to-End TTS system results in better listening comprehension, especially in the context of children's story synthesis.

Contents

Chapter Page

1 Introduction . . . 1
1.1 Phrase break prediction in Text-to-Speech Systems . . . 2
1.2 Scope of this thesis . . . 3
1.3 Organization of this thesis . . . 4

2 Phrase break prediction in Indian languages using word terminal syllables . . . 5
2.1 Database used in this study . . . 6
2.1.1 Annotation of phrase breaks . . . 6
2.2 Correspondence between phrase breaks and syntactic breaks . . . 7
2.2.1 Syntactic breaks used in the study . . . 9
2.2.2 Computing the correspondence between Phrase breaks and Syntactic breaks . . . 9
2.3 Correlation between word-terminal syllables and breaks . . . 10
2.4 Prediction of phrase breaks from text . . . 11
2.4.1 Simple rule based phrase break prediction . . . 12
2.4.2 Phrase break prediction using terminal syllable identity . . . 12
2.4.3 Phrase break prediction using POS tag identity . . . 13
2.4.4 Analysis of results . . . 14
2.5 Subjective evaluation of phrasing models . . . 15
2.6 Summary . . . 16

3 Phrase break prediction using continuous valued word representations/embeddings . . . 17
3.1 Word embeddings learnt using Latent Semantic Analysis . . . 18
3.2 Task specific neural word embeddings for phrase break prediction . . . 20
3.2.1 Transforming words into vectors . . . 20
3.2.2 Incorporating context . . . 21
3.2.3 Training the neural network . . . 22
3.3 Experiments . . . 22
3.3.1 Data used . . . 22
3.3.1.1 Annotation of phrase breaks . . . 22
3.3.2 Phrase break prediction using word embeddings learnt using Latent Semantic Analysis (LSA) . . . 23
3.3.3 Phrase break prediction using task specific neural word embeddings . . . 23
3.3.4 Systems built . . . 24
3.4 Results . . . 25
3.5 Summary . . . 26

4 Phrase break prediction using Recurrent Neural Networks . . . 28
4.1 RNN Architectures . . . 29
4.1.1 Simple RNN . . . 29
4.1.2 Long Short Term Memory (LSTM) . . . 32
4.2 RNNs for phrase break prediction . . . 34
4.3 Experimental results . . . 36
4.3.1 Data used . . . 36
4.3.1.1 Annotation of phrase breaks . . . 36
4.3.2 Experiment 1 : DNNs vs RNNs . . . 37
4.3.3 Experiment 2 : Effect of varying the size of the RNN hidden layer . . . 37
4.3.4 Experiment 3 : Effect of varying the dimension of word embeddings . . . 38
4.4 Summary . . . 39

5 Phrase break prediction using BERT . . . 40
5.1 Bidirectional Encoder Representations from Transformers (BERT) . . . 41
5.2 Experiments . . . 43
5.2.1 Data used . . . 43
5.2.2 Systems built . . . 44
5.2.2.1 Model 1: BLSTM with task-specific embeddings (BLSTM Model) . . . 44
5.2.2.2 Model 2: BERT model fine-tuned on phrase break prediction (BERT Model) . . . 46
5.2.3 Results . . . 47
5.3 Summary . . . 47

6 Application of phrase break prediction in End-to-End TTS systems . . . 49
6.1 End-to-End TTS system . . . 50
6.1.1 Architecture Details . . . 50
6.1.1.1 Tacotron2 model . . . 50
6.1.1.2 WaveRNN model . . . 52
6.1.2 Training details . . . 53
6.2 Phrasing models . . . 53
6.3 Experiments . . . 54
6.4 Summary . . . 56

7 Conclusions and future work . . . 58
7.1 Conclusions . . . 61
7.2 Future Work . . . 62

Bibliography . . . 65

List of Figures

Figure Page

2.1 Histograms of pause durations in all 6 languages 8

3.1 Neural network architecture to learn task-specific word embeddings for phrase break prediction 20
3.2 Selecting the embedding vector for word at index 992 by multiplying the embedding matrix **E** with a one-hot vector with 1 at dimension 992 and 0 everywhere else 21

4.1 Simple RNN architecture. The hidden layer includes a recurrent connection as part of its input. The hidden layer state at timestep t depends upon both the current timestep input and the hidden layer state at the previous timestep. $x(t), y(t)$ and $h(t)$ are the inputs, outputs and the hidden layer state at timestep t respectively 29
4.2 Simple RNN illustrated as a feedforward network. **W** represents the input layer to hidden layer weights, **U** represents the recurrence weights connecting the previous timestep hidden layer with the current timestep hidden layer, **V** represents the hidden layer to output layer weights, $h(t-1)$ is the hidden layer state at timestep $t-1$, $h(t)$ is hidden layer state at timestep t, $x(t)$ is input at timestep t and $y(t)$ is the output at timestep t 30
4.3 A simple RNN shown unrolled in time. Weights **U**, **W** and **V** are shared across all timesteps while h and y values are computed for each timestep 31
4.4 LSTM architecture. Inputs to the LSTM are the current timestep input $x(t)$, the previous timestep hidden layer state $h(t-1)$ and the previous timestep context $c(t-1)$. Outputs of the LSTM are the hidden layer state for the current timestep $h(t)$ and the updated context $c(t)$ 33
4.5 Phrase break prediction as a sequence labeling task using a RNN. Inputs are word embeddings which are randomly intialized and jointly trained along with the RNN model on task at hand, outputs are probabilities from a softmax layer over the set of possible tags (B and NB) 35

5.1 Transformer Encoder 42
5.2 BLSTM token classification model using task-specific static embeddings trained from scratch. Inputs to the model are word embeddings which are randomly initialized and jointly trained along with the model on the task at hand, outputs of the model are probabilities from a softmax layer over the set of possible tags (B and NB). 45
5.3 BERT model with a token classification head. The BERT model was pretrained on uncased English text and was later fine-tuned on phrase break prediction. The dense classification layer was randomly initialized, and its parameters were learned from scratch. 46

6.1 Architecture of the Tacotron2 model 51
6.2 Architecture of the WaveRNN model 52
6.3 Screenshot of the ABX test interface 55

List of Tables

Table		Page
1.1	Typical features used by machine learning models for phrase break prediction	3
2.1	Example sentence in Telugu annotated with locations of phrase and syntactic breaks (the word terminal syllables have been underlined) where *#B* denotes a phrase break, the numerical value in brackets denotes the break duration in milliseconds and | denotes a syntactic break	7
2.2	Example sentence in Hindi annotated with locations of phrase and syntactic breaks where *#B* denotes a phrase break, the numerical value in brackets denotes the break duration in milliseconds and | denotes a syntactic break	7
2.3	Correlation coefficients between syntactic breaks and phrase breaks for the six languages	10
2.4	Top word terminal syllables for all the languages. The figure in brackets is p(phrase break | terminal syllable)	11
2.5	F-Measure of rule based phrase break prediction in text	12
2.6	F-Measures of phrase break prediction for different contexts and pause duration $>$ 80ms for phrase breaks, where -1C and -2C represents one and two units context respectively to the left and +1C and +2C represents one and two units context respectively to the right	14
2.7	F-Measure of phrase break prediction, for different pause thresholds, where -2C represents two units context to the left and +2C represents two units context to the right	14
2.8	ABX Test Results. *** indicates statistically significant difference at 1% level as a result of a chi-squared test	15
3.1	Performance (in terms of the F-Measure) of the various systems on the phrase break prediction task	26
4.1	Parameter and hyperparameter values used in our models	36
4.2	Performance (in terms of the F-Measure) of the DNN, RNN and LSTM systems on the phrase break prediction task	37
4.3	Effect of varying the hidden (recurrent) layer size on the performance (in terms of the F-measure) of the Simple RNN model on the phrase break prediction task	38
4.4	Effect of varying the word embedding dimension on the performance (in terms of the F-measure) of the Simple RNN model on the phrase break prediction task	38
5.1	Model parameter and training hyperparameter values for the BLSTM token classification model using task-specific word embeddings trained from scratch	45

5.2 Fine-tuning parameter values for the BERT model with an additional token classification layer fine-tuned on phrase break prediction 47
5.3 Performance (in terms of the F-measure) of the BLSTM model trained from scratch and the fine-tuned BERT model on phrase break prediction 47

6.1 ABX evaluations between No Model, BLSTM Model and BERT Model. *** indicates statistically significant difference at 1% level as a result of a chi-squared test 55

Chapter 1

Introduction

Spoken utterances have an inherent structure in the sense that some words seem to group naturally together and some words seem to have a notable break or disjuncture between them. This can be described in terms of prosodic phrasing, meaning that a spoken utterance has a prosodic phrase structure, similar to how a written utterance has a syntactic phrase structure.

Phrase breaks in natural speech are important; they are physiologically essential, help emphasize content, and improve the intelligibility of speech. Phrase breaks can be classified into several levels [1]. They have a non-linear relationship with syntactic breaks [2], and are specific to a speaker [3, 4]. Phrase breaks are also a function of the type of text at hand. For example, the phrasing pattern of a speaker reading news text may be different from the phrasing pattern of the same speaker while reading a story book.

The process of inserting phrase breaks in an utterance is called phrasing. In the context of speech synthesis, phrasing is a crucial step. It breaks long utterances into meaningful units of information and improves the intelligibility of the synthesized speech. More importantly, phrasing is often the first step for other models of prosody, such as accent and intonation prediction [5, 6] and duration modeling [7]. Any errors made in the initial phrasing step are propagated to other prosody models, ultimately resulting in synthetic speech that is unnatural and difficult to understand. This is true even in the case of recent End-to-End speech synthesis architectures that are trained on <text, audio> pairs without the need for complex sub-systems.

To build and evaluate phrasing models, text data that has been annotated with phrase breaks is required. One widely adopted system for annotation of phrase breaks is the ToBI system (Tones and Break Indices) [1], which assigns a numeric value between 0 and 4 indicating different levels of breaks, using the following scheme:

Level 0; indicating clear phonetic marks of clitic groups

Level 1; indicating a non-break

Level 2; indicating an apparent disjuncture between words

Level 3; indicating an intermediate phrase break within an utterance

Level 4; indicating a major break, such as the end of an utterance

The Festival speech synthesis system [8,9] simplifies the ToBI system to include just Levels 1, 3 and 4. During synthesis, the default English phrasing model present in the Festival system decides whether a word boundary is a non-break (Level 1) or a break (Level 3). The system also places a Level 4 break by default at the end of an utterance.

Several corpora annotated using the ToBI system are available for building phrasing models. Two commonly used ones for English are the Boston University Radio News Corpus [10] and the MARSEC corpus [11]. However, a phrasing model trained on such corpora would be generic, and the TTS systems using such a model would not be able to adapt to a specific speaker's phrasing and speaking style. In such cases, it is entirely possible that downstream intonation models fail to appropriately capture the F0 pattern of the speaker as shown in [12]. Thus, it is important to build phrasing models targeted towards the voice being used to build the TTS system. Typically, this is achieved by deriving the locations for phrase breaks from the speech data. This can be achieved either by manually listening to the speech and annotating the text prompts, or by force aligning the speech with the corresponding text prompts which would yield the locations of the breaks introduced by the speaker while recording the utterances.

1.1 Phrase break prediction in Text-to-Speech Systems

The phrase break prediction task within TTS can be formally defined as follows: "*Given an utterance represented as a sequence of words intended for TTS synthesis, predict at each word boundary whether a pause should be inserted after that word.*"

Given an annotated corpus, two main approaches exist to building phrasing models [13]: (i) Rule based deterministic models, and (ii) Data driven models. Rule based deterministic models, such as those described in [14, 15] are easy to build and require no training data. On the other hand, data driven models use machine learning algorithms to make break/non-break predictions after being trained on large amounts of training data.

The simplest rule based deterministic model is the punctuation based model, where a break is inserted wherever there is a punctuation in the text. In [2], the authors show that while the precision of such a model is high, the recall is only around 50%. This means that while many the breaks predicted by the model are correct, the model predicts about only half of the total breaks present in the data. The model is easy to build, and is the default model when building a new voice in Festvox [16]. However, this approach faces an issue in that many languages are not well punctuated. For example, text in many Indian languages do not have punctuations apart from sentence endings. Even in languages that are normally well punctuated, such as English, there are forms of communication such as tweets, SMS and emails which use informal writing which lacks punctuation. In tasks such as Speech-to-Speech translation or dialog systems, the text to be synthesized may be the output of another module and lack punctuation.

Data driven phrasing models use machine learning algorithms for break/non-break classification. Various machine learning models have been used for the task: decision trees [17–19], Hidden Markov Models (HMMs) [2, 20], memory based learning [21], bayesian networks [22, 23], maximum entropy models [24] and neural networks [25].

In natural speech, prosodic phrases tend to be balanced. This means that a phrasing model should consider the location of the previous break while making a prediction. In models that use only local features (for example POS tags) a mechanism should be included to optimize the global phrasing over the utterance. For example, in [2] the authors use a 7-gram language model for break sequences and [20] use the distance from the previous break as a feature to parameterize their HMM model. Table 1.1 lists out some typical features used by machine learning models for phrase break prediction.

Table 1.1 Typical features used by machine learning models for phrase break prediction

part of speech	- POS tag of current word - POS tags of previous two words - POS tags of next two words
length features	- Total number of words and syllables in utterance - Distance of word boundary from the beginning and end of the utterance (in number of words) - Distance from last punctuation (in number of words)
punctuation	- type of following punctuation - type of previous punctuation

1.2 Scope of this thesis

This thesis addresses the problem of prediction and incorporation of phrase break models in Text-to-Speech systems. Typically, phrase breaks are manifested in the speech signal in the form of several acoustic cues like pauses as well as relative changes in the intonation and duration of syllables. Cues such as pre-pausal lengthening of rhyme, speaking rate, breaths, boundary tones and glottalization also play a role in indicating phrase breaks in speech [26–28]. However, representing these non-pause acoustic cues in terms of features is not easy and not well understood [3]. In [3, 4, 29–32] the authors limit their phrase break models to predicting the locations of pauses while synthesizing speech. In this thesis too, we limit our phrase break models to predict the location of pauses while synthesizing speech.

All the models presented in this thesis have been trained using text in which the location of pauses have been derived from the speech data. This has been achieved by force aligning the speech utterances with the corresponding text prompts.

1.3 Organization of this thesis

Chapter 2 presents our work on phrase break prediction for Text-to-Speech systems in Indian languages. We look at syllable level units and explore the use of terminal syllables to model phrase breaks. We utilize these terminal syllables in building models for phrase break prediction from text in six Indian languages and perform objective and subjective evaluations to study the performance of these models built using terminal syllables.

Chapter 3 presents our work on learning continuous valued word representations, i. e., representing words as points in a continuous dimensional space, also called word embeddings, for phrase break prediction. We present two methods for learning word embeddings, the first which learns word embeddings using a matrix factorization method based on latent semantic analysis (LSA), while the second learns task-specific neural word embeddings using a neural network architecture. We train several systems using both LSA word embeddings and task-specific neural word embeddings, on audiobook data for phrase break prediction and evaluate their performance.

Chapter 4 presents our work on RNNs for phrase break prediction and is primarily motivated by the following question: "Does capturing long-term temporal context using a recurrent architecture result in a better performance on phrase break prediction, as compared to the limited temporal context captured using a moving fixed-size window as in the case of DNNs?" We train DNNs, Simple RNNs and LSTMs on audiobook data for phrase break prediction, and perform experiments to study their performance.

Chapter 5 presents our work on phrase break prediction using bidirectional encoder representations. We train two models for phrase break prediction: (i) A BLSTM token classification model using task-specific static word embeddings trained from scratch, and (ii) A BERT model with an additional token classification layer fine-tuned on phrase break prediction. We evaluate the performance of both models on phrase break prediction report the results.

Chapter 6 presents our work on phrase break prediction in the context of End-to-End TTS systems, which is motivated by the following questions: (i) Is there any utility in incorporating an explicit phrasing model in an End-to-End TTS system?, and (ii) How do you evaluate the effectiveness of a phrasing model in an End-to-End TTS system? We perform subjective listening tests study the utility and effectiveness of explicit phrasing models in End-to-End TTS systems.

Finally, Chapter 7 discusses the contributions of this thesis, and presents some directions for future work.

Chapter 2

Phrase break prediction in Indian languages using word terminal syllables

Traditionally phrase break prediction has been achieved by using machine learning models in conjunction with data labelled with linguistic classes (such as part-of-speech (POS) tags, phrase structure etc.). Many different machine learning algorithms have been applied to phrase break prediction; for example, decision trees [18, 33]; n-gram models [2, 20]; finite state transducers [34] and memory based learning [35], all of which directly use POS tags as an input feature. Parlikar and Black [36] used POS tags to construct grammar based parse trees, which provided features for a decision tree based phrase break predictor. Previous work therefore suggests that POS tagging is a necessary first step in phrase break prediction.

All these methods assumed the availability of either hand labeled training data, or high quality POS taggers/shallow parsers which can generate POS tags for the training data with a high level of accuracy. As a result, these methods cannot be used for languages where the necessary linguistic resources are not readily available, and manual annotation of data is expensive and time-consuming. In view of these limitations, there was a lot of interest in unsupervised methods of inducing word representations to be used as surrogates for POS tags in the phrase break prediction task. Parlikar and Black [37] used the Ney-Essen clustering algorithm [38] to automatically induce POS tags from text using frequency analysis of words. However, this approach faces an issue when applied to Indian languages, which are agglutinative in nature. In these languages words are formed by joining morphemes together. Moreover, due to the postpositional nature of these languages, syllable level suffixes get attached to the ending of words. These suffixes give specific syntactic meaning in terms of tense, gender etc. These characteristics of Indian languages result in an increase in vocabulary size, i. e. the number of words, making it hard to work with Indian languages using this representation.

For Indian languages, a better solution would be to deal with sub-word units like syllables or multi-syllable units [39–41]. In [39], a set of *morpheme tags* were manually identified and used to model phrase breaks. The *morpheme tags* consist of one or two syllables, typically found at the end of the word. The experiments were conducted on Telugu. Manual identification of this set of morpheme tags required sufficient linguistic knowledge and was not easy.

In this chapter, we present our work on the use of word-terminal syllables, i. e., the last syllable in a word, to model phrase breaks in Indian languages. We automatically identify the set of terminal syllables in six Indian languages, and report results on the prediction of phrase breaks from text using these terminal syllables. Finally, we incorporate the proposed phrase break prediction model in a Text-to-Speech system, and demonstrate its usefulness with listening tests.

The rest of this chapter is organized as follows: Section 2.1 describes the database used in the study, Section 2.2 discusses the correlation between phrase breaks and syntactic breaks, Section 2.3 presents an emperical analysis which shows strong correlation between terminal syllables and phrase breaks, Section 2.4 presents our experiments on phrase break prediction using terminal syllables along with objective results, Section 2.5 presents subjective evaluation results by means of listening tests and Section 2.6 summarizes the chapter.

2.1 Database used in this study

In our study we looked at two different corpora that have speech in different styles.

1. The ***IIIT-MCIT (Lenina)*** corpus is a corpus developed at IIIT Hyderabad, which was used to build a synthetic voice in Telugu. This corpus consists of text prompts taken from a set of popular children's stories in Telugu, and the corresponding recordings recorded in a story telling style in a clean studio environment. The corpus has 4043 text prompts and the audio size is about 6 hours. The style of the corpus is "story telling".

2. The ***IIIT-H Indic*** [42] database consists of text and speech data in Telugu, Hindi, Kannada, Tamil, Malayalam, Marathi and Bengali. Each language in the database consists of a set of 1000 text prompts selected from Wikipedia articles in the corresponding language, selected in such a way as to cover the 5000 most frequent words in the corresponding languages. The corresponding recordings were recorded by a native speaker of each language in a clean studio environment. On an average the size of the audio is about 1.5 hours for each language. The style of this corpus is "isolated sentences".

2.1.1 Annotation of phrase breaks

As we did not have a corpus with hand annotated phrase breaks, we derived the location and duration of phrase breaks from the speech data. In order to derive the locations and durations of pauses introduced by the speaker, we force aligned the speech with the corresponding text prompts using the HMM tool in Festvox [43]. This gave us the locations of pauses introduced by the speaker while recording the utterances.

One question that arose at this point, was the following: "What should be the minimum duration of a pause for it to be considered a phrase break?" To answer the above question, we analyzed the durations

of the pauses introduced by the speaker for both *Lenina* and *IIITH-Indic* databases. Figure 2.1 shows the histogram plots of silence durations for all the six languages.

An analysis of the histogram plot showed that for Hindi the majority of pauses are less than 80 ms in duration, for Telugu (IIITH-Indic), Kannada and Tamil the majority of pauses range from a few milliseconds to 480 ms and for Bengali the pause durations range from a few milliseconds to 640 ms. In the case of the *Lenina* database, the pause durations range mainly from 80 ms to 640 ms.

We thus observed that the pause durations varied over a significant range within a language and also between languages. We experimented with thresholds of 25 ms, 50 ms and 80 ms, wherein all pauses with durations greater than the threshold were marked as phrase breaks. Furthermore, we also experimented with the case where we marked all pauses as phrase breaks regardless of their duration.

2.2 Correspondence between phrase breaks and syntactic breaks

One question that is often asked is the following: "Is there any correspondence between phrase breaks and syntactic breaks?" While it is known that there is some correspondence between syntax and prosody, the relationship between them is not formally defined [2], [4]. This is illustrated by means of two examples.

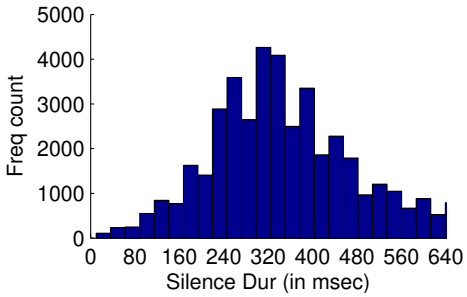
Table 2.1 Example sentence in Telugu annotated with locations of phrase and syntactic breaks (the word terminal syllables have been underlined) where *#B* denotes a phrase break, the numerical value in brackets denotes the break duration in milliseconds and | denotes a syntactic break

"brahmadattud:u #B (260ms) kaashiiraajyaanni paripaalin:chei kaalan:loo #B (440ms) aa nagaran:loo #B (410ms) dhanikud:aina #B (360ms) oka goppavartakud:un:d:eivaad:u"
"brahmadattud:u | kaashiiraajyaanni | paripaalin:chei | kaalan:loo | aa nagaran:loo | dhanikud:aina | oka goppavartakud:un:d:eivaad:u"

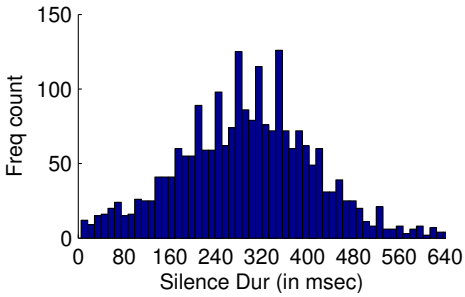
Table 2.2 Example sentence in Hindi annotated with locations of phrase and syntactic breaks where *#B* denotes a phrase break, the numerical value in brackets denotes the break duration in milliseconds and | denotes a syntactic break

"san:bhava hai ki #B (80ms) isakaa #B (65ms) aavishhkaara #B (10ms) isasei bhii bahuta pahalei huaa hoo"
"san:bhava | hai | ki | isakaa aavishhkaara | isasei bhii | bahuta pahalei | huaa hoo"

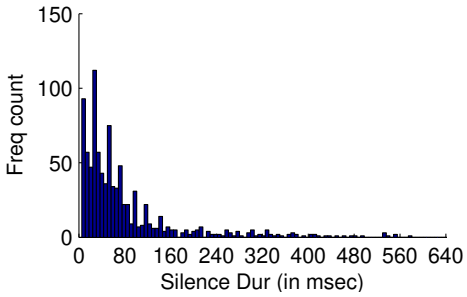
Consider the Telugu sentence (represented using the ITRANS transliteration scheme) in Table 2.1, taken from the *Lenina Database*, and the Hindi sentence in Table 2.2 taken from the *IIIH-Indic* database,



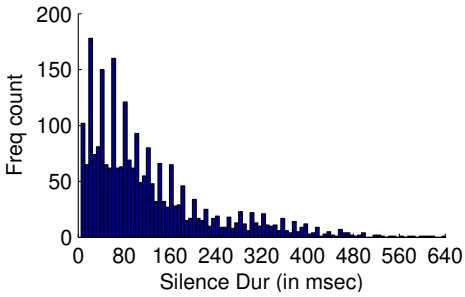
(a) Telugu (Lenina)



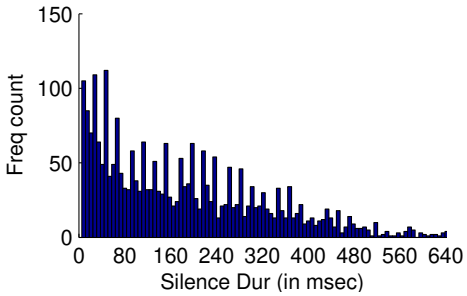
(b) Telugu (Indic)



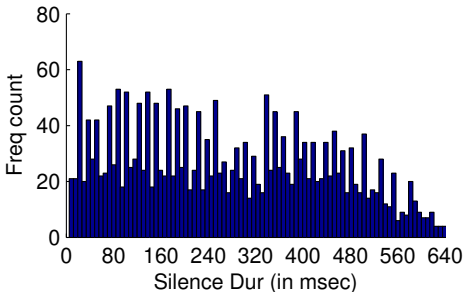
(c) Hindi



(d) Kannada



(e) Tamil



(g) Bengali

Figure 2.1 Histograms of pause durations in all 6 languages

both of which have been annotated with the location of prosodic and syntactic breaks. In both the examples it is clear that while there is some correspondence between the phrase breaks and the syntactic breaks, the relationship between them is not direct.

2.2.1 Syntactic breaks used in the study

The syntactic breaks were derived from text using the shallow parser [44] developed at IIIT Hyderabad, which uses conditional random fields (CRF) and transformational based learning (TBL) to perform chunking and POS tagging of text. In [44] the authors report accuracies of 77.37%, 78.66% and 76.08% on chunking and 79.15%, 80.97% and 83.74% on POS tagging, for Telugu, Hindi and Bengali respectively.

The location of syntactic breaks was derived by running the shallow parser on the text data. The tool parsed the text into syntactic constituents, which gave us the location of the syntactic breaks.

2.2.2 Computing the correspondence between Phrase breaks and Syntactic breaks

The correspondence between phrase and syntactic breaks, for all languages under consideration: Telugu, Hindi, Kannada, Tamil and Bengali were computed as follows. For each word, a binary feature indicating the presence or absence of a break after that word, was computed. The presence of a break was indicated by 1 and the absence of a break by -1.

Let $\mathbf{S} = [s_1, \cdots, s_w, \cdots, s_N]$ denote the sequence of binary features derived for the words in the database using syntactic break information, where N denotes the total number of words in the database.

Let $\mathbf{P} = [p_1, \cdots, p_w, \cdots, p_N]$ denote the sequence of binary features derived using phrase breaks (pauses in speech), where N denotes the total number of words in the database. These breaks were computed using the phrase break annotation described in 2.1.1.

The correlation coefficient between **S** and **P** was calculated using the following equation

$$c(\mathbf{S}, \mathbf{P}) = \frac{\sum_{w=1}^{N}(s_w - \bar{s})(p_w - \bar{p})}{\sqrt{\sum_{w=1}^{N}(s_w - \bar{s})^2}\sqrt{\sum_{w=1}^{N}(p_w - \bar{p})^2}}$$

where $\bar{s}$ and $\bar{p}$ denote the mean values of **S** and **P** respectively.

Table 2.3 shows the correlation coefficients between syntactic and phrase breaks for the six languages.

An observation of the values in Table 2.3 shows that the correlation coefficient between syntactic breaks and phrase breaks, does not exceed 0.3 for all the languages. This indicates that there is a significant variation between syntactic and phrase breaks, for the languages under consideration. As a result syntactic breaks cannot be used directly to model phrase breaks.

Table 2.3 Correlation coefficients between syntactic breaks and phrase breaks for the six languages

Language	Correlation Coefficient
Telugu (Lenina)	0.26
Telugu (Indic)	0.27
Hindi	0.12
Kannada	0.29
Tamil	0.18
Bengali	0.20

2.3 Correlation between word-terminal syllables and breaks

As mentioned earlier, in order to model phrase breaks, we look at syllable level features. An examination of the Telugu example in Table 2.1 shows that a phrase break has occurred after two words ending in the syllable *loo*, and after a word ending in the syllable *d:u*. The utterance ending break has also occurred after a word ending in the syllable *d:u*. This motivated us to look at word terminal syllables i. e. the last syllable of the word, as a feature set to model phrase breaks. We performed an analysis, whereby the correlation between the word terminal syllables and phrase breaks was studied. As part of this analysis we computed the conditional probability $p(\text{break} \mid \text{terminal syllable})$ for each terminal syllable, as follows

$$p(\text{break} \mid \text{terminal syllable}) = \frac{N(\text{break}, \text{terminal syllable})}{N(\text{terminal syllable})}, \quad \forall N(\text{terminal syllable}) > 50 \tag{2.1}$$

where $N(\text{break}, \text{terminal syllable})$ is the count of the number of times that a break occurs after a word ending in that particular terminal syllable, and $N(\text{terminal syllable})$ is the total number of words ending in that particular terminal syllable, respectively in the data.

Our analysis showed that for a few terminal syllables, the probability of a break occurring after a word ending in that syllable is high. That is, for a few terminal syllables the value of the conditional probability $p(\text{phrase break} \mid \text{terminal syllable})$ is high. This high conditional probability tapers off beyond these top few terminal syllables. Table 2.4 shows the top word terminal syllables, derived from this analysis for each of the six languages. The values in the parentheses are the values of $p(\text{phrase break} \mid \text{terminal syllable})$ for those particular syllables.

As can be observed from the Table 2.4 we can see that the value of $p(\text{phrase break} \mid \text{terminal syllable})$ for the top syllables are in the range 0.6–1.0. As a result, we hypothesize that these word terminal syllables are good candidates for a feature set to predict phrase breaks from text.

Table 2.4 Top word terminal syllables for all the languages. The figure in brackets is p(phrase break | terminal syllable)

Telugu(Lenina)	nai (0.98), buu (0.97), jaa (0.92), chchu (0.90), du (0.87), vaa (0.87), yyaa (0.85), daa (0.84), stei (0.83), yi (0.80), tei (0.80), di (0.79), chchi (0.79), t:ei (0.79), mmaa (0.78), d:u (0.78), llaa (0.75), chii (0.74), ppi (0.73), chi (0.73), $\cdots$
Telugu (Indic)	d:i (0.84), mu (0.82), loo (0.78), yi (0.77), du (0.76), di (0.70), nu (0.67), san: (0.65), llaa (0.64), d:u (0.63), vu (0.61), ru (0.58), d:aa (0.57), ki (0.54), ni (0.54), gaa (0.53), lu (0.53), ku (0.51), ran: (0.50), chi (0.48), $\cdots$
Hindi	hai (0.75), hain: (0.74), thaa (0.64), sha (0.43), da (0.40), ei (0.34), koo (0.26), yaa (0.24), nd~a (0.24), la (0.23), pa (0.23), kaa (0.22), ga (0.22), va (0.21), sa (0.21), na (0.20), ra (0.19), ti (0.18), kha (0.17), bhii (0.17), $\cdots$
Kannada	de (0.97), ki (0.93), lli (0.69), ru (0.67), re (0.65), nnu (0.62), l:u (0.61), gi (0.57), ge (0.57), da (0.54), ya (0.54), du (0.52), na (0.51), tra (0.50), ti (0.43), ga (0.43), vu (0.42), ttu (0.42), ka (0.41), ra (0.40), $\cdots$
Tamil	kum (1.00), n~ar (0.99), llai (0.96), lam (0.94), r:r:i (0.77), chan~ (0.73), ng~kal: (0.72), thu (0.69), rai (0.67), than~ (0.61), kal: (0.57), rkal: (0.54), ntha (0.53), n~r:u (0.48), yil (0.47), ththil (0.46), ththu (0.35), ya (0.35), ka (0.32), kku (0.32), $\cdots$
Bengali	hay (0.91), chhi (0.81), nya (0.78), sa (0.77), chhe (0.70), da (0.66), ja (0.60), naa (0.59), sha (0.58), ban: (0.57), i(0.57), ba (0.56), be (0.53), ke (0.51), na (0.51), t:a (0.50), re (0.50), nd~a (0.48), le (0.48), ga (0.48), $\cdots$

2.4 Prediction of phrase breaks from text

Prediction of phrase breaks from text can be achieved by building a phrasing model. Typically, as a first approach a punctuation based phrasing model is used. The output of this model is then refined by models built using POS tags and other linguistic information. However, text in Indian languages very rarely has any punctuations (except for sentence endings). Hence, when dealing with Indian languages a simple punctuation based phrasing model will not work and more sophisticated phrasing models are required. These models can either be a set of heuristic rules or a machine learning model trained on features extracted from text. Generally, the first step in building such phrasing models involves annotating the text with phrase breaks, which has been described in Section 2.1.1. This annotated text can be used in several ways. The text can be used to generate a set of heuristic rules, for deriving the location of phrase breaks. We can also extract several features from the annotated text, to train a machine learning model, for predicting phrase breaks in text.

We experimented with three different approaches and report the results of phrase break prediction from text, for both the *Lenina* and *IIITH-Indic* databases. In our first approach we generate a simple rule which utilizes the syntactic break location (obtained from the shallow parser (Section 2.2)) along with terminal syllable information to derive the location of the phrase breaks in text. Our second approach utilizes terminal syllable identity, extracted from text, to build a machine learning model for phrase break prediction. In our third approach, we use POS tags, (which we obtain from running the shallow parser over text (Section 2.2)) to build a machine learning model for phrase break prediction.

As text in these databases has already been annotated with prosodic phrase breaks, reference labels to compute the performance of our approaches is available. We report the performance of our approaches

in terms of the F-measure [45] which is defined as the harmonic mean of the precision and recall. F-measure values range from 0 to 1, with higher values indicating better performance.

2.4.1 Simple rule based phrase break prediction

We generated a simple rule to give us the location of phrase breaks in text. This rule uses syntactic break locations along with the terminal syllable information for deriving the locations of the phrase breaks in text.

During our analysis of the correlation between terminal syllables and phrase breaks (presented in Section 2.3) we observed that the top 50 word terminal syllables in each language have a high correlation of occurrence along with phrase breaks. We combined our knowledge of syntactic break locations (derived using the shallow parser (Section 2.2)) with this observation to generate the following heuristic rule for phrase break prediction from text.

"For each word in the text, if (a) A syntactic break occurs after that word, and (b) The last syllable of that word (the word-terminal syllable) is among the list of the top 50 word-terminal syllables for that language, then, a phrase break should be inserted after that word."

We used this rule to derive the locations of the phrase breaks in text for both the *Lenina* and *IIITH-Indic* databases.

Table 2.5 F-Measure of rule based phrase break prediction in text

Language	F-Measure
Telugu(Lenina)	0.62
Telugu(Indic)	0.57
Hindi	0.24
Kannada	0.55
Tamil	0.47
Bengali	0.49

Table 2.5 displays the F-measures of our heuristic rule based prediction of phrase breaks, for all six languages. An analysis of the numbers shows that, except Hindi, the rule based system performs with F-measures ranging from 0.45 to 0.75.

2.4.2 Phrase break prediction using terminal syllable identity

We used the identity of the word-terminal syllable, in a Classification and Regression Tree (CART) framework, to build a model (henceforth referred to as TS model) for predicting phrase breaks from text. To take context into account we used the word-terminal syllable identity of the previous two words and

the next two words, along with that of the current word. We also experimented with different context windows. We used 90% of the text in each language as training data while the remaining 10% was held back for testing. As it was a trivial task to predict breaks at utterance endings, we removed the feature vectors corresponding to utterance ending breaks from the training data.

The following example serves to illustrate how feature vectors were generated from the annotated text for our phrase break model. Consider the following Telugu sentence:
"brahmadattud:u #B kaashiiraajyaanni paripaalin:chei kaalan:loo #B aa nagaran:loo #B dhanikud:aina #B oka goppavartakud:un:d:eivaad:u"
annotated with the location of phrase breaks. For the word *nagaran:loo*, the previous two words are *kaalan:loo* and *aa*, the next two words are *dhanikud:aina* and *oka* and the variable to be predicted is *#B*. Considering a two word context on either side of the word in question (*nagaran:loo*), the feature vector for this word then becomes:

#B loo aa loo na ka

where *loo* and *aa* are the identities of the word-terminal syllables of the previous two words, *loo* is the identity of the word-terminal syllable of the current word and *na* and *ka* are the identities of the word-terminal syllables of the next two words.

As an initial experiment, we considered the case where word boundaries that coincide with pauses greater than 80ms were marked as phrase breaks, while all other word boundaries were marked as non breaks. We generated feature vectors for both breaks and non breaks, using the terminal-syllable identity along with context. As this was a binary classification task, we also ensured that the number of feature vectors of each for class (break and non break) were the same. As the number of non breaks were more than the number of breaks in the data, this was achieved by removing non break feature vectors till the numbers were the same. These feature vectors were then used to train the CART model.

We also experimented with different pause thresholds for marking phrase breaks, while keeping the context the same. For the purpose of this experiment, we considered the contextual information provided by the word-terminal syllable identities of the previous two and the next two words immediately adjacent to the current word. As before, we took 90% of the text in each language as training data while the remaining 10% was held back for testing and breaks corresponding to utterance endings were omitted from the training data. In this case also, we ensured that the number of training feature vectors for both classes (breaks and non breaks) were the same.

2.4.3 Phrase break prediction using POS tag identity

We used the same experimental setup as in Section 2.4.2 changing only the features used. We used the identity of the POS tag of each word, as features in a Classification and Regression Tree (CART) framework, to build a model (henceforth referred to as POS model) for predicting phrase breaks from text. As the POS tag identity is a word level feature, we experimented with different word level contexts in order to incorporate contextual information. We repeated all the experiments presented in Section 2.4.2 using POS tag identity as features.

2.4.4 Analysis of results

Table 2.6 F-Measures of phrase break prediction for different contexts and pause duration $>$ 80ms for phrase breaks, where -1C and -2C represents one and two units context respectively to the left and +1C and +2C represents one and two units context respectively to the right

Language	TS	POS	-1C, TS	-1C, POS	-1C, TS, +1C	-1C, POS, +1C	-2C, TS	-2C, POS	-2C, TS, +1C	-2C, POS, +1C	-2C, TS, +2C	-2C, POS, +2C
Telugu(Lenina)	0.63	0.62	0.69	0.63	0.72	0.69	0.69	0.64	0.72	0.75	**0.74**	**0.75**
Telugu(Indic)	0.47	0.56	0.49	0.56	0.53	0.59	0.47	0.52	0.53	0.60	**0.52**	**0.59**
Hindi	0.09	0.10	0.09	0.10	0.09	0.09	0.09	0.09	0.09	0.15	**0.06**	**0.18**
Kannada	0.37	0.41	0.40	0.42	0.40	0.41	0.39	0.42	0.40	0.44	**0.39**	**0.43**
Tamil	0.43	0.39	0.42	0.43	0.39	0.44	0.43	0.44	0.41	0.47	**0.42**	**0.45**
Bengali	0.41	0.56	0.34	0.53	0.47	0.53	0.37	0.53	0.49	0.56	**0.49**	**0.54**

Table 2.7 F-Measure of phrase break prediction, for different pause thresholds, where -2C represents two units context to the left and +2C represents two units context to the right

Language	pause $>$50 ms taken as break		pause $>$25 ms taken as break		all pauses taken as break	
	-2C, TS, +2C	-2C, POS, +2C	-2C, TS, +2C	-2C, POS, +2C	-2C, TS, +2C	-2C, POS, +2C
Telugu(Lenina)	0.72	0.75	0.73	0.75	0.73	0.75
Telugu (Indic)	0.54	0.60	0.54	0.62	0.54	0.62
Hindi	0.37	0.24	0.38	0.28	0.47	0.31
Kannada	0.46	0.51	0.47	0.50	0.55	0.59
Tamil	0.46	0.54	0.55	0.57	0.58	0.61
Bengali	0.51	0.59	0.53	0.59	0.53	0.58

Table 2.6 shows the performance of both the TS and POS models, when word boundaries greater than 80ms are marked as phrase breaks. An observation of the table shows that the performance of both the models for Hindi is poor. In this case, as observed in Section 2.1.1 the majority of the pause durations are less than 80ms. As as result the number of feature vectors for phrase breaks are very few, resulting in a poorly trained model. Also in case of Hindi the sentences are short, and so there are few pauses in the middle of the sentences.

Table 2.7 shows the performance for different pause thresholds, while keeping the context same in all cases.

An analysis of all the F-measure numbers shows that, for automatic prediction of phrase breaks from text, models built using terminal syllable identity perform nearly as well as models built using traditional features like part-of-speech (POS) tags.

2.5 Subjective evaluation of phrasing models

We performed subjective listening tests for Telugu, to compare utterances synthesized by incorporating the TS and POS models with utterances synthesized with no explicit phrasing model. The listening tests were set up as an ABX task, for native speakers of Telugu. Two phrasing models were compared at a time. An utterance was synthesized by incorporating both models and both versions were presented to the participants in a randomized order, and the participants were asked to mark the version they preferred. They also had an option of no preference if they could not pick one utterance over the other.

In the first listening task, we compared utterances synthesized by incorporating the POS model with utterances synthesized with no explicit phrasing model. For the second listening task, we compared utterances synthesized by incorporating the TS model with utterances synthesized with no explicit phrasing. Finally, we performed a third listening task where we compared utterances synthesized by using the POS model with utterances synthesized using the TS model. All the listening tasks were performed by 10 native speakers of Telugu, who evaluated 15 samples picked randomly from the test set. Table 2.8 shows the results of these listening tests.

Table 2.8 ABX Test Results. *** indicates statistically significant difference at 1% level as a result of a chi-squared test

Model A	**Model B**	**Preference A**	**Preference B**	**No Preference**	
No Phrasing Model	POS Model	10%	75%	15%	***
No Phrasing Model	TS Model	12%	73%	15%	***
TS Model	POS Model	35%	34%	31%	

An examination of the results in Table 2.8 shows that perceptually there is a preference for utterances synthesized incorporating a phrasing model over utterances synthesized with no explicit phrasing. The results also show that when the POS model and the TS model are compared with each other, perceptually there is no significant preference for one model over the other.

2.6 Summary

In this chapter, we described our work on phrase break prediction for Text-to-Speech systems in Indian languages. We looked at syllable level units and explored the use of terminal syllable to model phrase breaks. We demonstrated the correlation between terminal syllables and acoustic breaks (pauses) found in the speech signal. We also demonstrated that there is a nonlinear relationship between syntax and prosody, with significant variations between syntactic breaks and phrase breaks.

We utilized these terminal syllables in building models for phrase break prediction from text in six Indian languages and demonstrated by means of objective and subjective measures that models built using terminal syllables identities perform as well as traditional models built using part-of-speech (POS) tags.

The advantage of these terminal syllables, is that they can be directly derived from the text under consideration, thus eliminating the need for additional linguistic resources like shallow parsers or POS taggers, while also eliminating the need to model phrase breaks by computationally expensive unsupervised models.

Chapter 3

Phrase break prediction using continuous valued word representations/embeddings

Traditional methods of phrase break prediction have used discrete linguistic representations like POS tags/phrase structure [2, 18, 20, 33–36], induced POS tags [37], morpheme tags [39] and word terminal syllables (Chapter 2) [46]. However, such representations require a hard classification of words into a set of predefined discrete classes. This raises issues when there is ambiguity in the linguistic representation of the word. For example, the English word *plant* can be categorized as a noun or as a verb depending upon the context it occurs in. Also, such a representation does not capture the co-occurrence statistics of words, i. e., they do not take into account the distributional behavior of words.

To address these issues, there have been efforts towards learning continuous valued representations of words, i. e., representing words as points in a continuous dimensional space, also called word embeddings, for various natural language processing (NLP) applications. Learning these word embeddings typically entails learning a transformation for each word in a raw text corpus to a dense lower dimensional vector space. Most word embedding methods learn word representations with the distributional hypothesis [47, 48], i. e., words with similar contexts are expected to have similar meanings. These embeddings are static embeddings, meaning that for each word in the vocabulary, one fixed embedding is learnt for all possible contexts in which that particular word appears in the text corpus. There exist two broad approaches to learning word embeddings: (i) Matrix factorization methods, and (ii) Neural methods.

Matrix factorization methods [49, 50] learn word embeddings from global corpus statistics. Here words are mapped to points in a high-dimensional sparse space defined by the rows of a word co-occurrence matrix. A transformation is then applied to this matrix to project it into a dense lower dimensional space (called the latent space or embedding space). This latent space matrix captures the distributional behavior of the words in as few dimensions as possible.

In [51,52], the authors described a unified multitask architecture for NLP that learnt features relevant to the tasks at hand. Their approach utilized a convolutional neural network architecture trained in an end-to-end fashion. The input sentence was first processed through several layers of feature extraction, and these features were then automatically trained using gradient descent to be relevant to the task.

They reported results on the following six standard NLP tasks: POS Tagging, Chunking, Named Entity Recognition, Semantic Role Labeling, Language Modeling and Semantically Related Words. This was one of the first attempts to learn word embeddings using neural networks. Subsequently, neural embedding models such as word2vec's skip-gram and CBOW models [53] which learnt word embeddings by predicting contexts of the target word, were developed.

These word embeddings performed well on tasks such as word similarity and word analogy. This was due to the fact that these tasks required semantic information which is captured naturally using word contexts. However, as shown in [54–56] these word embeddings could not be optimized to a specific task (like text classification, sentiment analysis, information retrieval etc.), and were shown to be suboptimal when used in task-specific applications. One proposed solution was to refine/fine-tune the word embeddings to capture task-specific features in addition to the semantic information captured using word contexts. Another approach was to directly model task-specific features in order to construct the embedding space, thus learning task-specific word embeddings.

In this chapter, we present our work on word embeddings for phrase break prediction. We present a method to learn word embeddings based on Latent Semantic Analysis (LSA), and report results on directly using these embeddings for phrase break prediction. We also present a neural network architecture for directly learning task-specific word embeddings for phrase break prediction, as well as fine-tuning LSA embeddings for phrase break prediction, and report our results on the same.

The rest of this chapter is organized as follows: Section 3.1 describes a technique for obtaining word embeddings using a matrix factorization method based on Latent Semantic Analysis (LSA), Section 3.2 describes our neural network architecture for learning task-specific word embeddings for phrase break prediction, Section 3.3 describes our experiments, Section 3.4 reports the results of our experiments and Section 3.5 summarizes the chapter.

3.1 Word embeddings learnt using Latent Semantic Analysis

To learn word embeddings using Latent Semantic Analysis (LSA), we followed the same method as in [31]. This method is based on the first POS induction method in [49]. Here words are mapped to points in a high-dimensional sparse space defined by the rows of a word co-occurrence matrix. A transformation is then applied to this matrix to project it into a lower-dimensional embedding space. The rows of this embedding matrix are the embeddings representing each word type in the corpus.

The method is described in detail below.

1. Given a corpus of $|V|$ unique word types i. e. $|V|$ is vocabulary size of the corpus, a subset n is chosen as feature words (typically $n << |V|$).

2. A $|V| \times 2n$ co-occurrence matrix $\mathbf{C}$ is computed where:

   (a) C_{ij} is the count of number of times the i^{th} word type occurs with the j^{th} feature word as left-hand neighbor (left context)

(b) C_{ij+n} is the count of number of times the same i^{th} word type occurs with the same j^{th} feature word as right-hand neighbor (right context)

Each of the $|V|$ unique word types in the corpus are thus mapped to points in a sparse $2n$ dimensional space defined by the rows of the raw co-occurrence matrix.

3. This $|V| \times 2n$ co-occurrence matrix is sparse, and each vector has a dimension of $2n$. A transformation is applied to this raw co-occurrence matrix to project it into a dense lower dimensional space. A typical transformation is to apply Singular Value Decomposition (SVD) on the raw co-occurrence matrix.

$$\mathbf{C}_{|V|\times 2n} = \mathbf{U}_{|V|\times d}\mathbf{D}_{d\times d}\mathbf{V}^T_{d\times 2n}$$

where $\mathbf{D}$ is a $d \times d$ diagonal matrix whose diagonal entries are the top d singular values of $\mathbf{C}$, and the corresponding left and right singular vectors are the columns of $\mathbf{U}$ and $\mathbf{V}$ respectively.

4. The matrix $\mathbf{U}$ is a dense, lower dimensional representation of the co-occurrence matrix $\mathbf{C}$. Each row of matrix $\mathbf{U}$ is a dense d dimensional vector representing the corresponding word.

At the end of this process each word type in the corpus is represented by the corresponding row of matrix $\mathbf{U}$, i. e., a word embedding of dimension d.

These learnt word embeddings can be used for further processing in several ways. In [49], the embedding space is again quantized, by means of clustering the points representing the word types into a pre-specified number of clusters based on the cosine similarity between vectors. However, this step is omitted in [31], where the word embeddings are directly used in a regression tree (CART) framework for phrase break prediction.

One issue with using LSA for deriving word embeddings arises from the fact that choosing only a small subset n of the total unique word types $|V|$ ($n << |V|$) as feature words for computing co-occurrences in some sense constrains the context, resulting in only some of the distributional characteristics being captured. A naive solution to this issue would be to choose all $|V|$ unique word types as feature words. However, this would lead to a $|V| \times 2|V|$ matrix which would lead to sparse word vectors of high dimensionality. Also, these word embeddings are not task-specific and thus are likely to be suboptimal when used directly for phrase break prediction.

As a solution to the above-mentioned issues, we proposed a neural network architecture for directly learning task-specific word embeddings for phrase break prediction. There are two advantages of directly learning task-specific word embeddings using a neural network: (i) Both feature induction and break/non-break prediction are performed in a single step as part of the same architecture, and (ii) There are no constraints placed on the context. We also used the same neural network architecture to fine-tune the LSA word embeddings for phrase break prediction.

3.2 Task specific neural word embeddings for phrase break prediction

Figure 3.1 shows our proposed architecture to directly learn word embeddings specific to the phrase break prediction. The architecture consists of an embedding matrix and a multilayer perceptron (MLP) setup as a discriminative classifier.

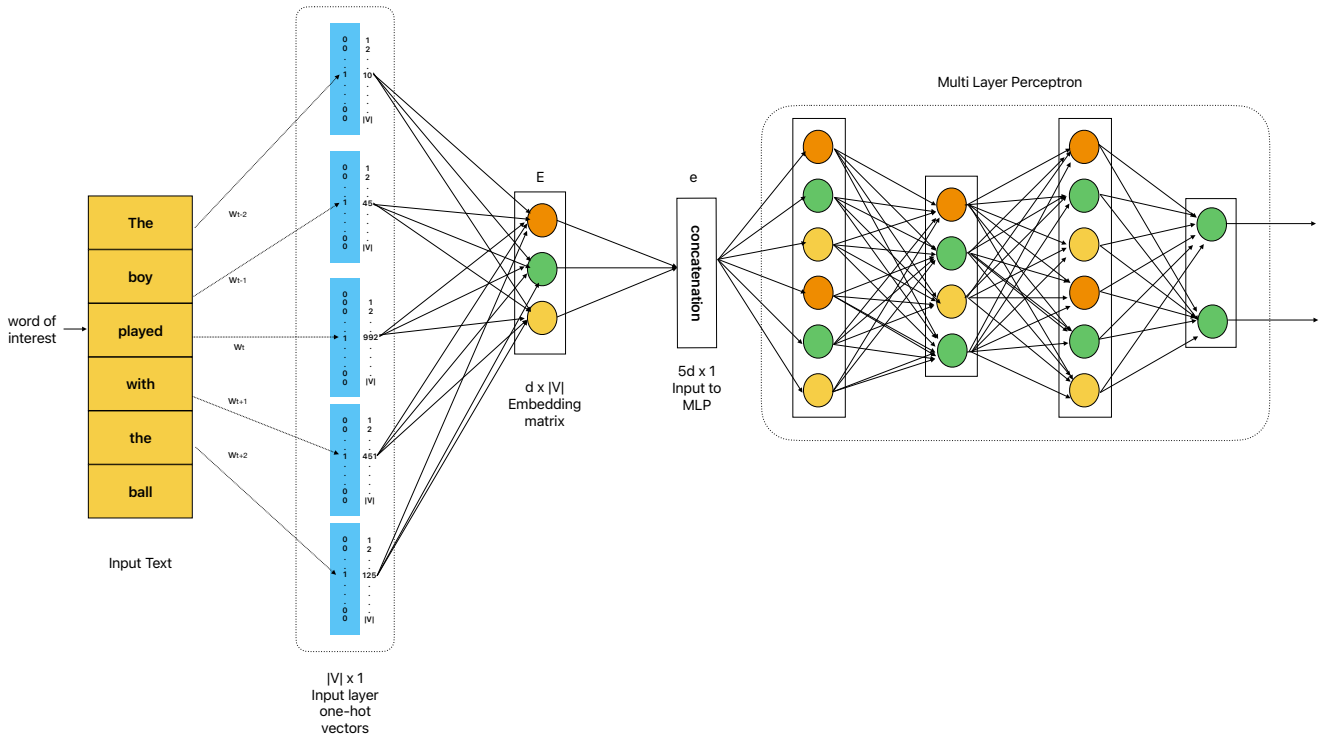


Figure 3.1 Neural network architecture to learn task-specific word embeddings for phrase break prediction

We describe our architecture in further detail below.

3.2.1 Transforming words into vectors

Each word is represented as a one-hot vector of length $|V|$, where $|V|$ is the size of the vocabulary. A one-hot vector is a vector that has one element equal to 1 (in the dimension corresponding to that particular word's index in the vocabulary), while all other elements are 0. For example, let us take the word "played" which has an index of 992 in our vocabulary. The one-hot vector representing "played" then would be

$$\begin{bmatrix} 0 & 0 & \cdots & 1 & \cdots & 0 \\ 1 & 2 & \cdots & 992 & \cdots & |V| \end{bmatrix}$$

As can be seen from the above example, the value at dimension 992 is 1 while all other values are 0.

Each of these one-hot vectors are multiplied by the embedding matrix $\mathbf{E}$, to obtain the embedding for each word. The embedding matrix $\mathbf{E}$ is of dimensionality $d \times |V|$, i. e., each column of the matrix $\mathbf{E}$ corresponds to a word in the vocabulary. Multiplying $\mathbf{E}$ by a one-hot vector that has only one non-zero element $x_t = 1$ simply selects out the relevant column for word t, resulting in an embedding vector of dimensionality d, for word t, as shown in Figure 3.2.

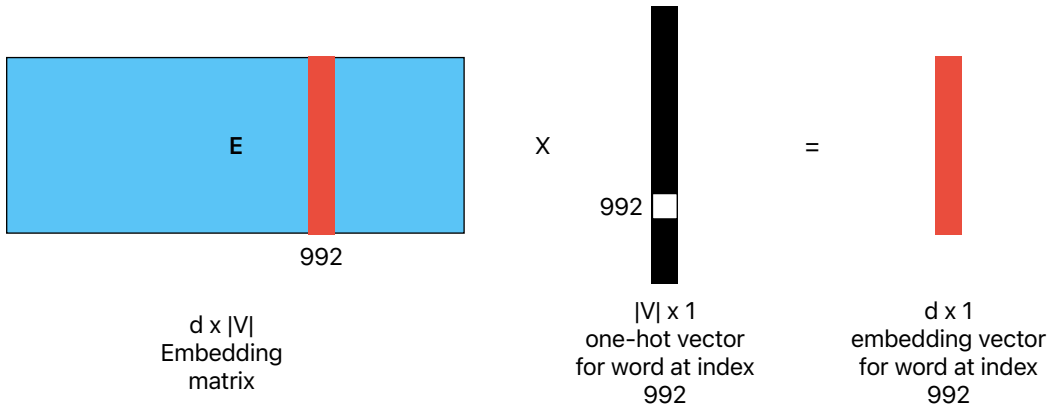


Figure 3.2 Selecting the embedding vector for word at index 992 by multiplying the embedding matrix $\mathbf{E}$ with a one-hot vector with 1 at dimension 992 and 0 everywhere else

Mathematically, this can be represented as

$$\mathbf{e_t} = \mathbf{E}\mathbf{x_t} \tag{3.1}$$

where, $\mathbf{x_t}$ is the $|V| \times 1$ one-hot column vector representing word w_t, $\mathbf{E}$ is the $d \times |V|$ embedding matrix and $\mathbf{e_t}$ is the $d \times 1$ embedding vector for the word w_t.

One point to note here is that the embedding matrix $|\mathbf{E}|$ is shared among all words in the vocabulary.

3.2.2 Incorporating context

Our model has a moving window that considers the previous two words and the next two words as context, in addition to the current word. For example, let us consider the scenario presented in Figure 3.1. Here, the word of interest (the current word) is "played" represented mathematically as w_t. The previous two words are "The" and "boy" represented mathematically as w_{t-2} and w_{t-1} respectively. The next two words are "with" and "the" represented mathematically as w_{t+1} and w_{t+2} respectively. The $|V| \times 1$ one-hot column vectors for each of these words are correspondingly $\mathbf{x_{t-2}}, \mathbf{x_{t-1}}, \mathbf{x_t}, \mathbf{x_{t+1}}$ and $\mathbf{x_{t+2}}$.

Each of these one-hot column vectors are multiplied by the $d \times |V|$ embedding matrix $\mathbf{E}$ as described in Figure 3.2 and Equation 3.1, resulting in the corresponding $d \times 1$ embedding vector for each of the words in the context window. These five embedding vectors are concatenated to produce $\mathbf{e}$, a $5d \times 1$ vector which is then used as input to the MLP.

Mathematically, this can be expressed as

$$\mathbf{e} = [\mathbf{E}\mathbf{x}_{\mathbf{t-2}}; \mathbf{E}\mathbf{x}_{\mathbf{t-1}}; \mathbf{E}\mathbf{x}_{\mathbf{t}}; \mathbf{E}\mathbf{x}_{\mathbf{t+1}}; \mathbf{E}\mathbf{x}_{\mathbf{t+2}}] \tag{3.2}$$

where the semicolon represents concatenation.

3.2.3 Training the neural network

The embedding matrix $\mathbf{E}$ is automatically updated during the neural network training. The MLP is set up as a discriminative classifier for phrase break prediction, and is trained using backpropagation. During each iteration of training, along with the weights in each layer of the MLP, the embedding matrix $\mathbf{E}$ is also updated. When training is complete, each column of the embedding matrix then represents a task-specific embedding of the corresponding word.

3.3 Experiments

3.3.1 Data used

We used speech and audio data taken from the following three audiobooks : Emma by Jane Austen (EMMA), Sense and Sensibility by Jane Austen (SAS) and Walden by Henry David Thoreau (WALDEN), in the LibriVox database[1]. Each book was recorded by a volunteer and the style of the corpus is "audiobook".

For each audiobook dataset, the speech and the corresponding text were segmented using the INTERSLICE tool [57].

3.3.1.1 Annotation of phrase breaks

We derived the location and duration of phrase breaks from the speech data, by force aligning the speech with the corresponding text prompts using the HMM tool in Festvox [43]. This gave us the locations of pauses introduced by the speaker while recording the utterances. We marked all pauses with duration $\geq$ 80 ms as a phrase break (B), and everything else as a non-break (NB).

[1] http://www.librivox.org

3.3.2 Phrase break prediction using word embeddings learnt using Latent Semantic Analysis (LSA)

We applied the procedure described in Section 3.1 to the text of EMMA, SAS and WALDEN audiobooks. For each corpus a train set and a test set was created by randomly removing 10% of the data to be used as a held out test set while the remaining 90% was used as a train set.

From each train set, the list of unique word types was compiled, giving us a vocabulary size of $|V| = 7137$ for EMMA, $|V| = 6726$ for SAS and $|V| = 10866$ for WALDEN respectively. In each corpus, the 250 most frequently occurring words were used as feature words ($n = 250$) and the value of d was set to 50. A $|V| \times 500$ raw co-occurrence matrix $\mathbf{C}$ was computed for each corpus and reduced to a $|V| \times 50$ matrix $\mathbf{U}$, using SVD.

One important issue which had to be addressed at this point, was handling unseen words during inference. We followed a similar approach to that in [58], where we took a portion of the train set (we used 100 infrequent words which occured in the train set only once) and rewrote them using a special unseen token. Word embeddings were then learnt for all the words in the train set (including the special unseen token) using the procedure described in Section 3.1. During inference, all unseen words were mapped to this special token and were represented by the corresponding word embedding.

3.3.3 Phrase break prediction using task specific neural word embeddings

We trained the neural network architecture described in Section 3.2 on the text of EMMA, SAS and WALDEN audiobooks in order to learn word embeddings specific to phrase break prediction. We used the same train/test split as in Section 3.3.2. The vocabulary size of each corpus was the same as in Section 3.3.2 ($|V| = 7137$ for EMMA, $|V| = 6726$ for SAS and $|V| = 10866$ for WALDEN respectively), and a randomly initialized $50 \times |V|$ matrix was used as embedding matrix in all three cases. We also experimented with initializing the embedding matrix with the word embeddings learnt using LSA described in Section 3.3.2.

Our model has a moving window that considers the previous two words and the next two words as context, in addition to the current word as described in Section 3.2.2. Each word in the context window is represented by a 50×1 embedding vector. All the embedding vectors corresponding to the five words in the context window were concatenated to generate a 250×1 vector which was the input to the MLP classifier.

The architecture of the MLP classifier was 250L 500N 50N 500N 2S, where L represents "linear" activation, N represents "$tanh()$" activation and S represents "$sigmoid()$" activation. To handle unseen words during inference we used the same method described in Section 3.3.2.

We also explored combining these neural word embeddings with additional linguistic features (like POS tags) from other sources. We hypothesized that these additional linguistic features would add complementary information to the task-specific features captured during the neural network training.

3.3.4 Systems built

1. **System CL : LSA word embeddings used directly for phrase break prediction in a CART model** System CL is a regression tree (CART) model, where the predictee is the phrase break index (B/NB) of each word and the predictors are the word embeddings of that particular word learnt using LSA. In order to include context, we concatenated the word embeddings of the previous two words and the next two words with the word embedding of the word under question, giving us a 250 dimensional feature vector in each case.

2. **System NL : LSA word embeddings used directly for phrase break prediction in a Neural network model** System NL is a neural network trained to be a discriminative classifier. The system is trained using backpropagation and outputs a B/NB decision given the features of the word as input. In this case, the input features are the word embeddings of that particular word learnt using LSA. As in System CL, we concatenated the word embeddings of the previous two words and the next two words with the word embedding of the word under question, in order to incorporate context. This resulted in an input vector of size 250, which was the input to the neural network classifier. The architecture of the neural network classifier was 250L 500N 50N 500N 2S, where L represents "linear" activation, N represents "tangential $tanh()$" activation and S represents "$sigmoid()$" activation.

3. **System RE : Task specific neural word embeddings for phrase break prediction with a randomly initialized embedding matrix** System RE is the neural network architecture described in Section 3.2 which uses a randomly initialized embedding matrix to embed words in a 50 dimensional space. This randomly intialized embedding matrix is then updated during the neural network training as described in Section 3.2.3. On completion of training, each column of the embedding matrix is a task-specific embedding of the corresponding word in a 50 dimensional space. During inference, these task-specific embeddings are used by the neural network to generate the B/NB decision for each word in the input sentence.

4. **System LE : Task specific neural word embeddings for phrase break prediction with an embedding matrix initialized using LSA word embeddings** System LE is the same as System RE, except for the fact that System LE uses an embedding matrix initialized with the word embeddings learnt using LSA described in Section 3.1, as opposed to a randomly initialized embedding matrix in System RE. This embedding matrix is then updated during neural network training as described in Section 3.2.3, resulting in the LSA embeddings being refined/fine-tuned for phrase break prediction. On completion of training, each column of the embedding matrix is a task-specific embedding of the corresponding word in a 50 dimensional space. During inference, these task-specific embeddings are used by the neural network to generate the B/NB decision for each word in the input sentence.

5. **System REP : Task specific neural word embeddings for phrase break prediction with a randomly initialized embedding matrix + POS features** System REP is the same as System RE except that in System REP the POS tag of each word is encoded using a real valued 5 dimensional vector, which is then concatenated with the 50 dimensional neural word embedding for that particular word. As a result, each word in the corpus is embedded in a 55 dimensional space. This embedding matrix is then updated during the neural network training as described in Section 3.2.3. On completion of training, each column of the embedding matrix is an embedding of the corresponding word in a 55 dimensional space, which encodes both task-specific features and some additional linguistic information. During inference, these embeddings are used by the neural network to generate the B/NB decision for each word in the input sentence.

6. **System LEP : Task specific neural word embeddings for phrase break prediction with an embedding matrix initialized using LSA word embeddings + POS features** System LEP is the same as System LE with the addition of the POS information of the word being included in embeddings. As in System REP, we encode each POS tag by a real valued 5 dimensional vector, which is concatenated with the 50 dimensional LSA word embedding for that particular word. As a result, each word in the corpus is embedded in a 55 dimensional space. This embedding matrix is then updated during the neural network training as described in Section 3.2.3, resulting in the LSA embeddings being refined/fine-tuned for phrase break prediction. On completion of training, each column of the embedding matrix is an embedding of the corresponding word in a 55 dimensional space, which encodes both task-specific features and some additional linguistic information. During inference, these embeddings are used by the neural network to generate the B/NB decision for each word in the input sentence.

3.4 Results

Table 3.1 shows the performance of all six systems described in Section 3.3.4, on the phrase break prediction task. We report our results in terms of the F-Measure [45] which is defined as the harmonic mean of precision and recall. F-Measure values range from 0 to 1, with higher values indicating better performance.

We analyze the results in detail below.

1. **System CL vs System NL** System CL and System NL show similar performance on the phrase break prediction task. This shows us that the choice of machine learning technique does not have an effect on phrase break prediction performance when using word embeddings learnt using LSA. We use System CL as our baseline system in all further analysis.

2. **System RE vs System CL** An observation of the results obtained for System RE and System CL, shows that there is an improvement in the phrase break prediction performance when using

Table 3.1 Performance (in terms of the F-Measure) of the various systems on the phrase break prediction task

Audiobook	**LSA Word Emebddings**		**Task specific neural word embeddings**			
	System CL	System NL	System RE	System LE	System REP	System LEP
EMMA	0.65	0.64	0.68	0.70	0.71	0.72
SAS	0.64	0.64	0.67	0.69	0.70	0.71
WALDEN	0.64	0.63	0.67	0.68	0.68	0.70

task-specific neural word embeddings as opposed to word embeddings learnt using LSA. This shows that directly modeling task-specific features in order to construct the embedding space, thus learning task-specific word embeddings gives better performance on phrase break prediction.

3. **System LE vs System RE** A comparison of the performance of System LE and System RE shows that System LE performs better than System RE on phrase break prediction. This performance improvement can probably be explained by the fact that when the embedding matrix is initialized with word embeddings learnt using LSA which are then fine-tuned, the initial word embeddings already have some semantic information captured and thus result in better task-specific word embeddings after fine-tuning as opposed to when the embedding matrix is randomly initialized.

4. **System REP vs System RE** System REP performs better than System RE on the phrase break prediction task. This validates our hypothesis that encoding additional linguistic features (like POS tags) along with the embedding matrix, results in complementary information being added to the task-specific features captured during the neural network training. This results in a better performance on the phrase break prediction task.

5. **System LEP vs System CL** An analysis of the performance of our final system, System LEP, shows an absolute increase of 0.07 (for EMMA and SAS) and 0.06 (for WALDEN) in the F-Measure values, as compared to the baseline (System CL).

3.5 Summary

In this chapter, we presented our work on learning continuous valued word representations, i. e. representing words as points in a continuous dimensional space, also called word embeddings, for phrase break prediction. We presented two methods for learning word embeddings, the first learnt word embeddings using a matrix factorization method based on latent semantic analysis (LSA), while the second was a neural network architecture for learning task-specific neural word embeddings.

LSA is a traditional method for generating word embeddings that uses a matrix factorization technique to identify semantic relationships between words in a corpus. The resulting embeddings capture the relationships between words based on the co-occurrence of words in the corpus. LSA word embeddings are not task-specific and can be used for a wide range of NLP tasks, including phrase break prediction. On the other hand, task-specific neural word embeddings are generated using neural network architectures that are specifically designed for the task at hand. In our particular case, the neural network was trained on a dataset of text and corresponding phrase break annotations to learn the appropriate embeddings for phrase break prediction. The advantage of directly learning task-specific neural word embeddings is that the network architecture combines feature induction and phrase break prediction in a single framework, and the learnt word embeddings would be directly optimized for the task in question.

We trained several systems using both LSA word embeddings and task-specific neural word embeddings, on audiobook data for phrase break prediction. We evaluated their performance and showed that task-specific neural word embeddings have the advantage of being specifically optimized for phrase break prediction and thus outperform LSA embeddings when the LSA embeddings were directly used in a machine learning framework. However, when LSA embeddings were fine-tuned on phrase break prediction using our task-specific architecture their performance was comparable to the task-specific neural word embeddings. Finally, we show that encoding additional linguistic features (like POS tags) along with the word embeddings give the best performance on phrase break prediction.

Chapter 4

Phrase break prediction using Recurrent Neural Networks

Language, both spoken and written, is inherently temporal in nature; i. e., it can be modeled as a sequence that unfolds in time. Some traditional models of language processing, such as Hidden Markov models (HMMs) take into account this temporal nature and proceed through the input one step at a time, carrying forward information learnt along the way. Other models of language processing do not have this temporal nature, and assume simultaneous access to all aspects of their input.

With the success of deep learning in speech recognition [59], deep learning techniques began to be applied to other areas of spoken language processing, such as language modeling, Text-to-Speech (TTS) synthesis, intent determination, sentiment detection/analysis, semantic utterance classification in spoken language understanding etc. However, while deep neural networks (DNNs) are able to effectively capture dependencies across features, they lack the ability to capture long-term relations that are spread across time. Limited temporal context can be captured in DNNs by using a moving fixed-size context window over the input. However, this is suboptimal as the network at any given point in time does not have access to any input context outside the fixed window.

On the other hand, recurrent neural networks (RNNs) have a mechanism to account for the temporal nature of language without the use of arbitrary fixed-size context windows as is the case with DNNs. The recurrent connections in RNNs allow the model to take into account context from the beginning of the sequence while making a decision.

In this work, we present work on using RNNs for phrase break prediction. This work has primarily been motivated by the following question: "Does capturing long-term temporal context using a recurrent architecture result in a better performance on phrase break prediction, as compared to the limited temporal context captured using a moving fixed-size window as in the case of DNNs?" We experiment our approach on audiobook data and attempt to answer the question posed above.

The rest of this chapter is organized as follows: Section 4.1 presents a discussion on common RNN architectures, Section 4.2 describes RNNs for phrase break prediction, Section 4.3 presents our experimental results with a discussion on the same and Section 4.4 summarizes the chapter.

4.1 RNN Architectures

In this work, we explore two common RNN architectures: (i) Simple RNN, also known as Elman RNN [60], and (ii) Long Short Term Memory (LSTM) [61]. We describe both RNN architectures below.

4.1.1 Simple RNN

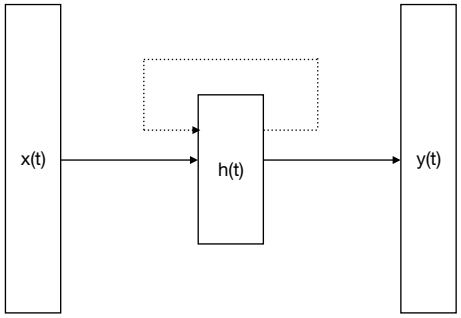


Figure 4.1 Simple RNN architecture. The hidden layer includes a recurrent connection as part of its input. The hidden layer state at timestep t depends upon both the current timestep input and the hidden layer state at the previous timestep. $x(t), y(t)$ and $h(t)$ are the inputs, outputs and the hidden layer state at timestep t respectively

Figure 4.1 illustrates the structure of a simple RNN. The hidden layer state from the previous timestep provides a form of memory or context, that encodes inputs from earlier timesteps and informs decisions at later timesteps. Most importantly, this approach does not impose a limit on the context, and thus the RNN can use information extending back to the beginning of the sequence.

The dynamics of a simple RNN are similar to that of a feedforward network, with the addition of recurrence weights which connect the hidden layer from the previous timestep to the hidden layer at the current timestep as shown in Figure 4.2

Mathematically, the dynamics of a simple RNN can be expressed as

$$h(t) = g(\mathbf{U}h(t-1) + \mathbf{W}x(t)) \tag{4.1}$$

$$y(t) = f(\mathbf{V}h(t)) \tag{4.2}$$

where, $\mathbf{W} \in \mathbb{R}^{d_h \times d_{in}}$, $\mathbf{U} \in \mathbb{R}^{d_h \times d_h}$, $\mathbf{V} \in \mathbb{R}^{d_{out} \times d_h}$, d_{in}, d_h, d_{out} are the dimensions of the input, hidden and output layers respectively. g and f are the non-linear activation functions of the hidden and the output layers respectively.

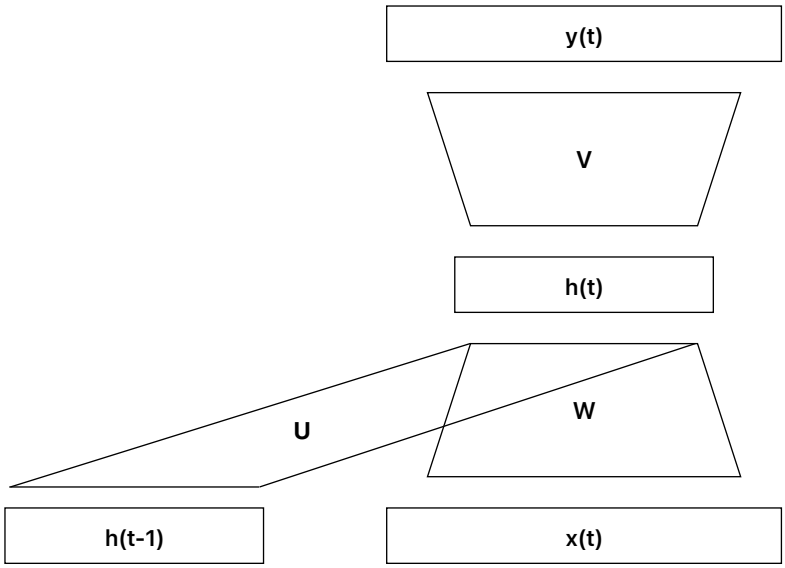


Figure 4.2 Simple RNN illustrated as a feedforward network. $\mathbf{W}$ represents the input layer to hidden layer weights, $\mathbf{U}$ represents the recurrence weights connecting the previous timestep hidden layer with the current timestep hidden layer, $\mathbf{V}$ represents the hidden layer to output layer weights, $h(t-1)$ is the hidden layer state at timestep $t-1$, $h(t)$ is hidden layer state at timestep t, $x(t)$ is input at timestep t and $y(t)$ is the output at timestep t

As can be seen from the above equations, the computation of output at timestep t requires the hidden layer state at timestep $t-1$. The algorithm for the forward pass in a simple RNN thus proceeds incrementally from the start of the sequence to the end as illustrated in Algorithm 1, taken from [62].

Algorithm 1: Forward pass algorithm for a Simple RNN

Input: Input sequence $x = \{x(1), x(2), \cdots, x(t-1), x(t)\}$

Output: Output sequence $y = \{y(1), y(2), \cdots, y(t-1), y(t)\}$

$h(0) \leftarrow 0$

for $i \leftarrow 1$ ***to*** $length(x)$ **do**

$h(i) \leftarrow g(\mathbf{U}h(i-1) + \mathbf{W}x(i))$

$y(i) \leftarrow f(\mathbf{V}h(i))$

end

The sequential nature of simple RNNs can be seen by unrolling the network in time as shown in Figure 4.3

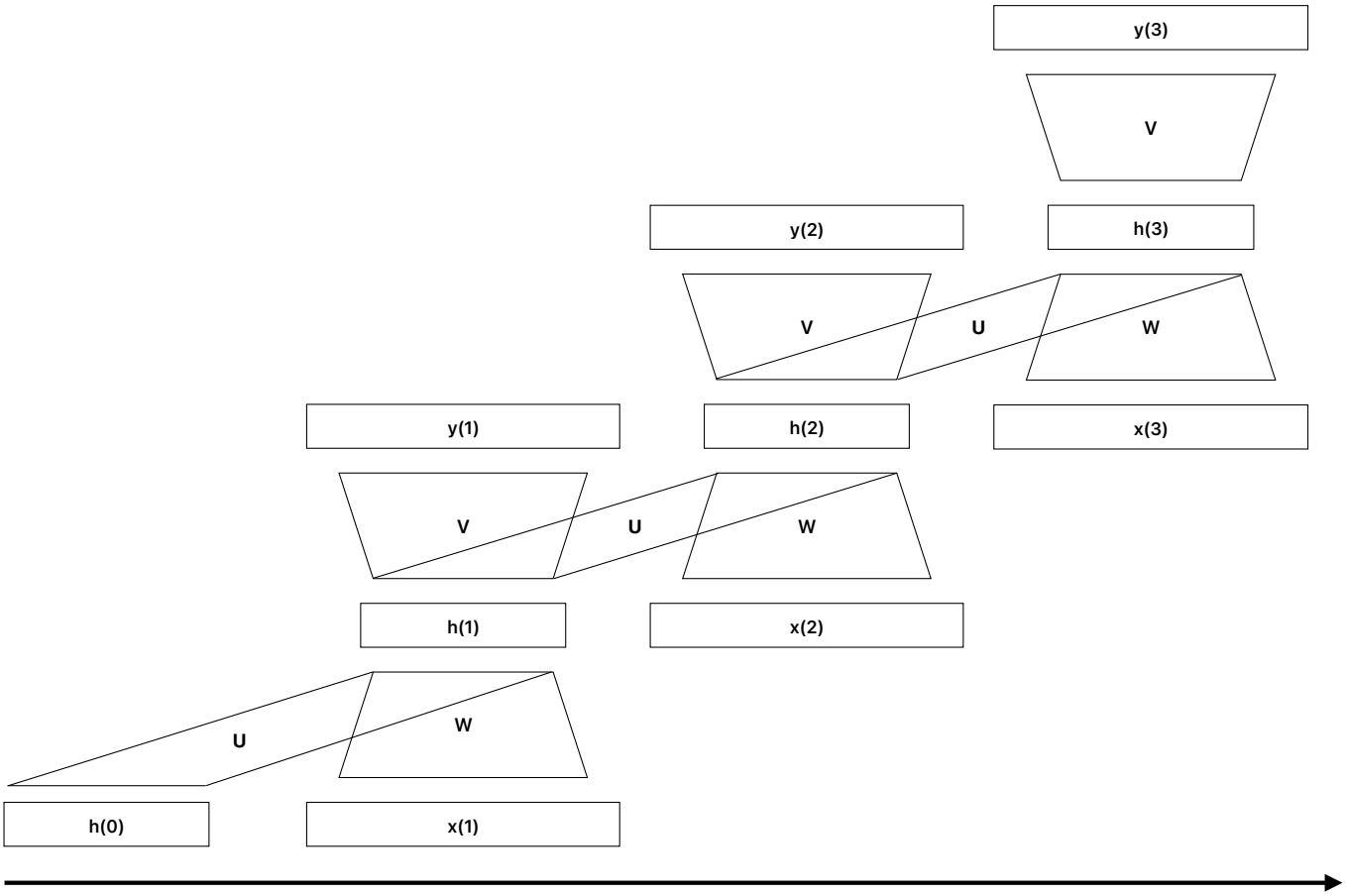


Figure 4.3 A simple RNN shown unrolled in time. Weights $\mathbf{U}$, $\mathbf{W}$ and $\mathbf{V}$ are shared across all timesteps while h and y values are computed for each timestep

Training an RNN using gradient descent is similar to training a feedforward network, with backpropagation being used to compute the gradients required to update $\mathbf{U}$, $\mathbf{W}$ and $\mathbf{V}$. However, there are two factors specific to RNNs which need to be considered:

1. Computation of the loss at a particular timestep t requires the hidden layer state at timestep $t-1$
2. The hidden layer state at timestep t influences the output at timestep t as well as the hidden layer state at timestep $t+1$ (and hence indirectly the output and loss at timestep $t+1$)

Tailoring the backpropagation algorithm to account for these factors results in a two-pass algorithm for computing gradients in RNNs called backpropagation through time (BPTT) [63, 64].

As shown in Figure 4.3 explicitly unrolling a simple RNN into a feedforward computational graph allows the network weights $\mathbf{U}$, $\mathbf{W}$ and $\mathbf{V}$ to be trained directly from data. For applications with very

long input sequences, unrolling the entire sequence may not be computationally feasible. In such cases, the RNN can be unrolled into multiple mangable fixed-length sequences which are treated as distinct training items.

RNNs can also be stacked together in layers, wherein the entire output sequence from one layer is used as input to the subsequent layer. A bidirectional RNN [65] combines two independent RNNs, one where the input is processed from start to end, and the other where the input is processed in the reverse direction i. e., from end to start. The representations generated by these two RNNs are then concatenated into a single representation that captures both left and right contexts of an input at each point in time. Both stacked and bidirectional RNNs outperform single-layer and unidirectional RNNs, however this improvement comes with increased costs of training.

4.1.2 Long Short Term Memory (LSTM)

One of the advantages of simple RNNs over DNNs is that simple RNNs have a mechanism to account for the temporal nature of language without the use of arbitrary fixed-size context windows. The recurrent connections in simple RNNs allow the model to take into account context from the beginning of the sequence while making a decision. However, in practice it is very difficult to train simple RNNs on tasks that require the network to use information distant from the current point of processing. The primary reason being, the network weights (which inform the hidden layer state at the current timestep) have to perform two tasks simultaneously: (i) Provide contextual information to make the decision at the current timestep, and (ii) Update and carry forward contextual information required for decisions at future timesteps.

Another issue with training simple RNNs on long sequences is the vanishing gradient problem. This problem arises from the need to backpropagate the error signal back through time. As explained in Section 4.1.1 the hidden layer state at timestep t contributes to the loss at timestep $t + 1$. As a result, during the backward pass, the hidden layer gradients are subjected to repeated multiplications, determined by the length of the sequence. A frequent result of this process is that the gradients are driven to zero, resulting in no learning taking place.

To address these issues, more complex network architectures were designed that explicitly manage the task of maintaining relevant context over time. One such popular architecture is the long short term memory (LSTM) [61] network. LSTMs address the content management problem in two ways: (i) By the addition of an explicit context layer to the LSTM architecture (in addition to the usual recurrent hidden layer), and (ii) Through the use of specialized neural units that make use of gates to control the flow of information into and out of the units that comprise the network layers. Figure 4.4 illustrates the LSTM architecture.

The gates in the LSTM architecture are implemented through the use of additional weights that operate sequentially on the current timestep input, previous timestep hidden layer and previous timestep context layer respectively. Each gate consists of a feedforward layer, followed by a sigmoid activation function, followed by a pointwise multiplication with the layer being gated. Combining the sigmoid

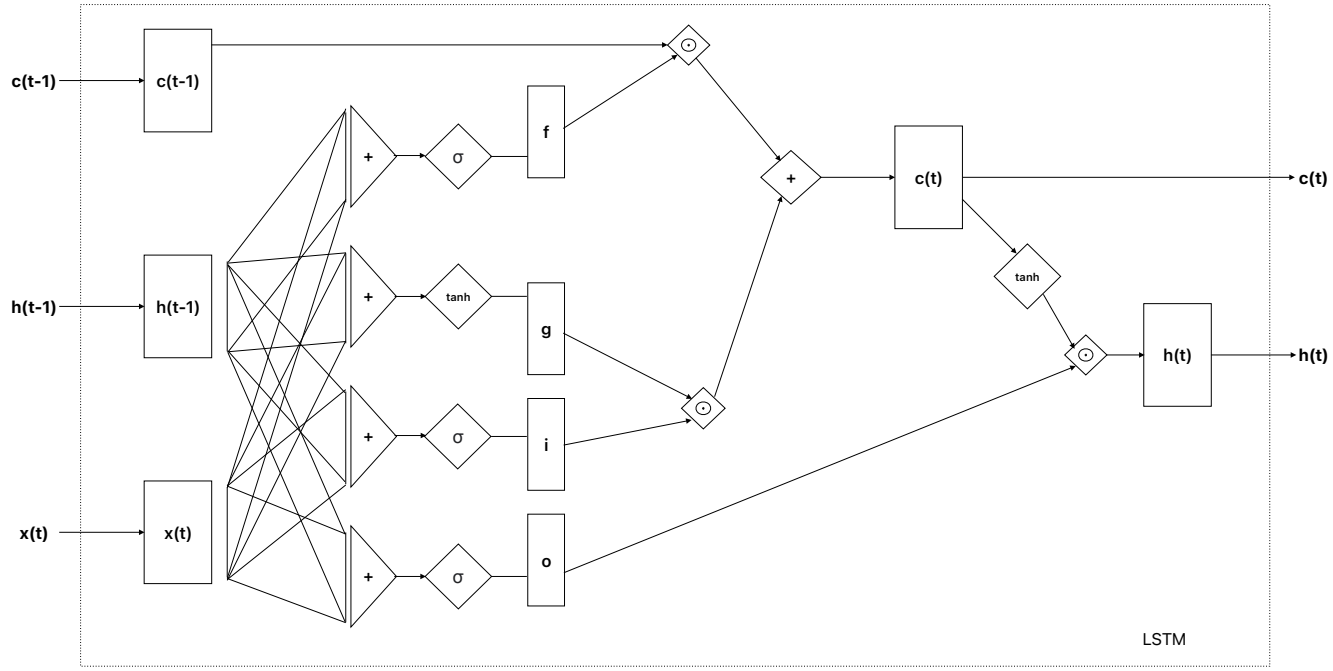


Figure 4.4 LSTM architecture. Inputs to the LSTM are the current timestep input $x(t)$, the previous timestep hidden layer state $h(t-1)$ and the previous timestep context $c(t-1)$. Outputs of the LSTM are the hidden layer state for the current timestep $h(t)$ and the updated context $c(t)$

activation with pointwise multipication has the same effect as a binary mask, wherein values in the layer being gated that align with 1's in the mask are passed thorough unchanged while other values are erased. The LSTM architecture uses three gates: (i) Forget gate, which deletes contextual information which is no longer needed, (ii) Add gate, which selectes information to add to the current timestep context, and (iii) Output gate, which decides what information is required for the current hidden state (as opposed to what contextual information needs to be preserved for future decisions).

Mathematically the dynamics of a LSTM can be expressed as

$$f(t) = \sigma(\mathbf{U}_f h(t-1) + \mathbf{W}_f x(t)) \tag{4.3}$$
$$k(t) = c(t-1) \odot f(t) \tag{4.4}$$
$$g(t) = tanh(\mathbf{U}_g h(t-1) + \mathbf{W}_g x(t)) \tag{4.5}$$
$$i(t) = \sigma(\mathbf{U}_i h(t-1) + \mathbf{W}_i x(t)) \tag{4.6}$$
$$j(t) = g(t) \odot i(t) \tag{4.7}$$
$$c(t) = j(t) + k(t) \tag{4.8}$$
$$o(t) = \sigma(\mathbf{U}_o h(t-1) + \mathbf{W}_o x(t)) \tag{4.9}$$
$$h(t) = o(t) \odot tanh(c(t)) \tag{4.10}$$

where, $\mathbf{U}_g$ is the recurrence weight connecting the previous timestep hidden layer with the current timestep hidden layer, $\mathbf{W}_g$ is the weight connecting the current timestep input with the current timestep hidden layer, $\mathbf{U}_f$, $\mathbf{U}_i$, $\mathbf{U}_o$ are the weights connecting the previous timestep hidden layer with the forget gate, add gate and output gate respectively, $\mathbf{W}_f$, $\mathbf{W}_i$, $\mathbf{W}_o$ are the weights connecting the current timestep input with the forget gate, add gate and output gate respectively, $h(t-1)$ is the hidden layer state at timestep $t-1$, $c(t-1)$ is the context at timestep $t-1$, $x(t)$ is the input at timestep t, $h(t)$ is hidden layer state at timestep t and $c(t)$ is the updated context at timestep t.

Similar to simple RNNs, LSTMs can also be stacked together in layers, wherein the entire output sequence from one layer is used as input to the subsequent layer. Bidirectional LSTMs combines two independent LSTMs, one where the input is processed from start to end, and the other where the input is processed in the reverse direction i. e. from end to start. The representations generated by these two LSTMs are then concatenated into a single representation that captures both left and right contexts of an input at each point in time. As in the case with simple RNNs, both stacked and bidirectional LSTMs outperform single-layer and unidirectional LSTMs, however this improvement comes with increased costs of training.

4.2 RNNs for phrase break prediction

In this work, we model the phrase break prediction task as a sequence labeling task. Given a sequence of words, we train a RNN model to label each word boundary with either B or NB indicating whether a

break should be inserted after that word or not. The inputs to the model are word embeddings which are randomly initialized and jointly trained along with the RNN model on the task at hand, and the outputs are probabilities generated by a softmax layer over the set of possible tags (B and NB in this case). Figure 4.5 shows our model architecture.

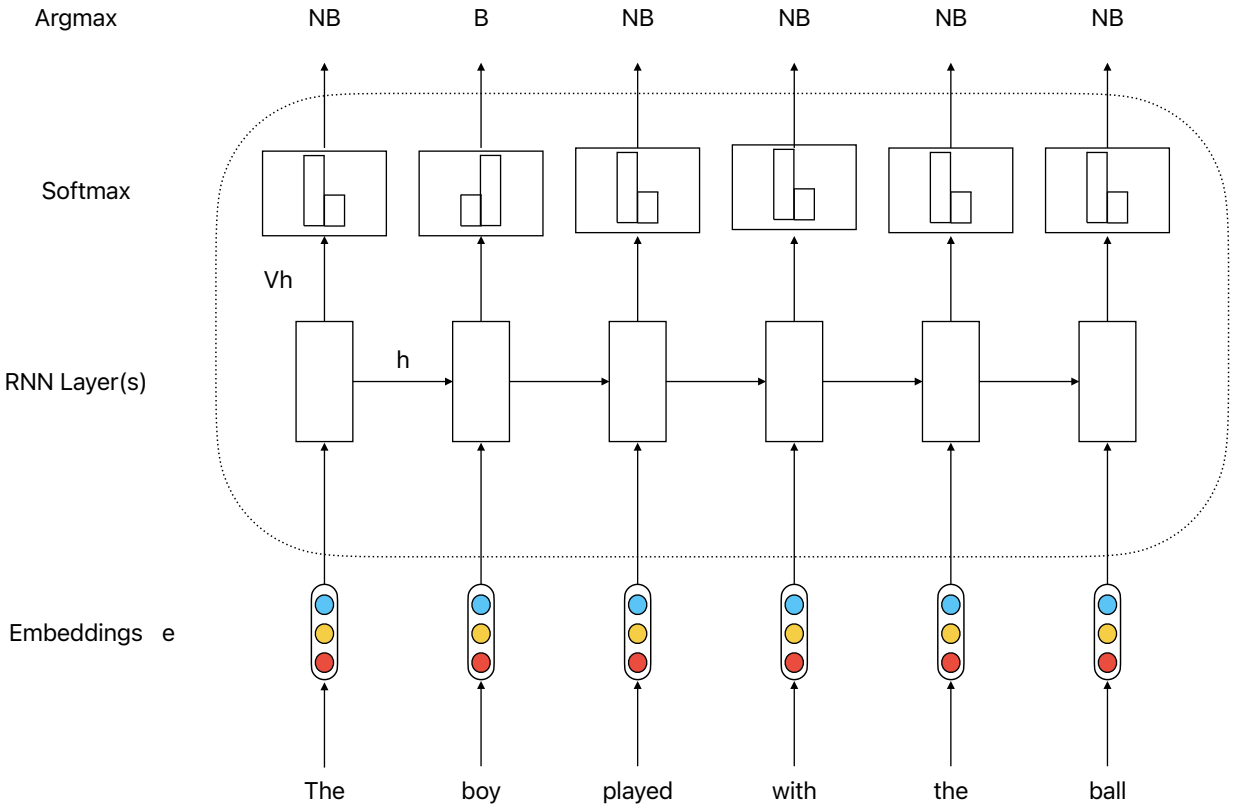


Figure 4.5 Phrase break prediction as a sequence labeling task using a RNN. Inputs are word embeddings which are randomly intialized and jointly trained along with the RNN model on task at hand, outputs are probabilities from a softmax layer over the set of possible tags (B and NB)

To generate a sequence of break indices for a given input, we run the forward pass over the input sequence and select the most likely break index (B or NB) from the softmax layer at each step. The RNN is trained using cross-entropy loss. We experiment with both simple RNNs and LSTMs for phrase break prediction. We also train a DNN model to use as a baseline, and compare the performance of RNN models with the DNN model on phrase break prediction. Table 4.1 shows the parameter as well as

hyperparameter values used in our models. All the code as well as data used in this work are available online [1].

Table 4.1 Parameter and hyperparameter values used in our models

Parameters	DNN	Simple RNN	LSTM
word embedding dimension	50	50	50
hidden layer size	200 : 200	200	200
activation function	$tanh()$	$tanh()$	$tanh()$
batch size	32	1	1
initial learning rate	0.01	0.01	0.01
nesterov momentum	0.9	0.3	0.3
weight initialization	glorot uniform [66]	glorot uniform [66]	glorot uniform [66]
inner cells initialization	–	orthogonal [67]	orthogonal [67]
forget gate bias	–	–	1

4.3 Experimental results

4.3.1 Data used

We use speech and audio data taken from the following three audiobooks : Emma by Jane Austen (EM), Mansfield Park by Jane Austen (MP) and Pride and Prejudice by Jane Austen (PAP), in the LibriVox database[2]. Each book was recorded by a volunteer and the style of the corpus is "audiobook".

For each audiobook dataset, the speech and the corresponding text were segmented using the INTERSLICE tool [57].

4.3.1.1 Annotation of phrase breaks

As we do not have a corpus with hand annotated phrase breaks, we derive the location and duration of the phrase breaks from the speech data. In order to derive the locations and durations of pauses introduced by the speaker, we force align the speech with the corresponding text prompts using the HMM tool in Festvox [43]. This gives us the locations of pauses introduced by the speaker while recording the utterances. We mark all pauses with duration $\geq$ 80 ms as a phrase break (B), and everything else as a non-break (NB).

[1] `https://github.com/anandaswarup/phrase_break_prediction_IS16`
[2] `http://www.librivox.org`

4.3.2 Experiment 1 : DNNs vs RNNs

In this experiment, we compare the performances of DNN, Elman RNN and LSTM models on the phrase break prediction task. We used the Keras[3] deep learning library, running the Theano [68, 69] backend to build our models.

Table 4.2 shows the results of the DNN, RNN and LSTM systems on the phrase break prediction task. We report our results in terms of the F-Measure [45] which is defined as the harmonic mean of precision and recall. F-measure values range from 0 to 1, with higher values indicating better performance.

Table 4.2 Performance (in terms of the F-Measure) of the DNN, RNN and LSTM systems on the phrase break prediction task

	F-Measure		
	EM	MP	PAP
DNN	0.86	0.86	0.86
RNN	0.92	0.92	0.93
LSTM	0.93	0.92	0.93

An examination of Table 4.2 shows that both the Elman RNN and LSTM models outperform the DNN model, while there is no difference in the performance of the Simple RNN as compared to the LSTM model. This experimentally proves that capturing long-term temporal dependencies using a recurrent architecture improves the performance of phrase break prediction, as compared to a DNN architecture, and answers the question raised at the beginning of the chapter.

4.3.3 Experiment 2 : Effect of varying the size of the RNN hidden layer

In this experiment we study the effect of varying the size of the RNN hidden layer on the performance of the model for phrase break prediction. We use the Simple RNN model from Experiment 1 (Section 4.3.2) with all parameter and hyperparameter values unchanged, except for the hidden (recurrent) layer size. We vary the hidden layer size and compute the performance, in terms of the F-measure. Table 4.3 shows the results of this experiment.

An examination of the results in Table 4.3 shows that there is not much difference in the performance on phrase break prediction when the hidden layer size is varied.

[3] `http://keras.io`

Table 4.3 Effect of varying the hidden (recurrent) layer size on the performance (in terms of the F-measure) of the Simple RNN model on the phrase break prediction task

	F-Measure		
Hidden layer size	EM	MP	PAP
200	0.92	0.92	0.93
500	0.92	0.92	0.92
1000	0.93	0.92	0.93

4.3.4 Experiment 3 : Effect of varying the dimension of word embeddings

In this experiment we study the effect of varying the word embedding dimension on the performance of the model for phrase break prediction. We use the Simple RNN model from Experiment 1 (Section 4.3.2) with all the parameters and hyperparemeters unchanged, except for the word embedding dimension. We vary the word embedding dimension and compute the performance, in terms of the F-measure. Table 4.4 shows the results of this experiment.

Table 4.4 Effect of varying the word embedding dimension on the performance (in terms of the F-measure) of the Simple RNN model on the phrase break prediction task

	F-Measure		
Word embedding dimension	EM	MP	PAP
50	0.92	0.92	0.92
100	0.93	0.93	0.93
150	0.93	0.92	0.93
200	0.93	0.93	0.93

An examination of the results in Table 4.4 shows that there is not much difference in the performance on phrase break prediction when the word embedding dimension is varied.

4.4 Summary

The work presented in this chapter was primarily motivated by the following question: "Does capturing long-term temporal context using a recurrent architecture result in a better performance on phrase break prediction, as compared to the limited temporal context captured using a moving fixed-size window as in the case of DNNs?"

We modeled phrase break prediction as a sequence labeling task. Given a sequence of words as input, we trained RNN models to label each word boundary with either B or NB indicating whether a break should be inserted after that word or not. Inputs to the RNN models were randomly initialized word embeddings which were jointly trained along with the RNN models on the task at hand. Outputs of the RNN models were probabilities generated by a softmax layer over the set of possible tags (B and NB, in this case). The RNN models were trained using cross-entropy loss. During inference, to generate a sequence of break indices for a given text input, a greedy approach was followed. We first ran the forward pass of the RNN model on the input sequence, and selected the most likely break index (B or NB) for each input token by applying an argmax over the probabilities generated by the softmax layer corresponding to that particular token.

We trained DNNs, Simple RNNs and LSTMs on audiobook data for phrase break prediction, and performed three experiments. In the first experiment, we used the DNN model as a baseline, and compared the performance of both RNN models with the DNN model. Experimental results showed that both RNN models outperformed the DNN model while there was not much difference in the performance among the two RNN models. In the remaining two experiments, we attempted to study the effect of varying the size of the RNN hidden layer as well as the effect of varying the input word embedding dimension, on the performance of the Simple RNN model. Results from these experiments showed that, varying either the RNN hidden layer size or the input word embedding dimension did not seem to have much of an effect on the performance of the Simple RNN model.

Chapter 5

Phrase break prediction using BERT

Word embeddings i. e., representing words as points in a continuous dimensional space, have been used in various NLP applications. Learning these word embeddings typically entails learning a transformation for each word in a raw text corpus to a dense lower dimensional vector space. Most word embedding methods learn word representations with the distributional hypothesis [47, 48], i. e., words with similar context are expected to have similar meanings. Traditionally, there existed two broad approaches to learning word embeddings: (i) Matrix factorization methods [49, 50], and (ii) Neural methods [51–53]. Methods were also developed to optimize these word embeddings to a specific task, either by fine-tuning the word embeddings to capture task specific features in addition to semantic features captured using word contexts, or by directly modeling task specific features in order to construct the embedding space thus learning task-specific word embeddings [54–56, 70, 71]. All the techniques described above, generate static or context-free word embeddings, meaning that for each word in the vocabulary, one fixed embedding is learnt for all possible contexts in which that particular word appears in the text corpus. While static embeddings have proved to be effective in NLP applications, they do not take into account the fact that words have different meanings in different contexts. For example, the English word *bank* can refer to the side of a river as in *river bank* or to a financial institution as in *bank account*. Static word embedding techniques would learn the same word embedding in both cases.

To address this issue, context-sensitive word representations, also called dynamic or contextual word embeddings were developed. ELMo (Embeddings from Language Models) [72], learned contextual word embeddings by combining the internal states of a bidirectional LSTM trained with a coupled language model objective on a large text corpus. The weights of the bidirectional LSTM were then frozen, and these contextual word embeddings were used in a downstream task's existing supervised model as additional input features augmenting the static word embeddings originally used as input to the model, and the supervised model was trained on the task at hand. Use of ELMo embeddings improved the state of the art across six NLP applications: sentiment analysis, natural language inference, semantic role labeling, coreference resolution, named entity recognition and question answering.

Although ELMo embeddings improved the state of the art in a diverse set of NLP applications, each application made use of a task-specific architecture. The GPT (Generative Pre-Training) model [73]

was an attempt to design a general task-agnostic model for learning contextual word embeddings. The GPT model used a transformer decoder trained with a language modeling objective on unlabeled text data, to learn contextual word embeddings. These contextual word embeddings were then adapted to a target task using the corresponding supervised objective. Unlike ELMo, where the parameters of the pretrained language model were frozen, the GPT model fine-tuned all the parameters in the pretrained transformer decoder during supervised learning of the downstream task. The GPT model improved the state of the art in nine NLP tasks, with minimal change in the model architecture. However, due to the autoregressive nature of transformer decoders, GPT could only encode context left-to-right.

As can be seen from the discussion above, ELMo encoded context bidirectionally but used a task-specific architecture, while GPT was task-agnostic but could only encode context left-to-right. BERT (Bidirectional Encoder Representations from Transformers) [74] addressed both issues, with a model which encoded context bidirectionally while requiring minimal architectural changes for a wide range of NLP tasks. BERT used a transformer encoder trained with a language modeling objective on vast amounts of unlabeled text data, to represent any token based on its bidirectional context, which were then fed to an additional task-specific output layer and fine-tuned on the task at hand. The addition of only a task-specific output layer during fine-tuning meant minimal changes in architecture. All the parameters of the pretrained transformer encoder were fine-tuned, while the additional task-specific output layer was trained from scratch during fine-tuning on the downstream task.

In this chapter, we present our work on phrase break prediction using bidirectional encoder representations. We present a method of fine-tuning a pretrained BERT model, with an additional token classification layer, on phrase break prediction. The rest of this chapter is organized as follows: Section 5.1 presents a discussion on the BERT model, Section 5.2 presents our experimental setup along with results and Section 5.3 summarizes the chapter.

5.1 Bidirectional Encoder Representations from Transformers (BERT)

BERT [74] is designed to pretrain deep bidirectional representations from large amounts of unlabeled text by jointly conditioning on both left and right context. As a result, the pretrained BERT model can be fine-tuned on a wide range of downstream NLP tasks with the addition of just a single task-specific output layer. BERT is based on the transformer architecture [75] which uses the idea of self attention to better capture an understanding of the context of each word, through focusing selectively on different parts of the input. The transformer architecture consists of an encoder and a decoder. The encoder takes a sequence of input tokens and produces a sequence of hidden representations, which capture the meaning of each token in context. The decoder then takes this hidden representation and autoregressively generates a sequence of output tokens, one at a time.

BERT's model architecture is stack of transformer encoders. Figure 5.1 shows the architecture of a transformer encoder. The original BERT paper [74] introduced two BERT models: (i) BERT$_{BASE}$ consisting of 12 transformer encoders stacked together, each with 12 multihead attention layers and hidden

size of 768, with a total of 110 million parameters, and (ii) BERT$_{LARGE}$ consisting of 24 transformer encoders stacked together, each with 16 multihead attention layers and hidden size of 1024, with a total of 340 million parameters.

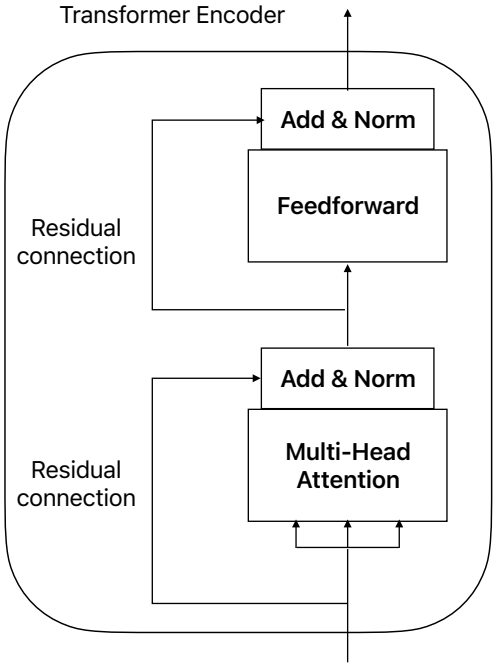


Figure 5.1 Transformer Encoder

Both models used a subword vocabulary consisting of 30,000 tokens generated using the WordPiece algorithm [76]. The use of WordPiece means that the first step in the input pipeline would then be subword tokenization using WordPiece, and all further processing would be on subword tokens rather than words. The advantage of using subword tokens is that it is an effective mechanism to deal with out of vocabulary words.

A fundamental issue with transformers is that the complexity of the model (both time and memory requirements) grow quadratically with the length of the input. As a result, when using transformers it is necessary set a fixed input length that is long enough to provide sufficient context for the model while keeping the model computationally tractable. In the case of BERT a fixed input size of 512 subword tokens was used.

BERT used two training strategies to pretrain on vast amounts of unlabeled text data: (i) Masked language modeling (MLM), and (ii) Next sentence prediction (NSP). We describe both training strategies below.

Masked language modeling is a training strategy where the model is presented with a series of sentences from the training corpus where a random sample of tokens for each training sequence is selected for use in the learning task. Once selected, a token is used by the model in one of three ways: (i) It is

replaced by the unique vocabulary token [MASK], (ii) It is replaced with another token from the vocabulary, which is randomly sampled based on token unigram probabilities, and (iii) It is left unchanged. Specifically, BERT sampled 15% of the input tokens in a training sequence for learning. Of these, 80% were replaced with [MASK], 10% were replaced with tokens randomly sampled from the vocabulary and the remaining were left unchanged. The training objective in MLM was to predict the original inputs for each of the masked tokens. The cross-entropy loss from these predictions drove the training process of all the parameters of the BERT model. The objective of the MLM strategy was predicting words from surrounding contexts with the goal of producing effective word-level representations.

In next sentence prediction the model is presented with pairs of sentences and asked to predict whether each pair consists of an actual pair of adjacent sentences from the training corpus, or a pair of unrelated sentences. Specifically, for training BERT 50% of the training sentences consisted of positive pairs (i. e. they were actual pairs of adjacent sentences from the training corpus) while the remaining 50% consisted of sentence pairs where the second sentence of the pair was randomly selected from elsewhere in the corpus. The NSP loss was based on how well the model could distinguish the true pairs from the random pairs.

BERT was trained on a text corpus of 3.3 billion words. Pairs of sentences were selected from the training corpus according to the NSP training regimen. The sentence pairs were sampled in such a way that their combined lengths was less than 512 tokens. Tokens within these sentence pairs were masked using the MLM strategy. BERT was then trained on MLM and NSP together with the goal of minimizing the combined losses of both strategies.

5.2 Experiments

5.2.1 Data used

We used the LibriTTS dataset [77], a multi-speaker English corpus of approximately 585 hours of read English speech designed for TTS research. Our phrasing models were trained using the open sourced LibriTTS alignments[1], created using the Montreal-Forced-Aligner[2] with the pretrained English model. These alignments provide the locations of pauses introduced by the speaker while recording the utterances, which were used to train the phrasing models. The 'train-clean-360' split was used for training while the 'dev-clean' and 'test-clean' splits were used for validation and test respectively. Since the LibriTTS dataset is a multi-speaker dataset, our phrasing models trained using this dataset are generic speaker independent phrasing models for English. This is the main difference between the phrasing models presented in this chapter and all the other phrasing models presented in the thesis, which are speaker dependent phrasing models trained on the same data used to build the TTS model.

[1] Available at `https://github.com/kan-bayashi/LibriTTSLabel`

[2] `https://github.com/MontrealCorpusTools/Montreal-Forced-Aligner`

5.2.2 Systems built

We train two phrase break prediction models on the dataset described in Section 5.2.1.

1. **Model 1: BLSTM with task-specific embeddings**: The first model is a BLSTM network, that uses task-specific static word embeddings trained from scratch. This model serves as our baseline, and is henceforth referred to as the BLSTM Model.

2. **Model 2: BERT model fine-tuned on phrase break prediction**: The second model consists of a pretrained BERT model fine-tuned for phrase break prediction. This model is henceforth referred to as the BERT Model.

We describe both models below. These models are generic speaker independent phrasing models for English, as opposed to all other phrasing models presented so far in this thesis, which are speaker dependent phrasing models trained on the same data used to build the TTS model. Integration and usage of these models in an End-to-End Text-to-Speech (TTS) system is addressed in Chapter 6. The code to train both models is available online at `https://github.com/anandaswarup/phrase_break_prediction`.

5.2.2.1 Model 1: BLSTM with task-specific embeddings (BLSTM Model)

This section details our first phrasing model, a BLSTM network for token classification. Given an input sequence of words, the model predicts for each word boundary whether a pause (break) should be inserted afterward. The model architecture consists of three main components, as shown in Figure 5.2:

1. **Word Embeddings**: Each word in the input sequence is mapped to a corresponding word embedding. These embeddings are randomly initialized and jointly trained along with the model on the task at hand i. e. phrase break prediction.

2. **BLSTM Layers**: The sequence of word embeddings is then fed into a stack of two BLSTM layers. These layers capture contextual information within the sequence.

3. **Binary Classifier**: The output from the stacked BLSTM layers, representing each input token, is fed into a simple binary classifier. This classifier consists of a dense feed-forward layer followed by a softmax activation function. The softmax layer outputs probabilities for each word boundary belonging to the set of possible break labels (Break "B" or No Break "NB").

This entire model was randomly initialized and trained to convergence using the Adam optimizer [78] and a cross-entropy loss. Table 5.1 shows the model parameter as well as training hyperparameter values for this model.

During inference, to generate a sequence of break indices for a given text input, a greedy approach is followed. We first run the forward pass of the model on the input sequence, and select the most likely break index (B or NB) for each input token by applying an argmax over the probabilities generated by the softmax layer corresponding to that particular token.

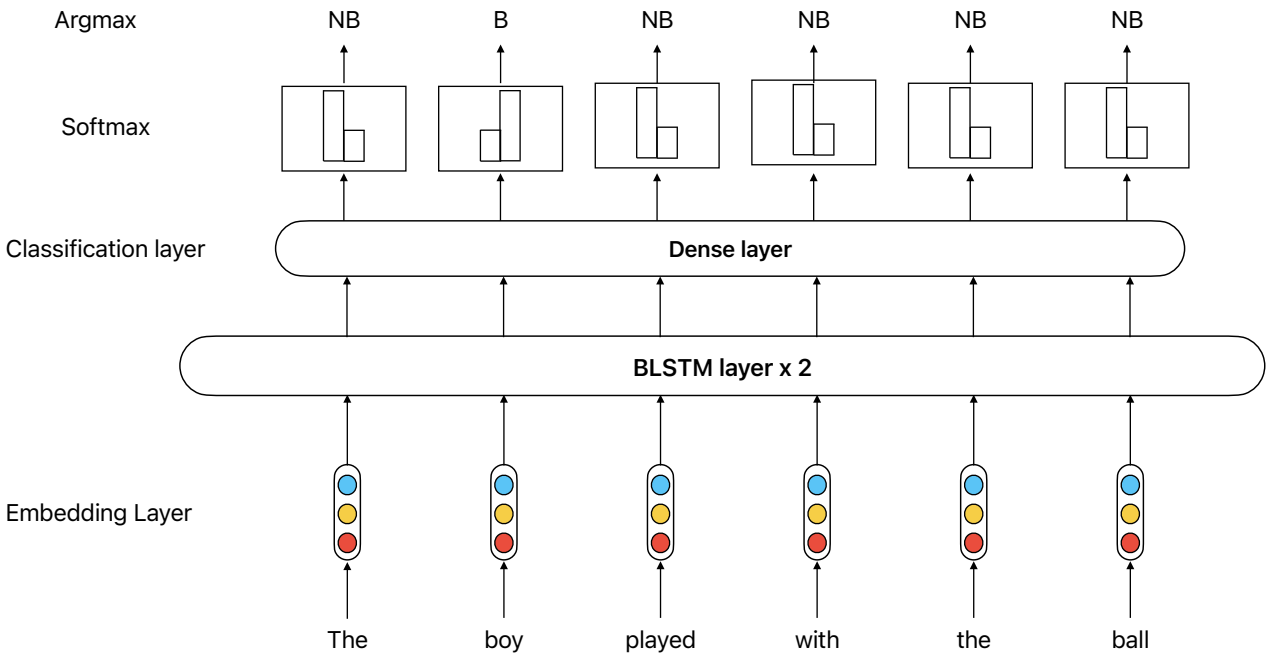


Figure 5.2 BLSTM token classification model using task-specific static embeddings trained from scratch. Inputs to the model are word embeddings which are randomly initialized and jointly trained along with the model on the task at hand, outputs of the model are probabilities from a softmax layer over the set of possible tags (B and NB).

Table 5.1 Model parameter and training hyperparameter values for the BLSTM token classification model using task-specific word embeddings trained from scratch

word embedding dimension	300
number of BLSTM layers	2
size of the hidden layer of each BLSTM	512
training batch size	64
training optimizer	Adam [78]
learning rate	0.001
number of training epochs	10

5.2.2.2 Model 2: BERT model fine-tuned on phrase break prediction (BERT Model)

This model consists of a pretrained BERT model, with an additional token classification layer to perform phrase break prediction. As we had neither the data nor the computational resources to train a BERT model from scratch, we made use of the Transformers[3] library which provides APIs and tools to easily download and fine-tune state-of-the-art pretrained models. We used the 'bert-base-uncased' model from the Transformers library, which is a pretrained BERT$_{BASE}$ model trained on uncased English text. Figure 5.3 shows the architecture of this model.

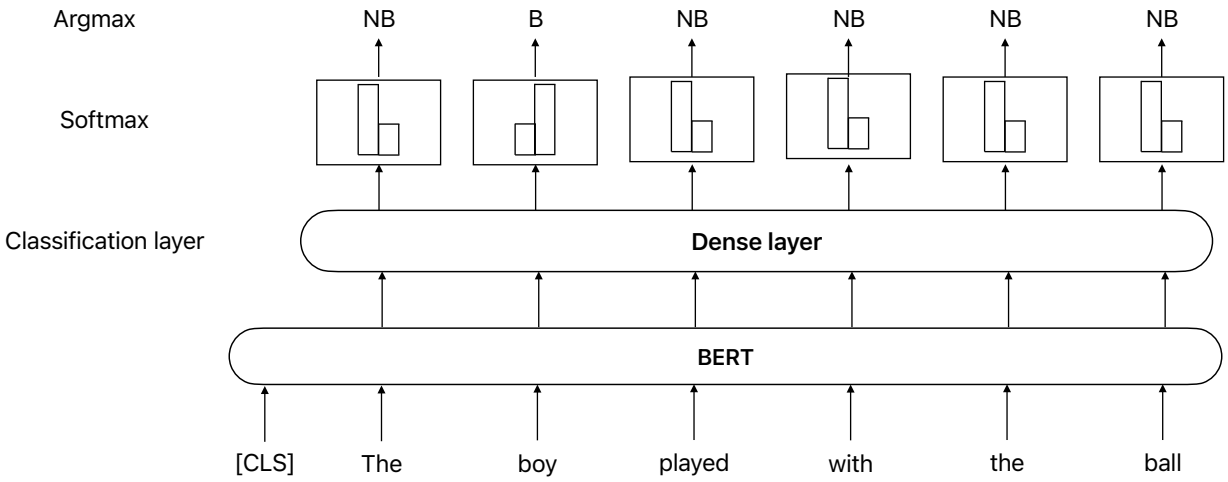


Figure 5.3 BERT model with a token classification head. The BERT model was pretrained on uncased English text and was later fine-tuned on phrase break prediction. The dense classification layer was randomly initialized, and its parameters were learned from scratch.

The dense classification layer was randomly initialized, and the entire model was fine-tuned on phrase break prediction using the Adam optimizer [78] and a cross-entropy loss. During training, the parameters of the dense classification layer were learned from scratch while the parameters of the pretrained BERT model were fine-tuned for phrase break prediction. Table 5.2 shows the fine-tuning parameter values used for this model.

The classification layer was randomly initialized, and the entire model was fine-tuned on phrase break prediction using the Adam optimizer [78] and a cross-entropy loss. During training, the parameters of the classification layer were learned from scratch while the parameters of the pretrained BERT model were fine-tuned on phrase break prediction. Table 5.2 shows the fine-tuning parameter values used for this model.

[3] `https://huggingface.co/docs/transformers/index`

Similar to the BLSTM model, the BERT model follows a greedy approach during inference to generate a sequence of break indices for a given text input. We first run the forward pass of the model on the input sequence, and select the most likely break index (B or NB) for each input token by applying an argmax over the probabilities generated by the softmax layer corresponding to that particular token.

Table 5.2 Fine-tuning parameter values for the BERT model with an additional token classification layer fine-tuned on phrase break prediction

bert model name	'bert-base-uncased'
batch size	64
learning rate	0.00001
gradient clipping norm	10
number of epochs	10

5.2.3 Results

Table 5.3 shows the results of both models described in Section 5.2.2 on the phrase break prediction task. We report our results in terms of the F-Measure [45] which is defined as the harmonic mean of precision and recall.

Table 5.3 Performance (in terms of the F-measure) of the BLSTM model trained from scratch and the fine-tuned BERT model on phrase break prediction

BLSTM Model	0.88
BERT Model	**0.92**

An examination of the results shows that the BERT model outperforms the BLSTM model on phrase break prediction.

5.3 Summary

In this chapter, we presented our work on phrase break prediction using bidirectional encoder representations. We trained two models for phrase break prediction: (i) A BLSTM token classification model using task-specific static word embeddings trained from scratch, and (ii) A BERT model with an additional token classification layer fine-tuned on phrase break prediction. As both models were trained on the same multi-speaker dataset, these are generic speaker independent phrasing models for English,

in contrast to all other phrasing models presented in this thesis, which are speaker dependent phrasing models trained on the same data used to build the TTS model.

The BLSTM token classification model, uses a stack of BLSTM layers followed by a dense classification layer to perform token classification. Given an input word sequence, the model labels each word boundary with either a B or NB indicating whether a break should be inserted after that word or not. Inputs to the model were randomly initialized word embeddings which were jointly trained along with the model on the task at hand. Outputs of the model were probabilities from the softmax layer over the set of possible tags (B and NB in this case). The entire model was randomly initialized and trained to convergence using the Adam optimizer and a cross-entropy loss.

The BERT model consists of a pretrained BERT model, with an additional token classification layer to perform phrase break prediction. As we had neither the data nor the computational resources to train a BERT model from scratch, we made use of the Transformers library which provides APIs and tools to easily download and fine-tune state-of-the-art pretrained models. We used the 'bert-base-uncased' model from the Transformers library, which is a pretrained BERT$_{BASE}$ model trained on uncased English text. The dense classification layer was randomly initialized, and the entire model was fine-tuned on phrase break prediction using the Adam optimizer and a cross-entropy loss. During training, the parameters of the dense classification layer were learned from scratch while the parameters of the pretrained BERT model were fine-tuned for phrase break prediction.

For both models, to generate a sequence of break indices for a given text input during inference, a greedy approach was followed. The forward pass of the model was first run on the input sequence, and the most likely break index (B or NB) was selected for each input token by applying an argmax over the probabilities generated by the softmax layer corresponding to that particular token.

We evaluated the performance of both models on phrase break prediction and showed that the BERT model with an additional token classification layer fine-tuned on phrase break prediction outperformed the BLSTM token classification model using task-specific static word embeddings trained from scratch.

Chapter 6

Application of phrase break prediction in End-to-End TTS systems

Text-to-Speech (TTS) can be considered as a signal inversion problem: Given a highly compressed source signal (text), we need to "decompress" it back to audio. This is a difficult task as there are multiple ways for the same text to be spoken. Over the past few years, along with the rise of Seq2Seq architectures in neural machine translation and speech recognition, there has been a push towards deep Seq2Seq TTS architectures that can be trained on <text, audio> pairs without the need for complex subsystems. However, unlike neural machine translation or speech recognition, TTS outputs are continuous, and the output sequences are much longer than the input sequences.

These models effectively integrate out unlabeled attributes such as style or prosody. As a result, during synthesis, explicit control of these latent attributes is not possible. To enable control of these attributes, typical End-to-End TTS systems have to be augmented with a mechanism for capturing such attributes. Thus, style modeling in End-to-End TTS system boils down to the problem of learning a latent space that represents prosody, and then sampling from that space to generate speech in the desired style.

There are two broad approaches to modeling and learning this latent prosody space: one which constructs the latent prosody space through the use of a deterministic encoder that projects reference speech in the desired style into a learned embedding space, and the other which models the latent space by means of a variational autoencoding framework, using Gaussian mixture priors. While there has been considerable work in style and prosody modeling in End-to-End TTS systems [79–87], in comparison very little work has been done for phrase break prediction in End-to-End TTS systems.

In this chapter, we present work on phrase break prediction in the context of End-to-End TTS systems. We explore the following key questions:

1. **Does including an external, explicit phrasing model benefit End-to-End TTS?** We hypothesize that incorporating a dedicated model for predicting phrase breaks can improve the system's capability to generate natural-sounding speech.

2. **How can we assess the effectiveness of an external phrasing model?** We propose using listener comprehension to evaluate the model's impact on the clarity and understandability of synthesized

speech. Our core hypothesis is that employing an external phrasing model within an End-to-End TTS system will lead to enhanced listener comprehension of the synthesized speech.

To address these questions, we design a framework that integrates and evaluates external phrase break prediction models in End-to-End TTS systems. The rest of this chapter is organized as follows: Section 6.1 describes the End-to-End TTS system used in this work, Section 6.2 describes the phrasing models used in this work, Section 6.3 presents our experimental setup along with results and Section 6.4 summarizes the chapter.

6.1 End-to-End TTS system

The End-to-End speech synthesis system consists of:

1. A Tacotron2 model with Dynamic Convolutional Attention, which modifies the hybrid location sensitive attention mechanism to improve generalization on long utterances. This model takes text as input and predicts a sequence of mel spectrogram frames as output.

2. A WaveRNN based vocoder, which generates a waveform from the mel spectrogram predicted by the Tacotron2 model.

6.1.1 Architecture Details

We describe the architecture details of both the Tacotron2 and WaveRNN models below. The mel spectrograms used for training both models are computed from 22,050 Hz audio using a 50 ms frame size, a 12.5 ms frame shift, an FFT size of 2048 and a Hann window. The FFT energies are then transformed to the mel scale using an 80 channel mel filterbank followed by log dynamic range compression.

6.1.1.1 Tacotron2 model

The Tacotron2 model used in this work is based on the system described in [88], which is composed of an encoder and decoder with attention. Figure 6.1 shows the block diagram of this model. We use phoneme sequence in combination with punctuation and word boundaries as input to the Tacotron2 model. This way mispronunciations are reduced, and the network learns appropriate pausing through the punctuation. More specifically, the Tacotron2 model learns to insert a pause whenever it encounters a comma in the text.

The encoder takes an input text sequence and maps this into a sequence of hidden states. The input phonemes are represented using a learned 512 dimensional embedding., which is then passed through a stack of 3 convolutional layers, each containing 512 filters of shape 5×1 followed by batch normalization and ReLU activation. These convolutional layers model longer-term context in the input text sequence. The output of the final convolutional layer is passed to a single bidirectional LSTM layer of

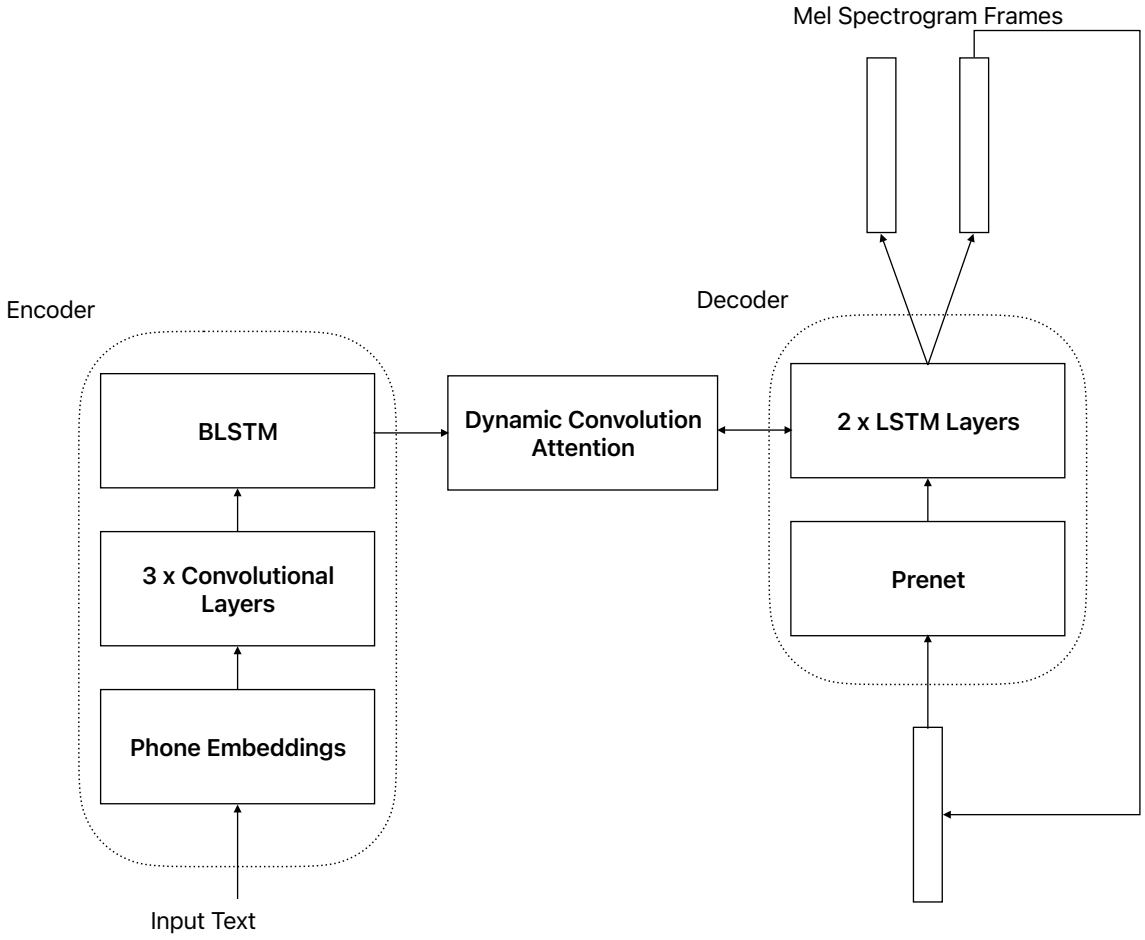


Figure 6.1 Architecture of the Tacotron2 model

512 units (256 forward units + 256 backward units) to generate the encoded hidden state sequence (also called the memory) which is the output of the encoder.

The output of the encoder is passed to the attention network which summarizes the full encoded sequence as a fixed-length context vector for each decoder output step. Our system uses dynamic convolution attention [88], which uses 1-D convolutions consisting of 8 filters of length 21 for the static and dynamic filters respectively, and a 1-D convolution consisting of 1 filter of length 11 for the causal prior filter.

The decoder is an autoregressive recurrent neural network which predicts a sequence of mel spectrogram frames from the encoded input sequence, one frame at a time. The prediction from the previous time step is passed to through a prenet containing 2 fully connected layers of 256 units with ReLU activations. This prenet acts as an information bottleneck and is essential for learning attention. The prenet output and the attention context vector are concatenated and passed through a stack of 2 LSTM layers of 1024 units each. The LSTM output is concatenated with the attention context vector and projected

through a linear transform to predict the target mel spectrogram frame. We use a reduction factor of 2, i. e. we predict two mel spectrogram frames for each decoder step.

All the convolutional layers in the network are regularized using dropout with probability 0.5 while the LSTM layers are regularized using dropout with probability 0.1. To ensure output variation at inference time, dropout with probability 0.5 is applied to the layers of the decoder prenet.

6.1.1.2 WaveRNN model

The WaveRNN model used in this work is based on the system described in [89, 90], which consists of a conditioning network and an autoregressive network. The conditioning network consists of a pair of bidirectional GRU layers of 128 units each. The autoregressive network is a single GRU layer of 896 units followed by a pair of affine layers and finally a softmax layer of 1024 units which predicts 10 bit mu-law quantized audio samples. Figure 6.2 shows the block diagram of this model.

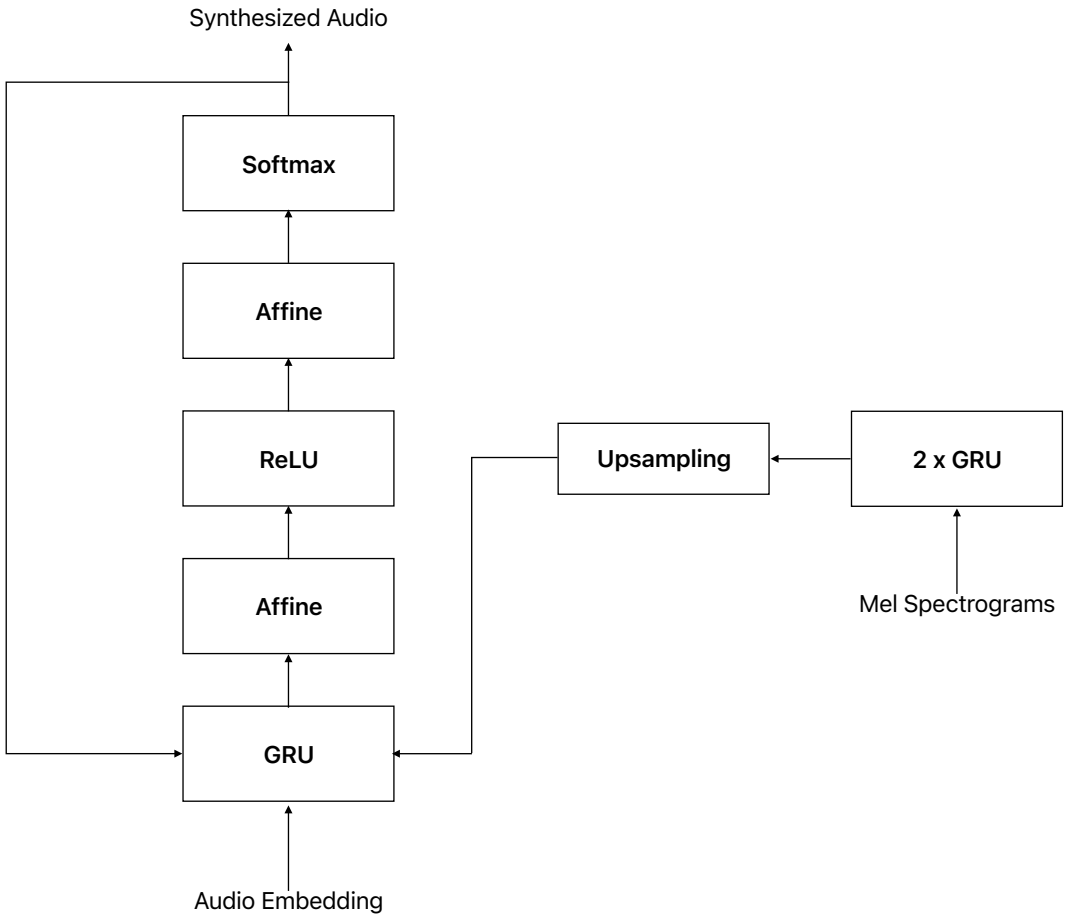


Figure 6.2 Architecture of the WaveRNN model

6.1.2 Training details

We train both the Tacotron2 and the WaveRNN models separately on the LJSpeech dataset [91], a public dataset consisting of 13,100 short audio clips of a single speaker, reading passages from 7 non-fiction books. The total length of the dataset is $\sim$24 hrs. For both models, we train on a 12,838 utterance subset ($\sim$23 hrs), while our validation and held out test subsets have 131 utterances each.

The Tacotron2 model is trained using teacher-forcing (the ground truth mel spectrogram is used as the input for each decoder step). We train using the Adam optimizer for 300k steps with a batch size of 128 on a single Nvidia GeForce RTX 2080 Ti GPU. We use an initial learning rate of 1×10^{-3} which is then reduced to 5×10^{-4}, 2.5×10^{-4}, 1.25×10^{-4}, 6.25×10^{-5} and 3.125×10^{-5} at 20k, 40k, 100k, 150k and 200k steps respectively.

Furthermore, we train the WaveRNN model using the Adam optimizer for 300k steps with a batch size of 32 on a single Nvidia GeForce RTX 2080 Ti GPU. We use an initial learning rate of 4×10^{-4} which is halved every 25k steps.

The code used to train both models can be found at `https://github.com/anandaswarup/TTS`.

6.2 Phrasing models

The phrasing models used in this work are the same models described in Chapter 5. These models are

1. **Model 1: BLSTM with task-specific embeddings**: The first model is a BLSTM network, that uses task-specific static word embeddings trained from scratch. This model serves as our baseline, and is henceforth referred to as the BLSTM Model.

2. **Model 2: BERT model fine-tuned on phrase break prediction**: The second model consists of a pretrained BERT model fine-tuned for phrase break prediction. This model is henceforth referred to as the BERT Model.

Both models were trained on the LibriTTS dataset [77], a multi-speaker English corpus of approximately 585 hours of read English speech designed for TTS research. The phrasing models were trained using the open sourced LibriTTS alignments[1], created using the Montreal-Forced-Aligner[2] with the pretrained English model. These alignments provide the locations of pauses introduced by the speaker while recording the utterances, which were used to train the phrasing models. The 'train-clean-360' split was used for training while the 'dev-clean' and 'test-clean' splits were used for validation and test respectively.

[1] Available at `https://github.com/kan-bayashi/LibriTTSLabel`

[2] `https://github.com/MontrealCorpusTools/Montreal-Forced-Aligner`

Since the LibriTTS dataset is a multi-speaker dataset, both the phrasing models described here are generic speaker independent phrasing models for English. This is the main difference between these models and all the other phrasing models presented in the thesis, which are speaker dependent phrasing models trained on the same data used to build the TTS model. For further details about either the architecture or the training details of these phrasing models, please refer to Section 5.2.

6.3 Experiments

This section describes the evaluation of the two phrasing models (described in Section 6.2), using listener comprehension as the primary metric. Our core hypothesis is that incorporating an external phrasing model within the End-to-End TTS system will lead to improved listener comprehension of the synthesized speech.

In written text, punctuation marks like commas typically indicate phrase breaks. Typically, TTS systems insert a pause in the synthesized speech whenever they encounter a comma in the text. As described in Section 6.1.1.1 the End-to-End TTS model used in this work is also trained to insert a pause whenever it encounters a comma in the text. Given unpunctuated text as input, both models described in Section 6.2 predict comma locations, essentially punctuating the text. The punctuated text is then passed to the TTS system for synthesis.

We collected children's stories with minimal or no punctuation from the web and synthesized them under three scenarios using the End-to-End TTS system described in Section 6.1. We describe all three different scenarios below

1. **Scenario 1** The story text is passed directly to the TTS system without predicting comma placement. This is referred to as the No Model in the evaluation.

2. **Scenario 2** The trained BLSTM model is used to punctuate the story text. The punctuated text is then passed to the TTS system for synthesis. This is referred to as the BLSTM Model in the evaluation.

3. **Scenario 3** The fine-tuned BERT model is used to punctuate the story text. The punctuated text is then fed to the TTS system for synthesis. This is referred to as the BERT Model in the evaluation.

To assess the quality of children's stories synthesized under these three scenarios, we conducted subjective listening tests. These tests were designed as pairwise ABX tasks. Participants were presented with the same story synthesized under two scenarios in a randomized order and asked to identify the preferred version for each story. Additionally, they had the option to select no preference" if they found the versions indistinguishable.

Five children's stories were randomly chosen and synthesized under all three scenarios, and used for the evaluation. The evaluation involved 70 post-graduate students from IIIT Hyderabad. All partici-

pants were placed in a quiet room and provided with headphones during the evaluation. The evaluation resulted in 350 responses for each pairwise comparison.

The ABX tests were hosted on `https://golisten.ucd.ie/`. Figure 6.3 shows a screenshot of the test interface.

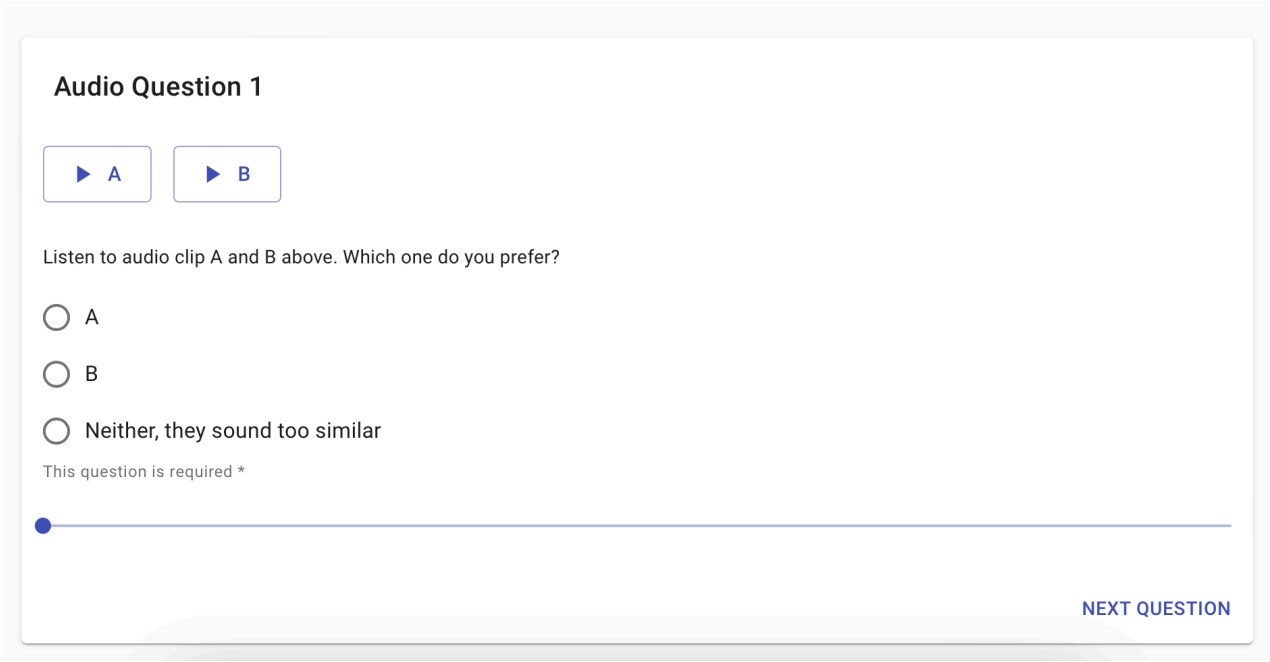


Figure 6.3 Screenshot of the ABX test interface

Table 6.1 ABX evaluations between No Model, BLSTM Model and BERT Model. *** indicates statistically significant difference at 1% level as a result of a chi-squared test

Model A	Model B	Preference A	Preference B	No Preference	
No Model	BLSTM Model	112	**156**	82	***
No Model	BERT Model	99	**183**	68	***
BLSTM Model	BERT Model	111	**163**	76	***

Table 6.1 presents the results of pairwise ABX evaluations conducted across the three scenarios. In the first comparison between the No Model and the BLSTM model, 112 responses expressed a preference for the No Model, 156 preferred the BLSTM model, and 82 indicated no preference. Similarly, in the second comparison between the No Model and the BERT model, 99 responses favored the No Model, 183 preferred the BERT model, and 68 had no preference. In the final comparison between the

BLSTM and BERT models, 111 responses preferred the BLSTM model, 156 favored the BERT model, and 76 expressed no preference.

These results demonstrate a clear preference for text punctuated using a trained phrasing model (either the BLSTM or BERT model) prior to synthesis, in contrast to text synthesized without prior comma prediction. Furthermore, the comparison between the two trained phrasing models indicates a preference for the BERT model over the BLSTM model.

To assess the statistical significance of the observed differences in preferences, a chi-squared test was performed for each comparison. The results revealed that the differences in preferences were statistically significant at the 1% significance level, suggesting a meaningful distinction between the models in terms of user preference.

These findings support the utility of incorporating explicit phrasing models within End-to-End TTS systems to enhance listener comprehension. Furthermore, these results in combination with those reported in Chapter 5 suggest that pretrained language models, such as BERT, offer potential advantages in phrase break prediction for TTS systems.

6.4 Summary

In this chapter, we presented our work on phrase break prediction in the context of End-to-End TTS systems, driven by two key questions: (i) Does incorporating an explicit external phrasing model benefit End-to-End TTS?, and (ii) How can we assess the effectiveness of an external phrasing model?

We evaluated the impact of phrase break prediction models in synthesizing children's stories. Listener comprehension served as the primary metric to assess the influence of these models on the clarity and understandability of the synthesized speech.

The End-to-End TTS system employed in this work consisted of two components: (i) A Tacotron2 model; with Dynamic Convolutional Attention which modifies the hybrid location sensitive attention mechanism to be purely location based, resulting in better generalization on long utterances, and (ii) A WaveRNN based vocoder; which takes the mel spectrogram predicted by the Tacotron2 model as input and generates a waveform as output. Typically, TTS systems introduce pauses at commas during synthesis. We trained our End-to-End TTS system to adhere to this.

Two phrasing models were investigated in this work: (i) A BLSTM model with task-specific embeddings trained from scratch, and (ii) A BERT model fine-tuned on phrase break prediction. Both models were trained on a multi-speaker English dataset. When presented with unpunctuated text, these models predict comma locations, essentially punctuating the text. The punctuated text is then passed to the TTS system for synthesis.

Unpunctuated children's stories were collected from the web and synthesized under three different scenarios using the End-to-End TTS system, as described in Section 6.3. We conducted subjective listening tests in a pairwise ABX format to evaluate the synthesized stories. The results of the listening tests revealed a clear preference for text punctuated using the trained phrasing models before being syn-

thesized, compared to text directly synthesized without comma prediction. Furthermore, the listening test results indicate a preference for the BERT model over the BLSTM model. These findings support the utility of incorporating explicit phrasing models within End-to-End TTS systems.

Chapter 7

Conclusions and future work

This thesis addresses the problem of prediction and incorporation of phrase break models in Text-to-Speech systems. Typically, phrase breaks are manifested in the speech signal in the form of several acoustic cues like pauses as well as relative changes in the intonation and duration of syllables. Cues such as pre-pausal lengthening of rhyme, speaking rate, breaths, boundary tones and glottalization also play a role in indicating phrase breaks in speech. However, representing these non-pause acoustic cues in terms of features is not easy and not well understood, and in most work the authors limit their phrase break models to predicting the locations of pauses while synthesizing speech. In this thesis too, we limit our phrase break models to predicting the location of pauses while synthesizing speech.

Chapter 2 presents our work on phrase break prediction for Text-to-Speech systems in Indian languages. We looked at syllable level units and explored the use of terminal syllable to model phrase breaks. We demonstrated the correlation between terminal syllables and acoustic breaks (pauses) found in the speech signal. Furthermore, we also demonstrated that there is a nonlinear relationship between syntax and prosody, with significant variations between syntactic breaks and phrase breaks.

We utilized these terminal syllables in building models for phrase break prediction from text in six Indian languages and demonstrated by means of objective and subjective measures that models built using terminal syllables identities perform as well as traditional models built using part-of-speech (POS) tags.

The advantage of these terminal syllables, is that they can be directly derived from the text under consideration, thus eliminating the need for additional linguistic resources like shallow parsers or POS taggers, while also eliminating the need to model phrase breaks by computationally expensive unsupervised models.

Chapter 3 presents our work on learning continuous valued word representations, i. e. representing words as points in a continuous dimensional space, also called word embeddings, for phrase break prediction. We presented two methods for learning word embeddings, the first learned word embeddings using a matrix factorization method based on latent semantic analysis (LSA), while the second was a neural network architecture for learning task-specific neural word embeddings.

LSA is a traditional method for generating word embeddings that uses a matrix factorization technique to identify semantic relationships between words in a corpus. The resulting embeddings capture the relationships between words based on the co-occurrence of words in the corpus. LSA word embeddings are not task-specific and can be used for a wide range of NLP tasks, including phrase break prediction. On the other hand, task-specific neural word embeddings are generated using neural network architectures that are specifically designed for the task at hand. In our particular case, the neural network was trained on a dataset of text and corresponding phrase break annotations to learn the appropriate embeddings for phrase break prediction. The advantage of directly learning task-specific neural word embeddings is that the network architecture combines feature induction and phrase break prediction in a single framework, and the learned word embeddings would be directly optimized for the task in question.

We trained several systems using both LSA word embeddings and task-specific neural word embeddings, on audiobook data for phrase break prediction. We evaluated their performance and showed that task-specific neural word embeddings have the advantage of being specifically optimized for phrase break prediction and thus outperform LSA embeddings when the LSA embeddings were directly used in a machine learning framework. However, when LSA embeddings were fine-tuned on phrase break prediction using our task-specific architecture their performance was comparable to the task-specific neural word embeddings. Finally, we show that encoding additional linguistic features (like POS tags) along with the word embeddings give the best performance on phrase break prediction.

Chapter 4 presents our work on RNNs for phrase break prediction and was primarily motivated by the following question: "Does capturing long-term temporal context using a recurrent architecture result in a better performance on phrase break prediction, as compared to the limited temporal context captured using a moving fixed-size window as in the case of DNNs?"

We modeled phrase break prediction as a sequence labeling task. Given a sequence of words as input, we trained RNN models to label each word boundary with either B or NB indicating whether a break should be inserted after that word or not. Inputs to the RNN models were randomly initialized word embeddings which were jointly trained along with the RNN models on the task at hand. Outputs of the RNN models were probabilities generated by a softmax layer over the set of possible tags (B and NB, in this case). The RNN models were trained using cross-entropy loss. During inference, to generate a sequence of break indices for a given text input, a greedy approach was followed. We first ran the forward pass of the RNN model on the input sequence, and selected the most likely break index (B or NB) for each input token by applying an argmax over the probabilities generated by the softmax layer corresponding to that particular token.

We trained DNNs, Simple RNNs and LSTMs on audiobook data for phrase break prediction, and performed three experiments. In the first experiment, we used the DNN model as a baseline, and compared the performance of both RNN models with the DNN model. Experimental results showed that both RNN models outperformed the DNN model while there was not much difference in the performance among the two RNN models. In the remaining two experiments, we attempted to study the effect

of varying the size of the RNN hidden layer as well as the effect of varying the input word embedding dimension, on the performance of the Simple RNN model. Results from these experiments showed that, varying either the RNN hidden layer size or the input word embedding dimension did not seem to have much of an effect on the performance of the Simple RNN model.

Chapter 5 presents our work on phrase break prediction using bidirectional encoder representations. We trained two models for phrase break prediction: (i) A BLSTM token classification model using task-specific static word embeddings trained from scratch, and (ii) A BERT model with an additional token classification layer fine-tuned on phrase break prediction. As both models were trained on the same multi-speaker dataset, these are generic speaker independent phrasing models for English, in contrast to all other phrasing models presented in this thesis, which are speaker dependent phrasing models trained on the same data used to build the TTS model.

The BLSTM token classification model, uses a stack of BLSTM layers followed by a dense classification layer to perform token classification. Given an input word sequence, the model labels each word boundary with either a B or NB indicating whether a break should be inserted after that word or not. Inputs to the model were randomly initialized word embeddings which were jointly trained along with the model on the task at hand. Outputs of the model were probabilities from the softmax layer over the set of possible tags (B and NB in this case). The entire model was randomly initialized and trained to convergence using the Adam optimizer and a cross-entropy loss.

The BERT model consists of a pretrained BERT model, with an additional token classification layer to perform phrase break prediction. As we had neither the data nor the computational resources to train a BERT model from scratch, we made use of the Transformers library which provides APIs and tools to easily download and fine-tune state-of-the-art pretrained models. We used the 'bert-base-uncased' model from the Transformers library, which is a pretrained BERT_{BASE} model trained on uncased English text. The dense classification layer was randomly initialized, and the entire model was fine-tuned on phrase break prediction using the Adam optimizer and a cross-entropy loss. During training, the parameters of the dense classification layer were learned from scratch while the parameters of the pretrained BERT model were fine-tuned for phrase break prediction.

For both models, to generate a sequence of break indices for a given text input during inference, a greedy approach was followed. The forward pass of the model was first run on the input sequence, and the most likely break index (B or NB) was selected for each input token by applying an argmax over the probabilities generated by the softmax layer corresponding to that particular token.

We evaluated the performance of both models on phrase break prediction and showed that the BERT model with an additional token classification layer fine-tuned on phrase break prediction outperformed the BLSTM token classification model using task-specific static word embeddings trained from scratch.

Finally, Chapter 6 presents our work on phrase break prediction in the context of End-to-End TTS systems, driven by two key questions: (i) Does incorporating an explicit external phrasing model benefit End-to-End TTS?, and (ii) How can we assess the effectiveness of an external phrasing model?

We evaluated the two phrasing models described in Chapter 5, using listener comprehension as the primary metric. Our core hypothesis was that incorporating an external phrasing model within the End-to-End TTS system will lead to improved listener comprehension of the synthesized speech.

We trained an End-to-End speech synthesis system consisting of: (i) A Tacotron2 model; with Dynamic Convolutional Attention which modifies the hybrid location sensitive attention mechanism to be purely location based, resulting in better generalization on long utterances, and (ii) A WaveRNN based vocoder; which takes the mel spectrogram predicted by the Tacotron2 model as input and generates a waveform as output.

In written text, punctuation marks like commas typically indicate phrase breaks. Typically, TTS systems insert a pause in the synthesized speech whenever they encounter a comma in the text. The End-to-End TTS model was also trained to insert a pause whenever it encounters a comma in the text. Given unpunctuated text as input, both phrasing models predict comma locations, essentially punctuating the text. The punctuated text is then passed to the TTS system for synthesis.

We collected children's stories with minimal or no punctuation from the web and synthesized them under three scenarios using the End-to-End TTS system described above: (i) **Scenario 1**: The story text is passed directly to the TTS system without predicting comma placement, (ii) **Scenario 2**: The trained BLSTM model is used to punctuate the story text. The punctuated text is then passed to the TTS system for synthesis, and (iii) **Scenario 3**: The fine-tuned BERT model is used to punctuate the story text. The punctuated text is then fed to the TTS system for synthesis.

The results demonstrated a clear preference for text punctuated using a trained phrasing model (either the BLSTM or BERT model) prior to synthesis, in contrast to text synthesized without prior comma prediction. Furthermore, the comparison between the two trained phrasing models indicated a preference for the BERT model over the BLSTM model. These findings support the utility of incorporating explicit phrasing models within End-to-End TTS systems.

7.1 Conclusions

The conclusions of this thesis are

1. Word terminal syllables (the last syllable of a word) can be used for phrase break prediction in Indian language TTS systems

2. Task-specific neural word embeddings have the advantage of being specifically optimized for phrase break prediction and thus outperform LSA embeddings when the LSA embeddings are directly used in a machine learning framework. However, when LSA embeddings are fine-tuned on phrase break prediction their performance is comparable to the task-specific neural word embeddings.

3. Capturing long-term temporal context using RNNs results in a better performance on phrase break prediction, as compared to the limited temporal context captured using a moving fixed-size window as in the case of DNNs

4. Bidirectional encoder representations using a BERT model with an additional token classification layer fine-tuned on phrase break prediction outperform task-specific static word embeddings learnt using a BLSTM token classification model trained from scratch

5. Incorporating explicit phrasing models in End-to-End TTS systems results in better listening comprehension, especially in the context of children's story synthesis

7.2 Future Work

The following are some potential directions for future work

1. Our work presented in Chapters 3, 4 and 5 were performed on English. One potential direction for future work would be to extend the work done in Chapter 5 to Indian languages. This would probably first entail pretraining BERT models on Indian languages, with all its attendant challenges and then fine-tuning them for phrase break prediction

2. This work primarily focuses on predicting pause locations during speech synthesis using our phrase break models. We acknowledge, as discussed in Section 1.2, that phrase breaks manifest in speech through various acoustic cues beyond pauses. These cues include relative changes in intonation, syllable duration, and other factors like pre-pausal lengthening, speaking rate, breaths, and glottalization. The task of representing and modeling these non-pausal cues for phrase break prediction remains open for future exploration.

3. The End-to-End TTS system utilized in this work leverages a Tacotron2 model with Dynamic Convolutional Attention and a WaveRNN vocoder. While this system achieves good performance, recent advancements in speech synthesis have seen the rise of generative models like SpearTTS [92] and VALL-E [93]. These models outperform autoregressive models like Tacotron2 in naturalness and expressiveness. The questions regarding the utility and effectiveness of phrasing models, as investigated in this work, can also be extended to these newer generative models. We posit that exploring the role of phrasing models within these advanced architectures is a promising direction for future research.

4. In Chapter 6 we use ABX preference tests to measure listener comprehension. Another potential direction for future work would be to design large scale subjective and objective evaluations for measuring listener comprehension, especially in the context of stories or audiobooks.

5. Integration of phrase break prediction models in End-to-End speech systems (like speech to speech translation systems, spoken dialog systems) poses engineering challenges and could be a potential direction for future work

6. Although End-to-End generative models have advanced speech synthesis, there is a need to explore whether they fully capture all relevant linguistic phenomena. A potential direction of work would be to examine the limitations of these models in handling subtle aspects of synthesis, and assess how techniques developed in this thesis might be adapted to manage other forms of linguistic control.

Publications

1. Anandaswarup Vadapalli, "An investigation of phrase break prediction in an End-to-End TTS system," *SN Computer Science*, vol. 6, no. 2, p. 91, 2025. [Online]. Available: https://doi.org/10.1007/s42979-024-03652-0

2. Anandaswarup Vadapalli and Suryakanth V Gangashetty, "An investigation of recurrent neural network architectures using word embeddings for phrase break prediction," in *INTERSPEECH 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, USA, September 8-12, 2016*, 2016, pp. 2308–2312. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2016-885

3. Anandaswarup Vadapalli and Kishore Prahallad, "Learning continuous-valued word representations for phrase break prediction," in *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*, 2014, pp. 41–45. [Online]. Available: http://www.isca-speech.org/archive/interspeech_2014/i14_0041.html

4. Anandaswarup Vadapalli, Peri Bhaskararao, and Kishore Prahallad, "Significance of word-terminal syllables for prediction of phrase breaks in text-to-speech systems for Indian languages," in *The Eighth ISCA Tutorial and Research Workshop on Speech Synthesis, Barcelona, Spain, August 31-September 2, 2013*, 2013, pp. 189–194. [Online]. Available: http://www.isca-speech.org/archive/ssw8/ssw8_189.html

Bibliography

[1] K. E. A. Silverman, M. E. Beckman, J. F. Pitrelli, M. Ostendorf, C. W. Wightman, P. Price, J. B. Pierrehumbert, and J. Hirschberg, "ToBI: a standard for labeling English prosody." in *ICSLP*. ISCA, 1992. 1

[2] P. Taylor and A. W. Black, "Assigning phrase breaks from part-of-speech sequences," *Comput. Speech Lang.*, vol. 12, no. 2, pp. 99–117, 1998. 1, 2, 3, 5, 7, 17

[3] K. Prahallad, E. V. Raghavendra, and A. W. Black, "Learning speaker-specific phrase breaks for text-to-speech systems," in *Proceedings of 7th ISCA Speech Synthesis Workshop (SSW7)*, Kyoto, Japan, 2010, pp. 148–153. 1, 3

[4] ——, "Semi-supervised learning of acoustic driven prosodic phrase breaks for text-to-speech systems," in *Proceedings of 5th International Conference on Speech Prosody (Speech Prosody 2010)*, Chicago, Illinois, May 2010. 1, 3, 7

[5] J. Hirschberg, "Pitch accent in context: Predicting intonational prominence from text," *Artif. Intell.*, vol. 63, no. 1-2, pp. 305–340, 1993. 1

[6] K. N. Ross and M. Ostendorf, "Prediction of abstract prosodic labels for speech synthesis," *Comput. Speech Lang.*, vol. 10, no. 3, pp. 155–185, 1996. 1

[7] J. P. H. van Santen, "Assignment of segmental duration in text-to-speech synthesis," *Comput. Speech Lang.*, vol. 8, no. 2, pp. 95–128, 1994. 1

[8] A. W. Black and P. Taylor, "The festival speech synthesis system," Human communication research center, University of Edinburgh, Tech. Rep., January 1997, available Online: http://www.cstr.ed.ac.uk/projects/festival. 2

[9] P. Taylor, A. W. Black, and R. Caley, "The architecture of the festival speech synthesis system," in *The Third ESCA/COCOSDA Workshop on Speech Synthesis, Blue Mountains, Australia, November 26-29, 1998*. ISCA, 1998, pp. 147–152. 2

[10] M. Ostendorf, P. Price, and S. Shattuck-Hufnagel, "The Boston University Radio News Corpus," 1995. 2

[11] P. Roach, G. Knowles, T. Varadi, and S. Arnfield, "MARSEC: A machine-readable spoken english corpus," *Journal of the International Phonetic Association*, vol. 23, no. 2, pp. 47–53, 1993. 2

[12] G. K. Anumanchipalli, L. C. Oliveira, and A. W. Black, "A statistical phrase/accent model for intonation modeling," in *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011*. ISCA, 2011, pp. 1813–1816. 2

[13] I. Read and S. Cox, "Stochastic and syntactic techniques for predicting phrase breaks," *Comput. Speech Lang.*, vol. 21, no. 3, pp. 519–542, 2007. 2

[14] J. Bachenko, E. Fitzpatrick, and C. Wright, "The contribution of parsing to prosodic phrasing in an experimental text-to-speech system," in *Association of Computational Linguistics*, New York, New York, 1986, pp. 145–153. 2

[15] J. Bachenko and E. Fitzpatrick, "A computational grammar of discourse-neutral phrasing of English," *Computational Linguistics*, vol. 16, no. 3, pp. 155–170, 1990. 2

[16] A. W. Black and K. Lenzo, "Building voices in the festival speech synthesis system," 2002, available Online: http://festvox.org/bsv. 2

[17] M. Q. Wang and J. Hirschberg, "Predicting intonational phrasing from text," in *29th Annual Meeting of the Association for Computational Linguistics, 18-21 June 1991, University of California, Berkeley, California, USA, Proceedings*, D. E. Appelt, Ed. ACL, 1991, pp. 285–292. 3

[18] M. Wang and J. Hirschberg, "Automatic classification of intonational phrase boundaries," *Computer Speech and Language*, vol. 6, pp. 175–196, 1992. 3, 5, 17

[19] P. Koehn, S. P. Abney, J. Hirschberg, and M. Collins, "Improving intonational phrasing with syntactic information," in *IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP 2000, 5-9 June, 2000, Hilton Hotel and Convention Center, Istanbul, Turkey*. IEEE, 2000, pp. 1289–1290. 3

[20] H. Schmid and M. Atterer, "New statistical methods for phrase break prediction," in *Proceedings of 20th International conference on Computational Linguistics,COLING '04*, Geneva, Switzerland, 2004. 3, 5, 17

[21] E. Marsi, M. Reynaert, A. van den Bosch, W. Daelemans, and V. Hoste, "Learning to predict pitch accents and prosodic boundaries in dutch," in *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan*, E. W. Hinrichs and D. Roth, Eds. ACL, 2003, pp. 489–496. 3

[22] P. Zervas, M. Maragoudakis, N. Fakotakis, and G. Kokkinakis, "Bayesian induction of intonational phrase breaks," in *8th European Conference on Speech Communication and Technology,*

EUROSPEECH 2003 - INTERSPEECH 2003, Geneva, Switzerland, September 1-4, 2003. ISCA, 2003. 3

[23] M. Maragoudakis, P. Zervas, N. Fakotakis, and G. K. Kokkinakis, "A data-driven framework for intonational phrase break prediction," in *Text, Speech and Dialogue, 6th International Conference, TSD 2003, Ceské Budejovice, Czech Republic, September 8-12, 2003, Proceedings*, ser. Lecture Notes in Computer Science, V. Matousek and P. Mautner, Eds., vol. 2807. Springer, 2003, pp. 189–197. 3

[24] F. Liu, H. Jia, and J. Tao, "A maximum entropy based hierarchical model for automatic prosodic boundary labeling in mandarin," in *6th International Symposium on Chinese Spoken Language Processing, ISCSLP 2008, 16-19 December, 2008, Kunming, China*, H. M. Meng, H. Jiang, J. Tao, and R. Wang, Eds. IEEE, 2008, pp. 257–260. 3

[25] Z. Ying and X. Shi, "An RNN-based algorithm to detect prosodic phrase for chinese TTS," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2001, 7-11 May, 2001, Salt Palace Convention Center, Salt Lake City, Utah, USA, Proceedings*. IEEE, 2001, pp. 809–812. 3

[26] C. Wightman, S. Shattuck-Hufnagel, M. Ostendorf, and P. J. Price, "Segmental durations in the vicinity of prosodic phrase boundaries," *Journal Acoustical Society of America*, vol. 91, no. 3, pp. 1707–1717, 1992. 3

[27] L. Redi and S. Shattuck-Hufnagel, "Variation in realization of glottalization in normal speakers," *Journal of Phonetics*, vol. 29, pp. 407–429, 2001. 3

[28] H. Kim, T. Yoon, J. Cole, and M. Hasegawa-Johnson, "Acoustic differentiation of L- and L-L% in switchboard and radio news speech," in *Proceedings of Speech Prosody*, Dresden, 2006. 3

[29] A. Parlikar and A. W. Black, "Minimum error rate training for phrasing in speech synthesis," in *Procedings of 8th ISCA Speech Synthesis Workshop (SSW8)*, Barcelona, Spain, September 2013, pp. 13–16. 3

[30] O. Watts, A. Stan, Y. Mamiya, A. Suni, J. M. Burgos, and J. M. Montero, "The Simple4All entry to the blizzard challenge 2013," in *Proceedings of the 2013 Blizzard Challenge Workshop*, August 2013. 3

[31] O. Watts, J. Yamagishi, and S. King, "Unsupervised continuous-valued word features for phrase-break prediction without a part-of-speech tagger," in *Proceedings of Interspeech*, Florence, Italy, August 2011, pp. 2157–2160. 3, 18, 19

[32] O. Watts, S. Gangireddy, J. Yamagishi, S. King, S. Renals, A. Stan, and M. Giurgiu, "Neural net word representations for phrase-break prediction without a part of speech tagger," in *Proceedings*

IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Florence, Italy, May 2014, pp. 2599–2603. 3

[33] E. Navas, I. Hernez, and I. Sainz, "Evaluation of automatic break insertion for an agglutinative and inflected language," *Speech Communication*, vol. 50, no. 11-12, pp. 888–899, 2008. 5, 17

[34] A. Bonafonte and P. Agüero, "Phrase break prediction using a finite state transducer," in *Proceedings of 11th International Workshop on Advances in Speech Technology*, 2004. 5, 17

[35] B. Busser, W. Daelemans, and A. van den Bosch, "Predicting phrase breaks with memory-based learning," in *Proceedings of 4th ISCA Speech Synthesis Workshop*, 2001. 5, 17

[36] A. Parlikar and A. W. Black, "A grammar based approach to style specific phrase prediction," in *Proceedings of Interspeech*, Florence, Italy, August 2011, pp. 2149–2152. 5, 17

[37] ——, "Data-driven phrasing for speech synthesis in low-resource langauges," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Kyoto, Japan, March 2012. 5, 17

[38] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependences in stochastic language modeling," *Computer Speech and Language*, vol. 8, pp. 1–38, 1994. 5

[39] N. S. Krishna and H. A. Murthy, "A new prosodic phrasing model for Indian language Telugu," in *INTERSPEECH-2004-ICSLP*, vol. 1, Oct 6-11 2004, pp. 793–796. 5, 17

[40] A. Bellur, K. Narayan, K. Krishnan, and H. Murthy, "Prosody modeling for syllable-based concatenative speech synthesis of Hindi and Tamil," in *Proceedings of 2011 National Conference on Communications*, 2011, pp. 1–5. 5

[41] S. C. Pammi and K. Prahallad, "POS tagging and chunking using decision forests," in *Proceedings of the IJCAI-07 workshop on Shallow Parsing in South Asian Languages*, Hyderabad, India, 2007. 5

[42] K. Prahallad, E. N. Kumar, V. Keri, S. Rajendran, and A. W. Black, "The IIIT-H Indic Speech Databases," in *Proceedings of Interspeech*, Portland, Oregon, USA, 2012. 6

[43] K. Prahallad, A. W. Black, and R. Mosur, "Sub-phonetic modeling for capturing pronunciation variations for conversational speech synthesis," in *Proccedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Toulouse, France, 2006, pp. 853–856. 6, 22, 36

[44] P. Avinesh and K. Gali, "Part-of-Speech tagging and chunking using conditional random fields and transformation based learning," in *Proceedings of the IJCAI-07 workshop on Shallow Parsing in South Asian Languages*, 2007. 9

[45] C. J. van Rijsbergen, *Information Retrieval*. Butterworth, 1979. 12, 25, 37, 47

[46] A. Vadapalli, P. Bhaskararao, and K. Prahallad, "Significance of word-terminal syllables for prediction of phrase breaks in Text-to-Speech systems in Indian languages," in *Proceedings of 8th ISCA Speech Synthesis Workshop (SSW8)*, Barcelona, Spain, September 2013, pp. 189–194. 17

[47] J. R. Firth, "A synopsis of linguistic theory 1930-55." vol. 1952-59, pp. 1–32, 1957. 17, 40

[48] Z. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954. [Online]. Available: https://link.springer.com/chapter/10.1007/978-94-009-8467-7_1 17, 40

[49] H. Schütze, "Distributional part-of-speech tagging," in *Proceedings of the 7th conference of the European Chapter of the Association of Computational Linguistics*, 1995, pp. 141–148. 17, 18, 19, 40

[50] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2177–2185. [Online]. Available: http://papers.nips.cc/paper/5477-neural-word-embedding-as-implicit-matrix-factorization.pdf 17, 40

[51] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008, pp. 160–167. 17, 40

[52] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011. 17, 40

[53] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781 18, 40

[54] F. Diaz, B. Mitra, and N. Craswell, "Query expansion with locally-trained word embeddings," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 367–377. [Online]. Available: https://aclanthology.org/P16-1035 18, 40

[55] H. Zamani and W. B. Croft, "Relevance-based word embedding," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 505–514. [Online]. Available: https://doi.org/10.1145/3077136.3080831 18, 40

[56] L.-C. Yu, J. Wang, K. R. Lai, and X. Zhang, "Refining word embeddings for sentiment analysis," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 534–539. [Online]. Available: https://aclanthology.org/D17-1056 18, 40

[57] K. Prahallad, "Automatic building of synthetic voices from audio books," Ph.D. dissertation, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 2010. 22, 36

[58] O. Watts, "Unsupervised learning for text-to-speech synthesis," Ph.D. dissertation, University of Edinburgh, 2012. 23

[59] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012. 28

[60] J. L. Elman, "Finding structure in time," *COGNITIVE SCIENCE*, vol. 14, no. 2, pp. 179–211, 1990. 29

[61] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *CoRR*, vol. abs/1503.04069, 2015. [Online]. Available: http://arxiv.org/abs/1503.04069 29, 32

[62] D. Jurafsky and J. H. Martin, *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 3rd Edition draft*, 2023. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/ 30

[63] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, D. E. Rumelhart and J. L. Mcclelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362. 31

[64] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990. 31

[65] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1109/78.650093 32

[66] Y. Bengio and X. Glorot, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of AISTATS 2010*, vol. 9, May 2010, pp. 249–256. 36

[67] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *CoRR*, vol. abs/1312.6120, 2013. [Online]. Available: http://arxiv.org/abs/1312.6120 36

[68] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio, "Theano: new features and speed improvements," Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012. 37

[69] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. 37

[70] A. Vadapalli and K. Prahallad, "Learning continuous-valued word representations for phrase break prediction," in *INTERSPEECH 2014, 15th Annual Conference of the International Speech Communication Association, Singapore, September 14-18, 2014*. ISCA, 2014, pp. 41–45. 40

[71] A. Vadapalli and S. V. Gangashetty, "An investigation of recurrent neural network architectures using word embeddings for phrase break prediction," in *INTERSPEECH 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*. ISCA, 2016, pp. 2308–2312. 40

[72] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, M. A. Walker, H. Ji, and A. Stent, Eds. Association for Computational Linguistics, 2018, pp. 2227–2237. [Online]. Available: https://doi.org/10.18653/v1/n18-1202 40

[73] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018. [Online]. Available: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf 40

[74] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186. [Online]. Available: https://doi.org/10.18653/v1/n19-1423 41

[75] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008. 41

[76] M. Schuster and K. Nakajima, "Japanese and korean voice search," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012, pp. 5149–5152. 42

[77] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "LibriTTS: A corpus derived from librispeech for text-to-speech," in *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, G. Kubin and Z. Kacic, Eds. ISCA, 2019, pp. 1526–1530. [Online]. Available: https://doi.org/10.21437/Interspeech.2019-2441 43, 53

[78] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980 44, 45, 46

[79] R. J. Skerry-Ryan, E. Battenberg, Y. Xiao, Y. Wang, D. Stanton, J. Shor, R. J. Weiss, R. Clark, and R. A. Saurous, "Towards End-to-End prosody transfer for expressive speech synthesis with Tacotron," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 4700–4709. 49

[80] Y. Wang, D. Stanton, Y. Zhang, R. J. Skerry-Ryan, E. Battenberg, J. Shor, Y. Xiao, Y. Jia, F. Ren, and R. A. Saurous, "Style Tokens: Unsupervised style modeling, control and transfer in End-to-End speech synthesis," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 5167–5176. 49

[81] Y. Zhang, S. Pan, L. He, and Z. Ling, "Learning latent representations for style control and transfer in End-to-End speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*. IEEE, 2019, pp. 6945–6949. 49

[82] Y. Lee and T. Kim, "Robust and fine-grained prosody control of End-to-End speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*. IEEE, 2019, pp. 5911–5915. 49

[83] W. Hsu, Y. Zhang, R. J. Weiss, H. Zen, Y. Wu, Y. Wang, Y. Cao, Y. Jia, Z. Chen, J. Shen, P. Nguyen, and R. Pang, "Hierarchical generative modeling for controllable speech synthesis," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. 49

[84] E. Battenberg, S. Mariooryad, D. Stanton, R. J. Skerry-Ryan, M. Shannon, D. Kao, and T. Bagby, "Effective use of variational embedding capacity in expressive end-to-end speech synthesis," *CoRR*, vol. abs/1906.03402, 2019. 49

[85] V. Aggarwal, M. Cotescu, N. Prateek, J. Lorenzo-Trueba, and R. Barra-Chicote, "Using VAEs and Normalizing Flows for One-Shot Text-to-Speech Synthesis of expressive speech," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*. IEEE, 2020, pp. 6179–6183. 49

[86] R. Valle, J. Li, R. Prenger, and B. Catanzaro, "Mellotron: Multispeaker Expressive Voice Synthesis by conditioning on Rhythm, Pitch and Global Style Tokens," in *2020 IEEE International*

Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020. IEEE, 2020, pp. 6189–6193. 49

[87] T. Hu, A. Shrivastava, O. Tuzel, and C. Dhir, "Unsupervised Style and Content Separation by Minimizing Mutual Information for Speech Synthesis," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*. IEEE, 2020, pp. 3267–3271. 49

[88] E. Battenberg, R. J. Skerry-Ryan, S. Mariooryad, D. Stanton, D. Kao, M. Shannon, and T. Bagby, "Location-relative attention mechanisms for robust long-form speech synthesis," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*. IEEE, 2020, pp. 6194–6198. [Online]. Available: https://doi.org/10.1109/ICASSP40776.2020.9054106 50, 51

[89] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 2415–2424. [Online]. Available: http://proceedings.mlr.press/v80/kalchbrenner18a.html 52

[90] J. Lorenzo-Trueba, T. Drugman, J. Latorre, T. Merritt, B. Putrycz, R. Barra-Chicote, A. Moinet, and V. Aggarwal, "Towards achieving robust universal neural vocoding," in *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, G. Kubin and Z. Kacic, Eds. ISCA, 2019, pp. 181–185. [Online]. Available: https://doi.org/10.21437/Interspeech.2019-1424 52

[91] K. Ito and L. Johnson, "The LJ speech dataset," https://keithito.com/LJ-Speech-Dataset/, 2017. 53

[92] E. Kharitonov, D. Vincent, Z. Borsos, R. Marinier, S. Girgin, O. Pietquin, M. Sharifi, M. Tagliasacchi, and N. Zeghidour, "Speak, read and prompt: High-fidelity text-to-speech with minimal supervision," *CoRR*, vol. abs/2302.03540, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2302.03540 62

[93] C. Wang, S. Chen, Y. Wu, Z. Zhang, L. Zhou, S. Liu, Z. Chen, Y. Liu, H. Wang, J. Li, L. He, S. Zhao, and F. Wei, "Neural codec language models are zero-shot text to speech synthesizers," *CoRR*, vol. abs/2301.02111, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2301.02111 62