Bachelorarbeit im Fach Informatik
RHEINISCH-WESTFÄLISCHE TECHNISCHE HOCHSCHULE AACHEN
Lehrstuhl für Informatik 6
Prof. Dr.-Ing. H. Ney

Punctuation Prediction for Speech Data

8. Juni 2020

vorgelegt von:
Nicolas Faber
Matrikelnummer 334647

Gutachter:
PD Dr. R. Schlüter
Prof. Dr.-Ing. H. Ney
Betreuer:
Christoph M. Lüscher

Erklärung

Faber, Nicolas Maximilian
Name, Vorname

334647
Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende ~~Arbeit~~/Bachelorarbeit/~~Masterarbeit*~~ mit dem Titel

Punctuation Prediction for Speech Data

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen, 8. Juni 2020
Ort, Datum

Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen, 8. Juni 2020
Ort. Datum

Unterschrift

Abstract

This work aims to improve the performance of punctuation prediction models on textual output of automatic speech recognition systems. Transformer based models were used to investigate the impact of sentence boundary information, case information, as well as the impact of supplementing or replacing input text data with speech data.

Contents

Abstract v

1 Introduction 1

2 Theoretical Foundations 3
- 2.1 Audio Features 3
  - 2.1.1 Mel Frequency Cepstrum Coefficients 3
  - 2.1.2 Tone 3
- 2.2 Neural Network Basics 4
  - 2.2.1 Multi Layer Perceptron 5
  - 2.2.2 Recurrent Neural Network 5
  - 2.2.3 Long Short-Term Memory Neural Network 5
- 2.3 Transformer 7
  - 2.3.1 Word2Vec 9
  - 2.3.2 Speech2Vec 10
- 2.4 Punctuation Prediction Basics 11
  - 2.4.1 Challenges 12
- 2.5 Evaluation 12
  - 2.5.1 Error Types 12
  - 2.5.2 Measures 13
- 2.6 Byte Pair Encoding (BPE) 15

3 Related Work 17

4 Experiments 19
- 4.1 Data 19
- 4.2 Preprocessing 21
  - 4.2.1 Punctuation Tokens 21
  - 4.2.2 Byte Pair Encodings 22
  - 4.2.3 Mel Frequency Cepstrum Coefficients 22
  - 4.2.4 Tone 22
- 4.3 Experiment Setups 23
  - 4.3.1 Experiments on Text Data 23
  - 4.3.2 Incorporating audio features 26

5 Experiment Results **29**
5.1 Experiments on Text Data 29
5.1.1 Comparison IWSLT2012 vs IWSLT2019 29
5.1.2 Rechunking 29
5.1.3 Casing 33
5.1.4 Embeddings 38
5.1.5 Comparison with other groups 38
5.2 Incorporating audio features 40

6 Conclusion **45**

A Appendix **47**

List of Figures **49**

List of Tables **51**

Bibliography **55**

Chapter 1

Introduction

Punctuation prediction (PP) is the process of taking a text that lacks punctuation symbols such as for example commas and periods, and predicting a version of this text that has punctuation symbols inserted at grammatically or stylistically correct positions.

Punctuation prediction can improve readability of unpunctuated texts and can also be used to retrieve sentence boundary information from punctuation symbols that mark the end of sentences [Cho & Niehues$^+$ 12].

The text output produced by automatic speech recognition (ASR) systems is often unpunctuated, and punctuation prediction systems could be used in a post processing step to make the output better humanly readable [Tilk & Alumäe 15].

Applications could be human dictation, but also to potentially gain improvements in later natural language processing (NLP) tasks performed on the text such as translation [Cho & Niehues$^+$ 12], natural language understanding [Chen 99, Christensen & Gotoh$^+$ 01] or sentiment analysis [Tilk & Alumäe 15].

Older approaches to punctuation prediction did not perform that well [Matusov & Mauser$^+$ 06, Chen 99], however with the emergence of neural network technology, big improvements have been made in natural language processing tasks [**?**].

The recently proposed transformer architecture [Vaswani & Shazeer$^+$ 17] has again brought improvements in performance.

Previous works have reported good results for punctuating text using models based on the transformer architecture [Wang & Chen$^+$ 18, Yi & Tao 19].

The goal of this work is to construct a punctuation prediction system using the transformer architecture for the purpose of punctuating transcripts produced by ASR systems. It is further the goal of this work to investigate whether improvements in performance can be gained, by not only utilizing text data for punctuation prediction, but to also utilize the audio data available in this context and augmenting the text data with it.

Chapter 2

Theoretical Foundations

2.1 Audio Features

2.1.1 Mel Frequency Cepstrum Coefficients

Mel frequency cepstrum coefficients (MFCCs) are representations of audio signals that have been shown to be useful for automatic speech recognition [Davis & Mermelstein 80].
Extracting MFCCs from audio signals requires multiple steps.

The signal is segmented into frames, for the purposes of this work 25ms long frames with a 10ms shift between frames.
A window function, here the hamming window function is applied to each frame.
Then, fourier transformation is applied to the windowed frames [Ittichaichareon & Suksri^{+} 12]
The frequencies scale of the resulting spectrum is then warped according to function 2.1 into the logarithmic mel-frequency scale [Ittichaichareon & Suksri^{+} 12]

$$f_{\text{mel}} = 2595 * log_{10}(1 + \frac{f_{lin}}{700\text{Hz}}) \tag{2.1}$$

A filterbank consisting of triangular band-pass filters is applied to the spectrum.Then discrete cosine transform, shown in equation 2.2, is applied to the log-energies X_k of the outputs of the filterbank. [Davis & Mermelstein 80]
$MFCC_i$ are then the resulting cepstrum coefficients in the mel frequency scale.

$$\text{MFCC}_i = \sum_{k=1}^{n} X_k \cos[i(k - \frac{1}{2})\frac{\pi}{20}] \tag{2.2}$$

2.1.2 Tone

Tone features are part of the intonation of speech and describe the pitch in the voice of the speaker [Huang & Seide 00].

One method to estimate the pitch of an audio signal is based on the average multitude difference function (AMDF) [Xiao-Dan Mei & Jengshyang Pan^{+} 01].

Based on a sequence of samples $s(n)$ extracted at regular time intervals from the audio signal, with $n \in [0, ..., N-1]$ and N being the number of samples, AMDF calculates the error in equation 2.3 between the actual pitch of the audio signal and an estimated pitch τ, as the averaged difference between $s(n)$, and $s(n)$ shifted by τ time intervals [Xiao-Dan Mei & Jengshyang Pan+ 01].
The estimated pitch $\text{tone}_{\text{AMDF}}$ of the audio signal is then obtained through the optimization function in equation 2.4 [Xiao-Dan Mei & Jengshyang Pan+ 01].

$$\begin{gathered} E_{\text{AMDF}}(\tau) = \frac{1}{N} \sum_{n=0}^{N-1} |s(n) - s(n+\tau)| \\ \text{where } \tau \in [\tau_{\min}, \tau_{\min}+1, ..., \tau_{\max}] \end{gathered} \tag{2.3}$$

$$\text{tone}_{\text{AMDF}} = \underset{\tau}{\operatorname{argmin}}(E_{\text{AMDF}}(\tau)) \tag{2.4}$$

2.2 Neural Network Basics

Artificial neural networks (ANNs) are models, that estimate the probability distribution $p(y_{\text{pot}}|x)$ in equation 2.5 for a input $x \in X_{\text{input}}$ and possible classes y_{pot}, through the function f_θ with parameters θ in equation 2.6 [Bishop 06]. Here C_{classes} is the number of classes.
When fed with training samples $[(x_1, y_{\text{pot},1}), ..., (x_{B_{\text{batch}}}, y_{\text{pot},B_{\text{batch}}})]$, the network is evaluated according to an objective function F_{obj}, for example F_{CE} in equation 2.7 [Bishop 06]. The weights θ of the network are then updated with an update function, for example stochastic gradient descent in equation 2.8, where α is the learning rate [Bishop 06].

$$p(y_{\text{pot}}|x) = p(y_{\text{pot}})p(x|y_{\text{pot}}) \tag{2.5}$$

$$f_\theta : X_{\text{input}} \to \mathbb{R}^{C_{\text{classes}}}, x \mapsto f_\theta(x) \tag{2.6}$$

$$F_{\text{CE}}(\theta) = -\sum_{i=1}^{B_{\text{batch}}} \log f_\theta(x_i)|_{y_{\text{pot}},i} \tag{2.7}$$

$$\theta_{j+1} = \theta_j - \alpha \nabla(F_{\text{obj}}(\theta_j)) \tag{2.8}$$

2.2.1 Multi Layer Perceptron

Multi layer perceptrons (MLPs) consist of multiple layers $l \in [1, ..., L]$ where L is the number of layers [Bishop 06]. Each layer calculates an output $y^{(l)}$ from an input $y^{(l-1)}$ with $y^{(0)}$ being the input x of the MLP. Here $y^{(1)}$ is called the input layer, $y^{(L)}$ the output layer, while the layers $[y^{(i)}; 1 < i < L]$ are called hidden layers [Bishop 06]. Layer l calculates it's output by multiplying a weight matrix $\mathrm{W}_1^{(l)}$ with it's input $y^{(l-1)}$, adding a bias term $b^{(l)}$ and applying a non-linear activation function σ, as shown in equation 2.9 [Bishop 06]. One possible activation function is σ_{tanh} in equation 2.10 where d is the dimension of z and $i \in [1, ..., d]$ [Bishop 06].

$$y^{(l)} = \sigma(W_1^{(l)} y^{(l-1)} + b^{(l)}) \tag{2.9}$$

$$\begin{aligned} \sigma_{\text{tanh}}(z) &= (\tanh(z_1), ..., \tanh(z_d)) \\ \tanh(z_i) &= 2 * \frac{1}{1 + \exp(-2z_i)} - 1 \end{aligned} \tag{2.10}$$

2.2.2 Recurrent Neural Network

The layers of a recurrent neural network (RNN) differ from the layers of an MLP, in that they do not operate on single inputs x, but on time sequences x_1^T with time step $t \in [1, ..., T]$ [Bodén 01]. This is possible because recurrent layers perform operations recurring in the time dimension [Bodén 01]. The output $y_t^{(l)}$ of a recurrent layer $l \in [1, ..., L]$, with L being the number of layers and $y_t^{(0)} = x_t$ is calculated not only based on the output of the previous layer $y_t^{(l-1)}$, but also on it's own output $y_{t-1}^{(l)}$ for the previous time step, as shown in equation 2.11 [Bodén 01]. Here $W_1^{(l)}$ and $W_2^{(l)}$ are the weight matrices of layer l. For recurrent layers, backpropagation is performed through time, otherwise optimization function and training are the same as for MLPs [Bodén 01].

$$y_t^{(l)} = \sigma(W_1^{(l)} y_t^{(l-1)} + W_2^{(l)} y_{t-1}^{(l)} + b^{(l)}) \tag{2.11}$$

2.2.3 Long Short-Term Memory Neural Network

A long short-term memory (LSTM) neural network is a variant of a recurrent neural network [Hochreiter & Schmidhuber 97]. Figure 2.1 shows the architecture of a single LSTM unit consisting of a memory cell, an input gate, an output gate, and a forget gate. Here, f_t is the activation vector of the forget gate. It is described by equation 2.13 [Sak & Senior$^+$ 14]. The forget gate controls the extent to which the memory cell state of previous time step c_{t-1} flow into the memory state of the

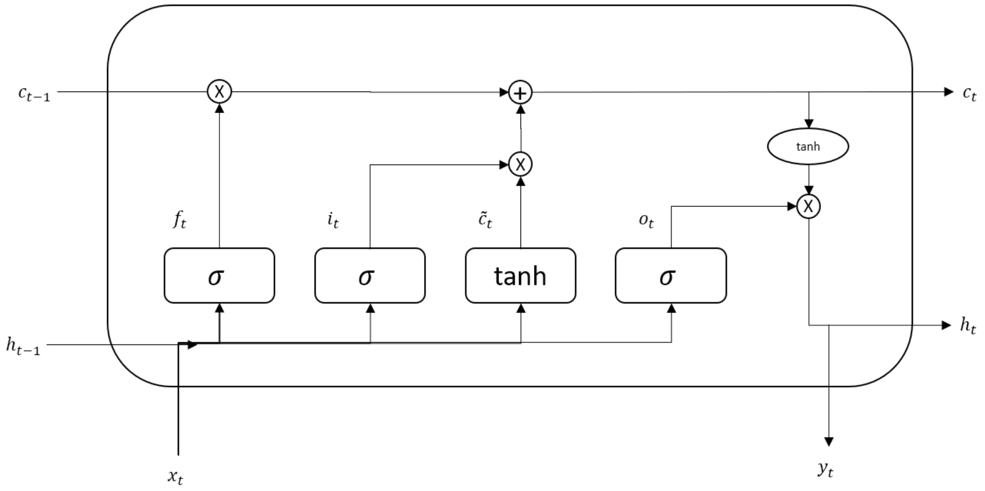


Figure 2.1: LSTM Unit

current time step c_t [Hochreiter & Schmidhuber 97].
i_t is the activation vector of the input gate, described by equation 2.12 [Sak & Senior$^+$ 14]. The input gate controls the extent to which the current input flows into the memory cell state of the current time step [Sak & Senior$^+$ 14]. The activation vector of the output gate is described by equation 2.15 [Sak & Senior$^+$ 14]. The output gate scales the amount by which the memory celll state of the current time step c_t flows into the current hidden state h_t and the current output y_t [Sak & Senior$^+$ 14]. Equations adapted from [Sak & Senior$^+$ 14].

$$i_t^{(l)} = \sigma(W_{\text{iy}}^{(l)} y_t^{(l-1)} + W_{\text{ih}}^{(l)} h_{t-1}^{(l)} + W_{\text{ic}}^{(l)} c_{t-1}^{(l)} + b_{\text{i}}^{(l)}) \tag{2.12}$$

$$f_t^{(l)} = \sigma(W_{\text{fy}}^{(l)} y_t^{(l-1)} + W_{\text{fh}}^{(l)} h_{t-1}^{(l)} + W_{\text{fc}}^{(l)} c_{t-1}^{(l)} + b_{\text{f}}^{(l)}) \tag{2.13}$$

$$c_t^{(l)} = f_t^{(l)} \odot c_{t-1}^{(l)} + i_t^{(l)} \odot \sigma_{\text{tanh}}(W_{\text{cy}}^{(l)} y_t^{l-1} + W_{\text{ch}}^{(l)} h_{t-1}^{(l)} + b_{\text{c}}^{(l)}) \tag{2.14}$$

$$o_t^{(l)} = \sigma(W_{\text{oy}}^{(l)} y_t^{(l-1)} + W_{\text{oh}}^{(l)} h_{t-1}^{(l)} + W_{\text{oc}}^{(l)} c_t^{(l)} + b_{\text{o}}^{(l)}) \tag{2.15}$$

$$h_t^{(l)} = o_t^{(l)} \odot \sigma_{\text{tanh}}(c_t^{(l)}) \tag{2.16}$$

$$y_t^{(l)} = W_{\text{yh}}^{(l)} h_t^{(l)} + b_{\text{y}}^{(l)} \tag{2.17}$$

Sometimes a combination of two LSTM layers is used in the form of a bidrection LSTM (BiLSTM). The layer called the forward layer performs the recurrent operation in the time dimension in the positive direction in time, while the backward layer performs the recurrent operation in the negative time direction [Ray & Rajeswar$^+$ 15].

2.3 Transformer

The transformer architecture is a self attention based encoder/decoder architecture, proposed by Vaswani et al. [Vaswani & Shazeer$^+$ 17]. An overview of the architecture can be seen in figure 2.3.

The two main sublayers from which the transformer architecture is built are a MLP, and the multi-head attention sublayer described in equation 2.19. The multi-head attention sublayer projects its inputs Q, K and V, the respective queries, keys and values of the attention function [Vaswani & Shazeer$^+$ 17], into h separate projection spaces corresponding to the h attention heads $head_i$. Then attention heads calculate in parallel the attention function in equation 2.18 on the projected inputs. The output of the multi-head attention sublayer is the concatenation of the outputs of the attention heads, projected into the model dimension d_{model} [Vaswani & Shazeer$^+$ 17].

A special case of the multi-head attention layer is the masked multi-head attention layer, where all future time steps are masked away [Vaswani & Shazeer$^+$ 17].

Encoder and decoder consist of a stack of one or more encoder and decoder layers respectively. A encoder layer consists of a multi-head attention sublayer and a feed forward sublayer.
A decoder layer consists of a masked multi-head attention sublayer, a multi-head attention sublayer and a feed forward sublayer. The multi-head attention sublayer of the decoder is conditioned partly on the output of the masked multi-head attention layer, and partly on the output of the stack of encoder layers. Layer normalization and residual connections are applied to all sublayers.
The output of the decoder stack is fed into a linear projection layer and finally a softmax layer which produces the output probabilities.

Input data is embedded, positionally encoded, and fed into the encoder stack. The output of the previous time steps is also embedded, positionally encoded, and fed into the input of the decoder stack [Vaswani & Shazeer$^+$ 17].

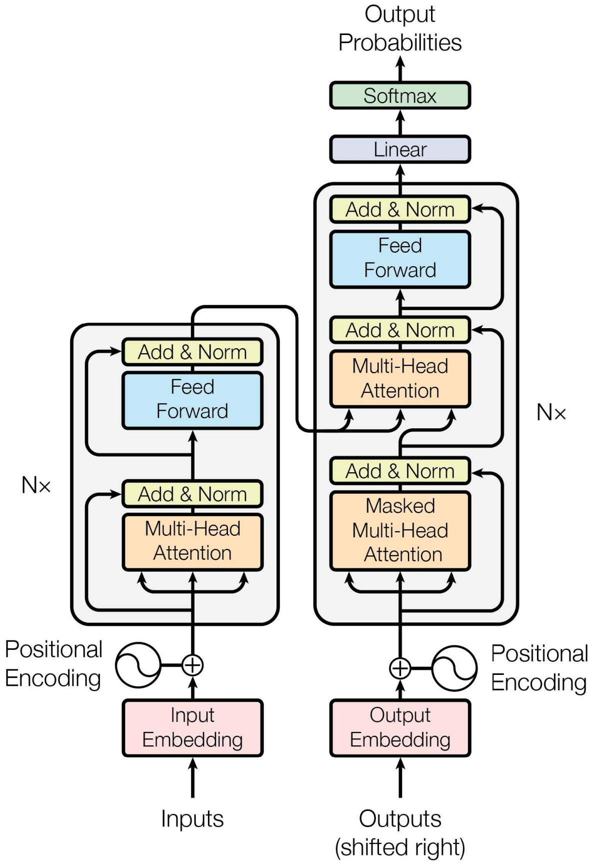
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.18}$$

$$\begin{aligned}\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W_O \\ \text{where } \text{head}_i = \text{Attention}(QW_{iQ}, KW_{iK}, VW_{iV}) \\ \text{with } W_{iQ} \in \mathbb{R}^{d_{model} \times d_k}, W_{iK} \in \mathbb{R}^{d_{model} \times d_k}, W_{iV} \in \mathbb{R}^{d_{model} \times d_v}, \\ W_O \in \mathbb{R}^{hd_v \times d_{model}}\end{aligned} \tag{2.19}$$

The decoder is setup similarly, a feed forward sublayer is fed by a multi-head attention sublayer, which in turn is fed by both the output of the decoder stack, as well as a third sublayer, which applies multi-head attention to previous time steps of the output, while future steps are masked away.

The overall architecture then consists of a stack of one or more encoder layers fed by the input data which is projected into an input embedding space and positionally encoded, followed by a stack of one or more decoder layers, which are fed by the output of the encoder stack as well as the outputs which are masked to only include previous time steps, projected into an output embedding space and positionally encoded, and finally a linear projection and a softmax layer producing the output probabilities of the network.

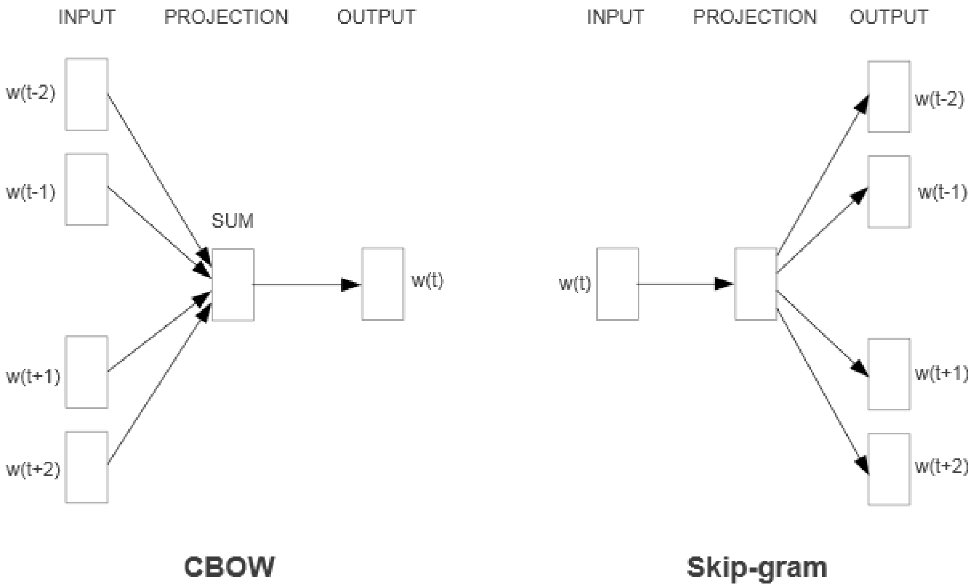
Figure 2.2: Transformer architecture from: [Vaswani & Shazeer+ 17]



2.3.1 Word2Vec

Word2Vec (W2V) is a model architecture that can learn to create embeddings for text based vocabularies [Mikolov & Chen+ 13]. Given a sequence of words w_1^N, a neural network iterates over each word and tries to predict the relationship between that word and the context of words around it [Mikolov & Chen+ 13].

Figure 2.3: Continuous bag of words (CBOW) and Skipgram Configurations of Word2vec. From: [Mikolov & Chen$^+$ 13]



There are two main variants of W2V, continuous bag-of-words (CBOW) and skip-gram [Mikolov & Chen$^+$ 13], shown in figure 2.3.1.

In both, input and targets consist of one-hot vector representations of words $w_i \in w_1^N$. Both consist of a hidden projection layer between the vector representations of the input and the vector representations in the output [Mikolov & Chen$^+$ 13].

With CBOW, the model learns to predict the one-hot vector representation of a word w_i from the context window $w_{i-j}, ...w_{i-1}, w_{i+1}, ..., w_{i+j}$ over w_1^N.
With skip-gram, the model learns to predict the context window $w_{i-j}, ...w_{i-1}, w_{i+1}, ..., w_{i+j}$ from the central word w_i [Mikolov & Chen$^+$ 13].

After training the model, the hidden weights h can be extracted and used as embedding for the used vocabulary [Mikolov & Chen$^+$ 13].

2.3.2 Speech2Vec

Speech2Vec (S2V) is an adaptation of Word2Vec that, instead of operating on text-based vocabularies, operates on sequences of audio features like MFCCs [Chung & Glass 18], where each sequence represents a spoken word. Speech2Vec then provides an embedding for such a sequence. Unlike the architecturally simpler Word2Vec, Speech2Vec is based around an encoder/ decoder based recurrent neural network, that does not operate on one-hot vector representations of words, but instead on audio sequences of variable length, where each sequence is corresponding to a spoken word [Chung & Glass 18]. Like Word2Vec, both the skip-gram and the CBOW approach can be used.

2.4 Punctuation Prediction Basics

The general concept of punctuation prediction is to predict a sequence of words ${w'}_1^M$ in equation 2.21, from a sequence of words w_i^N in equation 2.20 [Matusov & Mauser$^+$ 06]. Here the predicted words ${w'}_i$ include punctuation symbols, while the words w_i in the input do not [Matusov & Mauser$^+$ 06]. The goal is to predict a target sequence ${w'}_1^M$ that contains those punctuation symbols according to proper grammatical and linguistic rules, but is otherwise identical to w_1^N [Matusov & Mauser$^+$ 06].

$$w_1^N = w_1, ..., w_N \tag{2.20}$$

$${w'}_1^M = w'_1, ..., w'_M \tag{2.21}$$

$$F : w_1, ..., w_N \mapsto w'_1, ..., w'_M \tag{2.22}$$

$$w_1^N \to \hat{w'}_1^M(w_1^N) = \underset{{w'}_1^M}{\operatorname{argmax}}\{p({w'}_1^M|w_1^N)\} \tag{2.23}$$

A special case of this general approach is to not predict the entire new sequence consisting of words and punctuation symbols as in equation 2.22, but to instead produce a sequence of tokens (equation 2.24), with the same length N as the input sequence, which represent the punctuation symbols belonging in the corresponding positions in the input sequence [Cho & Niehues$^+$ 17].

This vastly reduces the size of the target vocabulary of the model (equation 2.25) [Cho & Niehues$^+$ 17].

$$a_1^N = a_1, ..., a_N \tag{2.24}$$

$$G : w_1, ..., w_N \mapsto a_1, ..., a_N \tag{2.25}$$

$$w_1^N \to \hat{a}_1^N(w_1^N) = \underset{a_1^N}{\operatorname{argmax}}\{p(a_1^N|w_1^N)\} \tag{2.26}$$

2.4.1 Challenges

There are several challenges with predicting the gramatically proper punctuation for unpunctuated text. Grammar can be ambiguous, leading to multiple possibilities of correct punctuation for a given text [Boháč & Rott$^+$ 17]. Different writers or writing styles may leave out or add punctuation symbols where others might not [Boháč & Rott$^+$ 17].

There is also the possibility, especially in the case of transcripts of speech, that the sentences are grammatically incorrect, end abruptly, include stuttering, or other disfluencies [Boháč & Rott$^+$ 17].

A different challenge is, that often, text might be formatted in such a way, that information about the correct segmentation into discrete sentences can be gained from line breaks and ends of paragraphs. A model trained on such text could then potentially perform worse on other kinds of text, where sentence segmentation information is absent or incorrect regarding the formatting.

2.5 Evaluation

2.5.1 Error Types

For the purpose of calculating the number of errors made by a model based on tokenized output, such as in equation 2.25, three different kinds of errors need to be considered [Che & Wang$^+$ 16]:

The first case is that in the reference, the token corresponding to a particular word in the input represents that no punctuation symbol belongs after that word, while in the hypothesis, the token corresponding to the same word in the input represents some punctuation symbol, we call this insertion (I), analogous to [Che & Wang$^+$ 16].

The second case is the inverse, where the token in the reference signifies that some punctuation symbols belong after the corresponding word in the input, while in the hypothesis, the token corresponding to the same word in the input represents a blank, a word not followed by any punctuation symbol. We call this deletion (D), analogous to [Che & Wang$^+$ 16].

The third case is that both tokens, in hypothesis and reference, corresponding to a particular word in the input, signify the existence of a punctuation symbol, but do not represent the same punctuation token. We call this substitution (S), analogous to [Che & Wang$^+$ 16].

Table 2.1: Example confusion matrix

		Hyp			
		Class A	Class B	Class C	Sum:
Ref	Class A	100	0	0	100
	Class B	30	70	0	100
	Class C	20	80	0	100
	Sum:	150	150	0	

2.5.2 Measures

From these types various error measures can be calculated [Makhoul & Kubala^{+} 99], with previous works mostly focussing on precision, recall and F$_1$-score [Tilk & Alumäe 15, Che & Wang^{+} 16, Yi & Tao 19].

Classification Outcomes

When predicting whether a token a belongs to a word w, there are foure possible outcomes: A true positive (TP) occurs when a is predicted to belong to w, and that prediction is correct. A false positive (FP) occurs when a is predicted to belong to w and that prediction is incorrect. A false negative (FN) occurs when a is not predicted to belong to w, but does belong to w. And finally, a true negative (TN) occurs when a is not predicted to belong to w, and a does not belong to w [Powers 11].

Confusion Matrix

Another way to display errors in a multi class scenario is a confusion matrix, it shows how often occurrences in the reference of a particular class are predicted to be a particular class in the hypothesis.

Table 2.1 shows an example of a confusion matrix, here each row corresponds to a class in the reference and the values in that row represent how often instances of that class in the reference were in the hypothesis predicted to be the class corresponding to the column. The fields where the class labels of the row and the column are the same contain the number of true positives for that class. The other fields in a particular column contain the number of false positives for the column class label, while the other fields in a row contrain the false negatives for the row class label. The values in the row marked "Sum:" shows the total number of occurences of the respective classes in the hypothesis, while the values in the column marked "Sum:" represent the total number of occurrences of the respective class in the reference.

Total Errors

The amount of total errors is the sum of the amounts of the individual error types, as shown in equation 2.27.

$$\text{Total Errors } (E) = I + D + S \tag{2.27}$$

Precision

Precision (P) is the ratio in equation 2.28, between the number of correct predictions of punctuation symbols (C) and total amount of predictions of punctuation symbols [Makhoul & Kubala$^+$ 99]

$$\text{Precision } (P) = \frac{\text{true positive}}{\text{true positive} + \text{false positive}} = \frac{C}{C + S + I} \tag{2.28}$$

The value of the precision measure ranges between 0 and 1 or 0% and 100%, with 1, or 100% denoting the best possible performance, and 0 or 0% the worst [Makhoul & Kubala$^+$ 99].

Recall

Recall (R), shown in equation 2.29 is the ratio between the number of correct predictions of punctuation symbols and the total number of punctuation symbols in the reference [Makhoul & Kubala$^+$ 99].

$$\text{Recall } (R) = \frac{\text{true positive}}{\text{true positive} + \text{false negatives}} = \frac{C}{C + S + D} \tag{2.29}$$

The value of the recall measure ranges between 0 and 1 or 0% and 100%, with 1, or 100% denoting the best possible performance, and 0 or 0% the worst [Makhoul & Kubala$^+$ 99].

$\mathbf{F_1}$-Score

The F_1-Score (F_1) shown in equation 2.31, also called weighted harmonic mean, is a special case of the general F_β-Score in equation 2.30, which can be understood as a combination of precision and recall [Makhoul & Kubala$^+$ 99].

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{(\beta^2 P) + R} \tag{2.30}$$

$$\text{weighted harmonic mean } (F_1) = 2 \cdot \frac{P * R}{P + R} \tag{2.31}$$

The value of the F_1-measure ranges between 0 and 1 or 0% and 100%, with 1, or 100% denoting the best possible performance, and 0 or 0% the worst [Makhoul & Kubala$^+$ 99].

Multi Class Scoring

The previously defined precision-, recall- , and F_1-scores are only measures for the performance on a single punctuation token. In order to achieve an overall measure, the scores of the individual token classes need to be combined [Che & Wang$^+$ 16]. In this work, the same method as used by [Che & Wang$^+$ 16], of calculating the total true positives over all punctuation symbol classes, replacing the number of per-class positives in equation 2.28 with the total number of predicted punctuation symbols, and replacing the number of occurrences of a class in equation 2.29 with the total number of punctuation symbols in the reference. The class representing the lack of punctuation symbols is omitted due to it dominating the measurement otherwise [Che & Wang$^+$ 16].

Classification Error Rate

The classification error rate (CER) 2.32 is a more general error measure and is the rate between total errors in equation 2.27 and the number of places M where a punctuation symbol can be predicted [Blandford & Vanderdonckt$^+$ 11].

$$\text{classification error rate } (CER) = \frac{E}{M} \tag{2.32}$$

Slot Error Rate

The slot error rate (SER) 2.33 is the rate between total errors in equation 2.27 and the number of punctuation symbols in the reference [Makhoul & Kubala$^+$ 99].

$$\text{slot error rate } (SER) = \frac{E}{C + S + D} \tag{2.33}$$

2.6 Byte Pair Encoding (BPE)

Byte Pair Encoding is a method to create a vocabulary of subword-units that replaces the word vocabulary of text data. The resulting vocabulary of subword-units can then form words that did not occur in the original vocabulary [Sennrich & Haddow$^+$ 16].

Chapter 3

Related Work

Che et al. propose extracting datasets for punctuation prediction [Che & Wang$^+$ 16], consisting of unpunctuated text and punctuation tokens, from English TED talk transcripts from IWSLT2012 [Federico & Cettolo$^+$ 12] data.
Cho et al. propose a method to represent punctuation tokens that also includes case information about the corresponding word [Cho & Niehues$^+$ 17].
Ueffing et al. report an overall F_1 score of 53.5% for punctuation prediction on IWSLT2011 data using conditional random field based models [Ueffing & Bisani$^+$ 13].
Peitz et al. [Peitz & Freitag$^+$ 11] propose treating punctuation prediction as a machine translation task. Yi and Tao adapt the transformer architecture by combining pretrained embeddings created using the word2vec [Mikolov & Chen$^+$ 13] and speech2vec [Chung & Glass 18] models and report a F_1 score of 72.9% on the IWSLT2011 test set [Yi & Tao 19]. Tilk and Alumäe train an LSTM in a two stage approach, first on text data only, and then on text data augmented with pause duration information [Tilk & Alumäe 15].
Szaszák proposes a LSTM based model that performs punctuation prediction based on speech prosody information and reports results on hungarian data [Szaszák 19].
Levy et al. propose a system consisting of several neural networks that utilizes pitch, intensity and pause duration ifnromation in order to detect commas and periods in speech [Levy & Silber-Varod$^+$ 12].

Chapter 4

Experiments

This chapter describes the data used in this work, outlines the different processing steps performed on the data. It also describes the different experiments that were performed on the data, as well as further details on the models that were used and the way in which they were trained.

4.1 Data

The focus of this work is on data from the International Workshop on Spoken Language Translation (IWSLT), which releases a new evaluation campaign every year. The data consists of English recorded speech, English, as well as translated text in various languages [iws 19]. The data stems from a collection of TED talks[1] [iws 19]. The individual TED talks are presentations covering a particular topic, but as a whole they cover a wide variety of topics and domains [Cettolo & Girardi$^+$ 12].

A lot of previous work on punctuation prediction focuses on data from the IWSLT2012 campaign [Che & Wang$^+$ 16] [Ueffing & Bisani$^+$ 13] [Yi & Tao 19]. IWSLT2012 data used in this work was obtained from the evaluation campaign website[2], in particular the english training data from the machine translation track and the tst2011 testset from the ASR track. Statistics about the obtained training and dev datasets based on IWSLT2012 data is shown in table 4.2.
In addition to the discrepancies on statistics of the tst2011 test set reported by Che et al. [Che & Wang$^+$ 16], statistics about the test set obtained deviates from both, the numbers reported be Che et al., as well as the numbers reported by Ueffing et al [Ueffing & Bisani$^+$ 13], as seen in table 4.1.

Due to these problems in reproducing this test set, as well as the impetus to work on a more recent one, this work focuses on data from the IWSLT2019 campaign [iws 19], which was the most recent IWSLT campaign at time of the start of this work.

---

[1] www.ted.com

[2] http://hltc.cs.ust.hk/iwslt/index.php/evaluation-campaign/ted-task.html

Table 4.1: Discrepancies between statistics about the IWSLT2011 test set reported by Ueffing et al., Che et al. and the statistics for the data obtained for this work

	Words	Commas	Periods	Q-Marks
[Che & Wang$^+$ 16]	12626	830	808	46
[Ueffing & Bisani$^+$ 13]	17207	1096	925	84
tst2011.en-fr.en	12306	733	813	46

Table 4.2: Punctuation statistics of obtained IWSLT2012 train and dev data

	Words	Sentences	no punctuation		Commas		Periods		Q-Marks	
	[k]	[k]	[k]	[%]	[k]	[%]	[k]	[%]	[k]	[%]
Train	2080	125	1797.6	86.4	148.0	7.11	12.5	6.0	9.0	0.43
Dev	293	17	253.3	86.4	21.3	7.25	17.3	5.9	1.4	0.48
Test	12	1	10.7	87.1	0.7	5.96	0.8	6.6	0.0	0.37

In particular, this work focuses on the english transcripts, as well as the corresponding audio data from the ”train”, ”dev2010” and ”tst2015” data from the parallel training data and parallel dev data of the IWSLT2019 speech translation task. The transcripts contains upper- and lowercasing, punctuation symbols, and sentence segmentation information.
Statistics about the obtained datasets based on IWSLT2019 data are shown in table 4.3. Test1 is based on the tst2015 dataset parallel dev data, Test2 is based on the tst2015 dataset of the parallel training data of the IWSLT2019 speech translation task. It is noteworthy that around 86.9% of words in Test1 and 87.6% of words in Test2 are not followed by punctuation symbols. This means that performing no punctuation prediction at all would result in CERs of only 13.1% and 12.4% respectively.

Pretrained word embeddings, based on Word2Vec [Mikolov & Chen$^+$ 13] and Speech2Vec [Chung & Glass 18] trained on Librispeech [Panayotov & Chen$^+$ 15] data using the skipgram method with dimension 50, were obtained[3].

---

[3] https://github.com/iamyuanchung/speech2vec-pretrained-vectors

Table 4.3: Punctuation statistics of IWSLT2019 data. Test1 is based on tst2015 from parallel dev data, Test2 is based on tst2015 from parallel training data

	Words	Sentences	no punctuation		Commas		Periods		Q-Marks	
	[k]	[k]	[k]	[%]	[k]	[%]	[k]	[%]	[k]	[%]
Train	2593	171	2256.0	87.0	176.7	6.82	149.1	5.75	10.9	0.42
Dev	17	1	15.2	87.8	1.2	6.73	0.9	5.05	0.1	0.40
Test1	18	1	15.5	86.9	1.3	7.02	1.0	5.57	0.1	0.47
Test2	15	1	13.4	87.6	1.0	6.56	0.8	5.34	0.1	0.51

Table 4.4: Punctuation symbol substitution rules, punctuation symbols that won't be predicted but are grammatically close to commas or periods are substituted in the dataset

Occurence:	Exclamation	Semicolon	Colon	Dash
Replacement:	Period	Period	Comma	Comma

4.2 Preprocessing

4.2.1 Punctuation Tokens

For the preparation of a dataset containing unpunctuated text and punctuation tokens, similar to [Cho & Niehues$^+$ 17], a text or collection of texts is processed by at first applying substitution rules from table 4.4 on punctuation symbols, and then removing all other punctuation symbols that are not part of the set of punctuation symbols to be predicted. Text sequences are not concatenated into one long string of words like described by [Che & Wang$^+$ 16], but are preserved.

In a next step, as a further simplification, after each word all punctuation symbols in excess of the first one following that word and preceding the next word are discarded, so that each word is followed by either one punctuation symbol or none.

Then, as shown in table 4.5, for the source side data, all punctuation symbols are dropped, so that only words remain. For the target side data, tokens corresponding to the words in the input are created. The tokens signify either a blank word, meaning that the corresponding word is not followed by a punctuation symbol, or they signify the particular punctuation symbol that follows that word.

The source vocabulary is then a vocabulary over the words from the text file and the target vocabulary the set of tokens corresponding to the different punctuation symbols to be predicted with the addition of the blank token.

Table 4.5: Data preparation example. First, substitution and omission rules are applied on punctuation symbols. Then, the data is split into a source sequence only consisting of words and a target sequence consisting of punctuation tokens

Raw:	Then	he	said:	”A	storm	is	coming!	We	need	to	find	shelter.”
Sub:	Then	he	said,	A	storm	is	coming.	We	need	to	find	shelter.
Source:	Then	he	said	A	storm	is	coming	We	need	to	find	shelter
Target:	B	B	,	B	B	B	.	B	B	B	B	.

For some experiments, the tokens were pepared in such a way that they also signify whether the corresponding word in the input is uppercase or lowercase, then the target vocabulary consists of each combination of uppercase/lowercase and the predicted pucntuation symbols/blanks.

4.2.2 Byte Pair Encodings

The subword-nmt tool [Sennrich & Haddow$^+$ 16] was used to create byte pair encoded representations, with a set vocabulary size of 10000, of the unpunctuated source data word sequences.

4.2.3 Mel Frequency Cepstrum Coefficients

MFCCs were extracted from the from the audio portion of the ”train” ”dev2010” and ”tst2015” corpora in the parallel training data of the IWSLT2019 speech translation task. Frames with width of 25ms and a 10ms shift between frames were used. Sequence boundaries were kept in order to maintain mapping between sequences of MFCCs and transcript sequences.

4.2.4 Tone

Tone information was estimated using the Snack Sound Toolkit[4] implementation of the AMDF method from the audio portion of the ”train” ”dev2010” and ”tst2015” corpora in the parallel training data of the IWSLT2019 speech translation task. Sequence boundaries were kept in order to maintain mapping between sequences of tone features and transcript sequences.

---

[4] http://www.speech.kth.se/snack/

4.3 Experiment Setups

Experiments were performed with the RETURNN [Doetsch & Zeyer$^+$ 17] framework based on Tensorflow [Abadi & Agarwal$^+$ 15].

4.3.1 Experiments on Text Data

Initially, several experiments were performed on exclusively text-based data from different IWSLT corpora.

Unless specified otherwise, transformer based models were setup with 8 attention heads, a model dimension of 512, feed-forward layers with dimension 2048, and encoder and decoder stacks consisting of 6 layers respectively. Adam optimizer and newbob learning rate control are also used. The embedding layer has been replaced by pretrained W2V and S2V embeddings each with dimension 50, which were then concatenated in the feature dimension to form a combined embedding with dimension 100. The embedding layer is implemented as a lookup table and not updated during training.

Since the length of a target sequence of punctuation tokens is expected to always be equal to the length of the source sequence of words, this information about the targets sequence length is provided to the model during training and search.

All experiments on text data were evaluated on the Test1 dataset or modified versions thereof.

Comparison IWSLT2012 vs IWSLT2019

In order to compare model performance on the obtained IWSLT2012 and IWSLT2019 datasets, a transformer based model was trained on training data and dev data from the IWSLT2012 data prepared as explained above, while another was trained on training data and dev data from the IWSLT2019 data prepared in the same manner. Then both were evaluated on the tst2011 test set from the IWSLT2012 data, as well as the tst2015 test set from the IWSLT2019 data.

Rechunking

Training a model on word sequences that maintain correct sentence segmentation, where the end of a word sequence coincides with the end of the corresponding sentence, could potentially lead to the model learning to associate the end of a sequence with punctuation symbols that occur at the end of sentences such as periods or questionmarks.

In order to measure the extent to which this occurs, several models are trained and evaluated on data which had it's sentence segmentation information removed. This is achieved by first concatenating all chunks of the respective dataset together into one long chunk, and then cutting it back down into smaller chunks of either a fixed size, or a random size between a minimum and a maximum value.

A unrechunked dataset was prepared, retaining the original sentence segmentation information.
A rechunked dataset was prepared by separately rechunking the original training dataset twice, first to random chunk sizes between 25 and 50 and second to random chunk sizes between 50 and 75. The two resulting rechunked datasets are then combined.
In order to adjust for the rechunked dataset containing the original data twice, the EpochSplit parameter was adjusted, it determines the size of the fraction of the dataset seen during one training subepoch. An EpochSplit parameter of 6 means that one sixth of the dataset is seen during one subepoch.
One model was trained on the unrechunked dataset with the EpochSplit parameter set to 3. Another model was trained on the rechunked dataset with EpochSplit parameter set to 6.

In a next experiment, the effect of rechunking the test set was investigated.
For this purpose, several test sets were prepared by rechunking the original test set with different fixed chunk sizes between 15 and 90. A training set was prepared by rechunking the original dataset to random chunksizes between 50 and 75. A model was then trained on the training with EpochSplit parameter set to 3, and evaluated on the different test sets.

Finally, the difference between removing the sentence segmentation information by rechunking, and keeping the sentence segmentation information, was investigated.
For this purpose, two models were evaluated. One was trained on the unrechunked training set. The other one was trained on the rechunked dataset resulting from the combination of rechunking the original dataset twice. The dev sets were prepared accordingly to the training sets. The models were then evaluated on both the unrechunked test set and the test set rechunked to fixed chunk sizes of 65.

Casing

Several Experiments were performed to evaluate the impact of leaving or removing upper- and lowercase information from the source data.

Table 4.6: Data preparation example for tokens including case information.

Raw:	Then	he	said:	"A	storm	is	coming!	We	need	to	find	shelter."
Sub:	Then	he	said,	A	storm	is	coming.	We	need	to	find	shelter.
Source:	Then	he	said	A	storm	is	coming	We	need	to	find	shelter
Target:	U	L	L,	U	L	L	L.	U	L	L	L	L.

For this purpose, several datasets were prepared.

The first dataset, containing no case information in the source data, and no case information in the target tokens, was prepared by turning all words in the source data into lowercase words. Punctuation tokens in the target data were kept the same.

A second dataset was prepared, containing no case information in the source data, but containing case information in the target data. All the words in the source data were turned into lowercase words. Punctuation tokens in the target data were replaced with new tokens. These new tokens carry two kinds of information, information about punctuation symbols following the word corresponding to the token, but also the information whether the word is uppercase or lowercase [Cho & Niehues$^+$ 17].
Each of the normal punctuation tokens in the vocubalary is replaced with two new classes of tokens, an uppercase version and a lowercase version of the previous classes. Predicting these tokens then predicts both punctuation, and the casing of the input sequences.
An example of the creation of these tokens is shown in table 4.3.1.

A third dataset was prepared, keeping the case information in the source data, and punctuation tokens in the target data.

For each of the three datasets, a model was trained on the dataset and evaluated on a test dataset prepared in the same manner.

Embeddings

Using the skipgram word2vec implementation [Mikolov & Chen$^+$ 13] of the gensim library[5], a word2vec model was trained on IWSLT2019 training data transcripts. The pretrained W2V embeddings in the W2VS2V transformer model were then

---

[5] https://radimrehurek.com/gensim/

replaced with these new embeddings.
The model was then trained on the combination of data rechunked to chunk sizes between 25 and 50 and data rechunked to chunk sizes between 50 and 75 and evaluated on test data rechunked to a fixed chunk size of 65.

4.3.2 Incorporating audio features

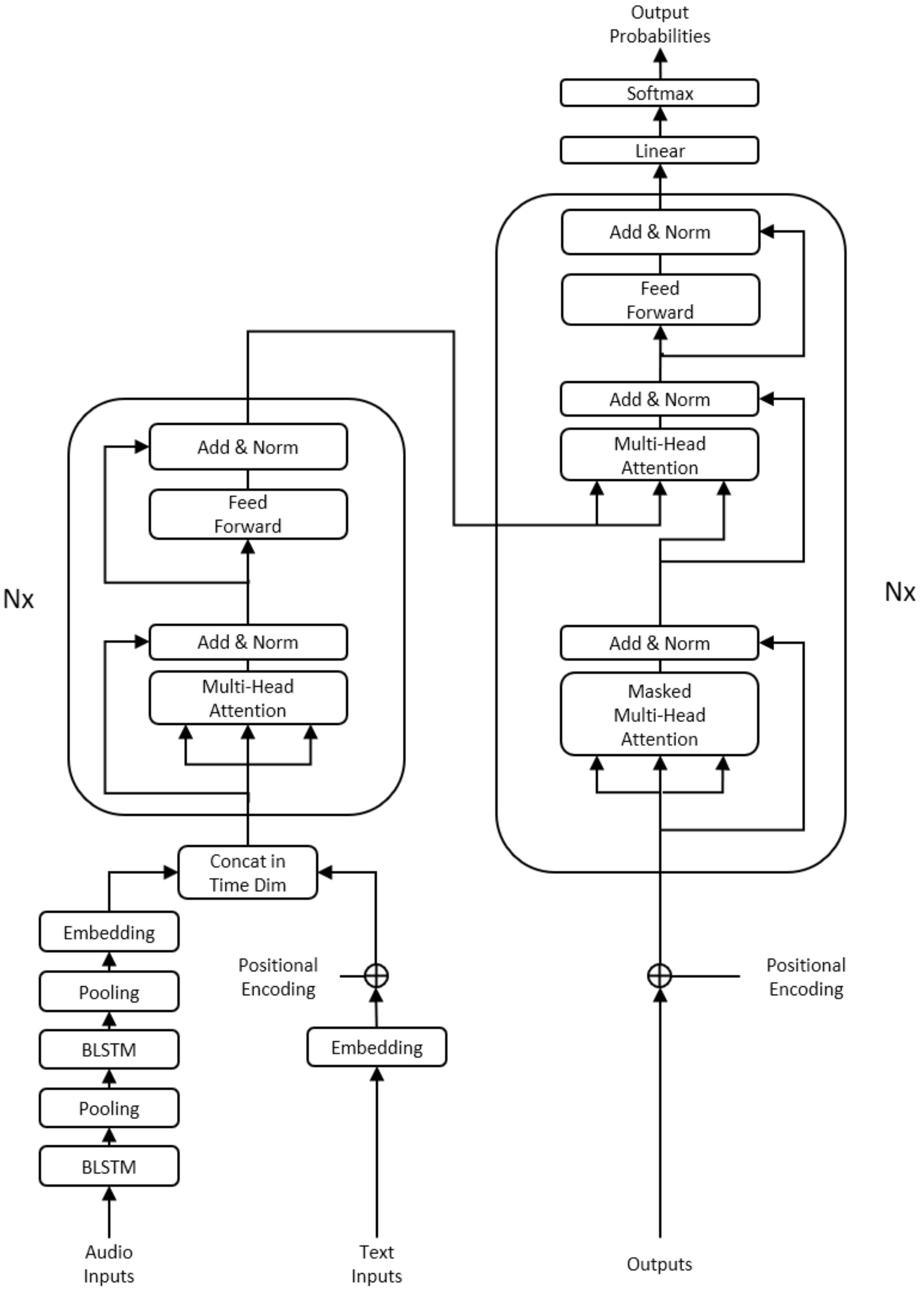
Several experiments were performed investigating whether punctuation prediction performance could be improved by predicting punctuation tokens based on audio features, such as MFCCs or tone features.

In order to investigate the impact of supplementing text data with tone features or MFCCs in the time dimension, the architecture shown in figure 4.3.2 was created, based on the transformer architecture [Vaswani & Shazeer^{+} 17].

The input consists of two parts, text based features, such as words or bpe tokens, and audio features, extracted from speech corresponding to the text.
The text based features are first linearly projected into a vector representation serving as input embedding for the text, and positional encoding is applied. The audio data is fed into a two layer BLSTM with pooling that serves as a substitute for positional encoding. The output of that two layer BLSTM is then linearly projected into a vector representation of the same size as the vector representation of the text.
The corresponding sequences of vector representations of audio and text data are then concatenated in the time dimension. The resulting combined sequence of vector representations is then fed into a otherwise normal transformer architecture. The targets are punctuation tokens. Information about the target sequence length was provided to the model during training and search.

The transformer part of the architecture is set up with 5 encoder layers and 5 decoder layers, model dimension size $d_{model} = 512$, dimension size of the feed forward sublayers of 2048, and 8 attention heads. Due to the low size of the target vocabulary, the output embedding is omitted.
Using this architecture, a model was trained using MFCCs as audio features, another was trained using tone features as audio features.
Additionally, two models were trained on a modified architecture that omits the concatenation of embedded text data, one with MFCCs and one with tone features.
A fifth model was trained on a modified architecture that omits the concatenation of embedded audio data.
For evaluation of these 5 models, the Test2 dataset is used.

Figure 4.1: Architecture of model trained on combined audio and text data



Chapter 5

Experiment Results

The following chapter presents the results of the previously described experiments.

5.1 Experiments on Text Data

First, results of the experiments performed on the models trained on textual data, described in section 4.3.1 are presented and discussed.

5.1.1 Comparison IWSLT2012 vs IWSLT2019

Table 5.1 shows the results of models trained on IWSLT2012 training and dev data, evaluated on both the obtained tst2011 test set and the tst2015 test set. Performance on comma- and period- punctuation token classes is similar, with a slightly better performance on the iwslt2011 test set, while there is a higher recall for the question mark punctuation token class on the tst2015 test set. Because question marks only make up a small fraction of the occurrences of punctuation tokens in the reference, overall scores are similar.

Table 5.2 shows the results of models trained on IWSLT2019 training and dev data, evaluated on both the obtained tst2011 test set and the tst2015 test set. There is higher precision for the comma punctuation token class on the tst2015 test set, but in turn higher precission for the period punctuation token class on the tst2011 test set. Performance for the question mark punctuation token class is again higher on the tst2015 test set. Overall scores are again similar.

These two experiments show that no large difference in overall performance is expected when exchanging the two test sets, however performance on question marks is higher when using the tst2015 test set.

5.1.2 Rechunking

The results of the experiment contrasting the model trained on rechunked data with the model trained on unrechunked data, both evaluated on unrechunked test data

Table 5.1: Models trained on IWSLT2012 data evaluated on iwslt2011 and iwslt2015 test sets

Testset	CER	SER	Overall		
			P	R	F1
	[%]	[%]	[%]	[%]	[%]
tst2011	5.1	39.4	82.1	74.0	77.9
tst2015	5.5	42.9	82.5	70.5	76.0

Testset	Comma			Period			Questionmark		
	P	R	F1	P	R	F1	P	R	F1
	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
tst2011	64.0	53.6	58.4	96.1	93.5	94.8	86.2	54.3	66.7
tst2015	70.5	49.8	58.3	92.1	96.1	94.0	87.7	67.9	76.5

Table 5.2: Models trained on IWSLT2019 data evaluated on iwslt2011 and iwslt2015 test sets

Model	Testset	CER	SER	Overall		
				P	R	F1
		[%]	[%]	[%]	[%]	[%]
W2VS2V	tst2011	5.0	38.9	80.0	77.0	78.5
W2VS2V	tst2015	5.1	39.9	80.9	75.9	78.3

Model	Testset	Comma			Period			Questionmark		
		P	R	F1	P	R	F1	P	R	F1
		[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
W2VS2V	tst2011	62.6	59.2	60.9	95.1	93.7	94.4	78.9	65.2	71.4
W2VS2V	tst2015	71.1	60.6	65.4	90.5	95.9	93.1	84.4	77.4	80.7

Table 5.3: Models trained on unrechunked and rechunked training data, evaluated on unrechunked test data. Difference in performance is especially high for periods and questionmarks

Train &Dev	CER	SER	Overall		
			P	R	F1
	[%]	[%]	[%]	[%]	[%]
no rechunking	5.1	39.9	80.9	75.9	78.3
rechunking	11.0	85.3	58.6	28.0	37.9

Train &Dev	Comma			Period			Questionmark		
	P	R	F1	P	R	F1	P	R	F1
	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
no rechunking	71.1	60.6	65.4	90.5	95.9	93.1	84.4	77.4	80.7
rechunking	63.6	50.8	56.5	20.7	2.5	4.4	12.5	1.2	2.2

are shown in table 5.3.
The rechunked dataset here was achieved by rechunking the original dataset twice, to random chunk sizes between 25 and 50 and to random chunk sizes between 50 and 75 and then combining the two resulting datasets.

The results show that the model trained on rechunked data performs very poorly, it almost completely fails to predict periods with a Recall of only 2.5% as well as questionmarks with a Recall of only 1.2%, both of which occur predominantly at the end of sequences in the unrechunked test data, while it somewhat succeeds to predict commas, though still worse than the model trained on unrechunked data. The model trained on unrechunked training data expectedly performs extremely well in predicting periods and to a lesser extent questionmarks.

The results of the opposite case, using test data that was rechunked to remove sentence boundary information, shown in table 5.4 shows that the model trained on unrechunked training data performs worse across the board, and has particular difficulties with questionmarks. It's performance on commas especially is much worse compared to the performance of the other model in the previous experiment. It has some success predicting periods but is vastly outperformed by the model trained on rechunked data. The model trained on rechunked data beats the model trained on unrechunked data in all categories, though it is noteworthy that it's performance overall is slightly worse than the performance of the unrechunked model in the pre-

Table 5.4: Models trained on unrechunked and rechunked training data, evaluated on rechunked test data. Precision for periods remains high for model trained on unrechunked training data

Rechunking	CER	SER	Overall		
			P	R	F1
	[%]	[%]	[%]	[%]	[%]
without	9.4	73.1	71.3	37.8	49.4
with	5.5	42.8	80.8	70.0	75.0

Rechunking	Comma			Period			Questionmark		
	P	R	F1	P	R	F1	P	R	F1
	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
without	60.2	33.2	42.8	85.7	46.7	60.5	53.3	9.5	16.2
with	75.3	54.0	62.9	85.9	91.9	88.8	74.3	61.9	67.5

vious experiment.

When considering the results of both of these experiments together, they suggest that when considering whether to train a model on rechunked or unrechunked data in the case that sentence boundary information is present in the training dataset, it is important to also consider what kind of data the model will later be evaluated on.
In the case that sentence boundary information is expected to be present in the test data, it might be more advantageous to train the model on unrechunked data. The superior performance of the model trained on unrechunked data but evaluated on rechunked data as opposed to the model trained on rechunked data but evaluated on unrechunked data additionally supports this.

The results in table 5.5 of the experiment on the effect of different chunk sizes for the test set while keeping the model the same, shows that the performance of the model peaks for test data chunksizes around 60, and slowly drops off for larger or smaller chunk sizes of the test data. This happens to roughly align with the average of the upper and lower boundaries 50 and 75 for the random chunk sizes of the training data that the model was trained on.
This suggests that chunk size of the test data is an important hyperparameter that needs to be considered when operating on data with unknown sentence boundaries.

Table 5.5: Evaluation of the same model on tst2015 test set rechunked to different fixed chunk sizes. Performance is best at chunk sizes 60 and 65, which aligns with the average chunk size of the training data of 62.5

Test Chunksize	CER	SER	Overall		
			P	R	F1
	[%]	[%]	[%]	[%]	[%]
15	7.1	55.3	72.8	61.3	66.5
25	6.2	48.3	77.7	66.1	71.5
50	5.6	44.1	80.4	68.9	74.3
55	5.5	43.1	80.5	69.9	74.8
60	5.4	42.4	81.4	70.2	75.4
65	5.5	42.8	80.8	70.0	75.0
70	5.5	43.1	80.6	69.6	74.7
75	5.6	43.4	80.8	69.0	74.4
80	6.2	48.6	78.6	65.6	71.5
90	7.1	55.6	77.5	58.2	66.5

Test Chunksize	Comma			Period			Questionmark		
	P	R	F1	P	R	F1	P	R	F1
	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
15	63.5	46.3	53.5	81.5	84.6	83.0	53.3	19.0	28.1
25	70.6	51.2	59.4	84.1	87.5	85.8	72.2	46.4	56.5
50	74.7	53.0	62.0	85.6	90.7	88.1	75.0	60.7	67.1
55	75.2	54.6	63.3	85.2	91.3	88.2	74.6	56.0	63.9
60	76.2	53.9	63.1	86.3	92.4	89.2	71.8	60.7	65.8
65	75.3	54.0	62.9	85.9	91.9	88.8	74.3	61.9	67.5
70	75.0	53.5	62.4	85.7	91.3	88.4	75.0	64.3	69.2
75	75.1	52.6	61.8	86.2	91.4	88.7	71.4	59.5	64.9
80	73.6	50.2	59.7	83.3	86.0	84.6	71.2	61.9	66.2
90	74.5	43.6	55.0	79.9	78.1	79.0	75.5	47.6	58.4

5.1.3 Casing

Table 5.6 shows the results of the experiment about case prediction. One model was trained and evaluated with case information removed from the sources and which has punctuation tokens as targets. Another model was trained and evaluated on data with case information removed from the sources, but which had the token target classes amended with information about the casing. And the third model was trained and evaluated normally with case information in the sources and which has punctuation tokens as targets.

Table 5.6: Comparison between model trained on datasets prepared differently regarding case information. Models marked "no" regarding casing in source were trained on data that had all words in the source turned into lowercase. Models marked "no" regarding casing in target data were trained on data with punctuation tokens. Models marked "yes" regarding casing in targets were trained on data in which the punctuation tokens in the targets were replaced with tokens that carry both punctuation information and information about the case of the corresponding word.

casing in source	casing in targets	CER	SER	Overall		
				P	R	F1
		[%]	[%]	[%]	[%]	[%]
no	no	8.6	65.7	62.5	46.3	53.2
no	yes	11.6	51.7	65.1	54.3	59.2
yes	no	5.5	42.8	80.8	70.0	75.0

casing in source	casing in targets	Comma			Period			Questionmark		
		P	R	F1	P	R	F1	P	R	F1
		[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
no	no	63.7	36.7	46.6	61.5	59.0	60.2	65.3	38.1	48.1
no	yes	64.9	46.4	54.1	65.2	64.4	64.9	67.1	53.6	59.6
yes	no	75.3	54.0	62.9	85.9	91.9	88.8	74.3	61.9	67.5

casing in source	casing in targets	Lowercase			Uppercase		
		P	R	F1	P	R	F1
		[%]	[%]	[%]	[%]	[%]	[%]
no	no	-	-	-	-	-	-
no	yes	97.1	98.1	97.6	81.7	74.9	78.1
yes	no	-	-	-	-	-	-

The results show that the model that had neither case information in the source data, nor in the target data, with resulting confusion matrix in table 5.7 performed the worst across all metrics except classification error rate.

Table 5.7: Confusion matrix for model trained and evaluated on data without case information in source or target data

		Hyp				
		Blank	Period	Comma	Questionmark	Sum:
Ref	Blank	15223	128	140	10	15501
	Period	290	586	117	1	994
	Comma	564	222	460	6	1252
	Questionmark	30	17	5	32	84
	Sum:	16107	953	722	49	

Table 5.8: Confusion matrix of model trained on input data without case information and target tokens amended by case information. "L" signifies lowercase, "U" signifies uppercase. Classes "L" and "U" represent lowercase and uppercase blank words respectively while classes "L." "L," "L?" "U." "U," "U?" represent a lowercase or uppercase word followed by the respective punctuation symbol

		Hyp								
		L	U	L.	L,	L?	U.	U,	U?	Sum:
Ref	L	13346	263	98	148	11	1	6	0	13873
	U	384	1201	12	6	2	7	16	0	1628
	L.	193	10	598	111	1	6	5	0	924
	L,	403	12	181	493	7	1	6	0	1103
	L?	20	0	13	4	43	0	0	0	80
	U.	7	10	10	8	0	26	9	0	70
	U,	16	22	11	5	1	17	77	0	149
	U?	0	1	0	1	1	0	0	1	4
	Sum:	14369	1519	923	776	66	58	119	1	

Table 5.9: Comparison of W2VS2V transformer models with pretrained and self trained W2V embeddings

W2V training	CER	SER	Overall		
			P	R	F1
[epochs]	[%]	[%]	[%]	[%]	[%]
pretrained	5.5	42.8	80.8	70.0	75.0
10	5.7	43.5	80.6	68.2	73.8
100	5.9	44.9	78.6	69.1	73.5
1000	5.6	43.2	80.6	68.5	74.0

W2V training	Comma			Period			Questionmark		
	P	R	F1	P	R	F1	P	R	F1
[epochs]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
pretrained	75.3	54.0	62.9	85.9	91.9	88.8	74.3	61.9	67.5
10	74.4	51.7	61.0	86.2	88.8	87.5	75.3	69.0	72.0
100	72.0	52.8	60.9	84.8	90.4	87.5	71.0	58.3	64.1
1000	75.5	51.2	61.0	85.1	90.7	87.8	74.3	61.9	67.5

The model which had no case information in the source data, but had case information in the target data, and can therefore predict the case of words from text without case information in addition to predicting punctuation, performs better then the model with case information in neither source nor target data, with the exception of the classification error metric. The bad performance on classification error rate could stem from the fact that this model can make classifcation errors between an upper case blank and a lower case blank token, which the other models cannot. The resulting confusion matrix can be seen in table 5.8

The model trained with case information in the source data outperforms the other models, suggesting that case information contributes significantly to the punctuation prediction performance. It performs especially better on the prediction of periods, with a possible reason being that an uppercase word marks the beginning of a sentence in the english language, which often coincides with a period at the end of the previous one.

Table 5.10: Confusion matrix for W2VS2V transformer model with W2V embeddings trained for 1000 epochs

		Hyp				
		Blank	Period	Comma	Questionmark	Sum:
Ref	Blank	15229	62	197	13	15501
	Period	77	902	11	4	994
	Comma	539	71	641	1	1252
	Questionmark	7	25	0	52	84
	Sum:	15852	1060	849	70	

Table 5.11: Confusion matrix for W2VS2V transformer model with W2V embeddings trained for 100 epochs

		Hyp				
		Blank	Period	Comma	Questionmark	Sum:
Ref	Blank	15175	68	246	12	15501
	Period	77	899	11	7	994
	Comma	525	65	661	1	1252
	Questionmark	7	28	0	49	84
	Sum:	15784	1060	918	69	

Table 5.12: Confusion matrix for W2VS2V transformer model with W2V embeddings trained for 10 epochs

		Hyp				
		Blank	Period	Comma	Questionmark	Sum:
Ref	Blank	15229	56	204	12	15501
	Period	86	883	19	6	994
	Comma	540	64	647	1	1252
	Questionmark	5	21	0	58	84
	Sum:	15860	1024	870	77	

5.1.4 Embeddings

The results in table 5.9 of training a W2VS2V transformer model with W2V embeddings trained on IWSLT2019 data show that the self trained W2V embeddings are almost as good as the pretrained embeddings based on librispeech data. Training the W2V model for even longer might have the potential to provide even better results.

5.1.5 Comparison with other groups

Table 5.13: Comparison best own models with results from literature on tst2011 test set. SBI stands for whether sentence boundary information was left intact during training and search. W2VS2V1 was trained on unrechunked IWSLT2012 data. W2VS2V2 was trained on rechunked IWSLT2019 data and evaluated on tst2011 test set rechunked to fixed chunksize of 65. *: results from literature. **: removal of sentence boundary information unclear. ***: the W2VS2V models trained here deviate from [Yi & Tao 19] in that the embedded feature vectors are concatenated, not added.

Model	SBI	CER	SER	Overall		
				P	R	F1
		[%]	[%]	[%]	[%]	[%]
[Wang & Chen^{+} 18]*	**	-	-	78.2	74.4	77.4
[Yi & Tao 19]*	removed	-	-	76.7	69.6	72.9
W2VS2V1***	intact	5.5	42.8	80.8	70.0	75.0
W2VS2V2***	removed	5.3	40.8	80.3	72.0	76.0

Model	SBI	Comma			Period			Questionmark		
		P	R	F1	P	R	F1	P	R	F1
		[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
[Wang & Chen^{+} 18]*	**	57.2	50.8	55.9	96.7	97.3	96.8	70.6	69.2	70.3
[Yi & Tao 19]*	removed	67.4	61.1	64.1	82.5	77.4	79.9	80.1	70.2	74.8
W2VS2V1***	intact	75.3	54.0	62.9	85.9	91.9	88.8	74.3	61.9	67.5
W2VS2V2***	removed	67.7	54.2	60.2	89.7	88.6	89.1	76.9	65.2	70.6

In table 5.13, the performance of two W2VS2V based models W2VS2V1 and W2VS2V2, is compared on the tst2011 test set with results from literature. W2VS2V1 was trained on unrechunked IWSLT2012 data and evaluated on the obtained unrechunked tst2011 test set.W2VS2V2 was trained on IWSLT2019 data rechunked twice to random chunk sizes between 25 and 50 as well as between 50 and 75, evaluated on the obtained tst2011 test set rechunked to fixed chunk size of 65. Both models outperform the results reported in the W2VS2V paper [Yi & Tao 19] regarding performance on the period class and overall scores, while performing worse on the comma and question mark classes with lower recall. The results reported by [Wang & Chen$^+$ 18] dominate in the period class measures, but are slightly outperformed in the comma and question mark classes.

Table 5.14: Current best own results on IWSLT2019 data. SBI stands for whether sentence boundary information was left intact during training and search. W2VS2V3 was trained on unrechunked IWSLT2019 data and evaluated on unrechunked tst2015(Test1) test set. W2VS2V4 was trained on rechunked IWSLT2019 data and evaluated on tst2015(Test1) test set rechunked to fixed length of 60. ***: the W2VS2V models trained here deviate from [Yi & Tao 19] in that the embedded feature vectors are concatenated, not added.

Model	SBI	CER	SER	Overall		
				P	R	F1
		[%]	[%]	[%]	[%]	[%]
W2VS2V3***	intact	5.1	39.9	80.9	75.9	78.3
W2VS2V4***	removed	5.4	42.4	81.4	70.2	75.4

Model	SBI	Comma			Period			Questionmark		
		P	R	F1	P	R	F1	P	R	F1
		[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
W2VS2V3***	intact	71.1	60.6	65.4	90.5	95.9	93.1	84.4	77.4	80.7
W2VS2V4***	removed	76.2	53.9	63.1	86.3	92.4	89.2	71.8	60.7	65.8

In table 5.14 the performance of the two best performing W2VS2V models on IWSLT2019 data is displayed. W2VS2V3 was trained on unrechunked IWSLT2019 data and evaluated on unrechunked tst2015(Test1) test set, while W2VS2V4 was trained on IWSLT2019 data rechunked twice to random chunk sizes between 25 and 50 as well as between 50 and 75, and evaluated on the tst2015(Test1) test set rechunked to a fixed chunk size of 60.

Table 5.15: Comparison of models trained exclusively on audio data, models trained on a combination of audio and text data, and a model only trained on text data

Model	CER	SER	Overall		
			P	R	F1
	[%]	[%]	[%]	[%]	[%]
mfcc	12.8	103.8	12.8	0.7	1.3
tone	12.3	100.0	-	0.0	-
bpe+mfcc	8.9	72.6	66.1	42.6	51.8
bpe+tone	7.4	60.5	67.2	61.5	64.2
bpe	7.1	58.0	67.9	65.2	66.5

Model	Comma			Period			Questionmark		
	P	R	F1	P	R	F1	P	R	F1
	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]	[%]
mfcc	-	0.0	-	12.8	1.6	2.9	-	0.0	-
tone	-	0.0	-	-	0.0	-	-	0.0	-
bpe+mfcc	63.4	16.7	26.4	67.5	76.3	71.6	51.4	23.4	32.1
bpe+tone	65.5	37.0	47.2	68.8	92.2	78.8	56.8	54.5	55.6
bpe	64.5	43.0	51.6	70.1	92.4	79.7	66.7	64.9	65.8

5.2 Incorporating audio features

The following section presents the results of the experiments including audio features, which were performed as described in section 4.3.2.

Due to time constraints, the training of models involving mfcc and tone features has not concluded, the reported results represent the best models after 230 subepochs or 23 full epochs, compared to 50 epochs in the previous experiments involving only text.
For this reason, and because the networks trained on audio features have only 5 transformer encoder and decoder layers, as opposed to the 6 layers in previous experiments, results are compared with the results of a separate model. This model was trained for 23 full epochs. The neural network architecture and the training hyperparameters of this model are identical to the model trained on a combination of bpe and tone features, except that the concatenation of the embedded tone fea-

Table 5.16: Results of models exclusively trained on tone features, all predicted outputs were of the blank token class, suggesting that tone features alone do not provide enough information for punctuation prediction.

		Hyp				
		Blank	Period	Comma	Questionmark	Sum:
Ref	Blank	12128	0	0	0	12128
	Period	733	0	0	0	733
	Comma	893	0	0	0	893
	Questionmark	77	0	0	0	77
	Sum:	13831	0	0	0	

Table 5.17: Confusion matrix of model exclusively trained on MFCCs

		Hyp				
		Blank	Period	Comma	Questionmark	Sum:
Ref	Blank	12050	78	0	0	12128
	Period	721	12	0	0	733
	Comma	892	1	0	0	893
	Questionmark	74	3	0	0	77
	Sum:	13737	94	0	0	

tures onto the embedded bpe features is omitted.

The results can be seen in table 5.15, the corresponding confusion matrices in tables 5.16 to 5.20

The model trained exclusively on tone data failed to converge beyond predicting a complete lack of punctuation symbols.
The model trained exclusively on MFCC data failed to converge beyond predicting a lack of punctuation symbols with the exception of a small amount of periods with a recall of only 0.7%. The CER of 12.8% and the SER of 103.8% suggest that this model performs worse on these measures than doing no punctuation prediction at all.

The model trained on a combination of bpe and tone features performed better overall compared to the model, especially when predicting commas and questionmarks.

Table 5.18: Results of model trained on a combination of byte pair encoded text and tone features.

		Hyp				
		Blank	Period	Comma	Questionmark	Sum:
Ref	Blank	11753	185	167	23	12128
	Period	44	676	7	6	733
	Comma	464	96	330	3	893
	Questionmark	10	25	0	42	77
	Sum:	12271	982	504	74	

Table 5.19: Results of model trained on a combination of byte pair encoded text and MFCCs

		Hyp				
		Blank	Period	Comma	Questionmark	Sum:
Ref	Blank	11868	171	81	8	12128
	Period	162	559	5	7	733
	Comma	674	68	149	2	893
	Questionmark	29	30	0	18	77
	Sum:	12733	828	235	35	

Both models trained on a combination between audio and text however performed worse than the comparison model where the audio features were omitted.

Table 5.20: Results of model trained exclusively on byte pair encoded text

		Hyp				
		Blank	Period	Comma	Questionmark	Sum:
Ref	Blank	11733	174	202	19	12128
	Period	42	677	9	5	733
	Comma	411	97	384	1	893
	Questionmark	9	18	0	50	77
	Sum:	12195	966	595	75	

Chapter 6

Conclusion

This work investigated multiple approaches to improve the performance of punctuation prediction models on textual ASR output and speech data. Some insights were gained with regard to the necessisity or lack thereof of rechunking training data that contains sentence segmentation information depending on whether the test data can be expected to also include such information or not.

It was further shown, that the existence or lack of casing information can have an impact on performance and that performance on punctuation prediction on sources with no casing information could possibly improved by amending the punctuation tokens with target case information.

Finally, an attempt was made to improve upon punctuation prediction performance on text by concatenating embedded speech data to the embedded text data in the time dimension, altough this hasn't been fruitful yet.

Future work could include evaluating the impact of replacing the pretrained S2V input embeddings based on librispeech data with S2V embeddings trained on IWSLT data. The impact of further training the W2V model on IWSLT data could be investigated. When training models on a combination of text and speech data, instead of using embeddings learned by the model, the W2VS2V approach could be used to project the text inputs into the W2V embedding space and the speech inputs into the S2V embedding space.

Another question is whether the approach to concatenate the embedded features in the time dimension could be improved upon by explicitly defining a distinction between the embedded audio and text features, for example by introducing a token embedding that marks the separation between the time sequence of embedded audio features and the time sequence of embedded text features. Restricting the vector spaces of embedded audio data and embedded text data to be disjoint from each other, mimicking the approach to concat W2V and S2V embeddings in the feature dimension, could be another approach.

It could be investigated to what extent the impact seen when rechunking text data is present when combining audio and text data. For this purpose, an alignment between speech and text could be applied. Then both text and speech could be rechunked together.

Appendix A

Appendix

List of Figures

2.1 LSTM Unit 6
2.2 Transformer architecture from: [Vaswani & Shazeer$^+$ 17] 9
2.3 Continuous bag of words (CBOW) and Skipgram Configurations of Word2vec. From: [Mikolov & Chen$^+$ 13] 10

4.1 Architecture of model trained on combined audio and text data . . . 27

List of Tables

2.1 Example confusion matrix 13

4.1 Discrepancies between statistics about the IWSLT2011 test set reported by Ueffing et al., Che et al. and the statistics for the data obtained for this work 20
4.2 Punctuation statistics of obtained IWSLT2012 train and dev data . 20
4.3 Punctuation statistics of IWSLT2019 data. Test1 is based on tst2015 from parallel dev data, Test2 is based on tst2015 from parallel training data 21
4.4 Punctuation symbol substitution rules, punctuation symbols that won't be predicted but are grammatically close to commas or periods are substituted in the dataset 21
4.5 Data preparation example. First, substitution and omission rules are applied on punctuation symbols. Then, the data is split into a source sequence only consisting of words and a target sequence consisting of punctuation tokens 22
4.6 Data preparation example for tokens including case information. . . 25

5.1 Models trained on IWSLT2012 data evaluated on iwslt2011 and iwslt2015 test sets 30
5.2 Models trained on IWSLT2019 data evaluated on iwslt2011 and iwslt2015 test sets 30
5.3 Models trained on unrechunked and rechunked training data, evaluated on unrechunked test data. Difference in performance is especially high for periods and questionmarks 31
5.4 Models trained on unrechunked and rechunked training data, evaluated on rechunked test data. Precision for periods remains high for model trained on unrechunked training data 32
5.5 Evaluation of the same model on tst2015 test set rechunked to different fixed chunk sizes. Performance is best at chunk sizes 60 and 65, which aligns with the average chunk size of the training data of 62.5 33

5.6 Comparison between model trained on datasets prepared differently regarding case information. Models marked ”no” regarding casing in source were trained on data that had all words in the source turned into lowercase. Models marked ”no” regarding casing in target data were trained on data with punctuation tokens. Models marked ”yes” regarding casing in targets were trained on data in which the punctuation tokens in the targets were replaced with tokens that carry both punctuation information and information about the case of the corresponding word. 34

5.7 Confusion matrix for model trained and evaluated on data without case information in source or target data 35

5.8 Confusion matrix of model trained on input data without case information and target tokens amended by case information. ”L” signifies lowercase, ”U” signifies uppercase. Classes ”L” and ”U” represent lowercase and uppercase blank words respectively while classes ”L.” ”L,” ”L?” ”U.” ”U,” ”U?” represent a lowercase or uppercase word followed by the respective punctuation symbol 35

5.9 Comparison of W2VS2V transformer models with pretrained and self trained W2V embeddings 36

5.10 Confusion matrix for W2VS2V transformer model with W2V embeddings trained for 1000 epochs 37

5.11 Confusion matrix for W2VS2V transformer model with W2V embeddings trained for 100 epochs 37

5.12 Confusion matrix for W2VS2V transformer model with W2V embeddings trained for 10 epochs 37

5.13 Comparison best own models with results from literature on tst2011 test set. SBI stands for whether sentence boundary information was left intact during training and search. W2VS2V1 was trained on unrechunked IWSLT2012 data. W2VS2V2 was trained on rechunked IWSLT2019 data and evaluated on tst2011 test set rechunked to fixed chunksize of 65. ⋆: results from literature. ⋆⋆: removal of sentence boundary information unclear. ⋆⋆⋆: the W2VS2V models trained here deviate from [Yi & Tao 19] in that the embedded feature vectors are concatenated, not added. 38

5.14 Current best own results on IWSLT2019 data. SBI stands for whether sentence boundary information was left intact during training and search. W2VS2V3 was trained on unrechunked IWSLT2019 data and evaluated on unrechunked tst2015(Test1) test set. W2VS2V4 was trained on rechunked IWSLT2019 data and evaluated on tst2015(Test1) test set rechunked to fixed length of 60. ***: the W2VS2V models trained here deviate from [Yi & Tao 19] in that the embedded feature vectors are concatenated, not added. 39
5.15 Comparison of models trained exclusively on audio data, models trained on a combination of audio and text data, and a model only trained on text data 40
5.16 Results of models exclusively trained on tone features, all predicted outputs were of the blank token class, suggesting that tone features alone do not provide enough information for punctuation prediction. 41
5.17 Confusion matrix of model exclusively trained on MFCCs 41
5.18 Results of model trained on a combination of byte pair encoded text and tone features. 42
5.19 Results of model trained on a combination of byte pair encoded text and MFCCs 42
5.20 Results of model trained exclusively on byte pair encoded text . . . 43

Bibliography

[Abadi & Agarwal$^+$ 15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[Bishop 06] C. M. Bishop. *Pattern recognition and machine learning.* springer, 2006.

[Blandford & Vanderdonckt$^+$ 11] A. Blandford, J. Vanderdonckt, P. Gray. *People and Computers XV—Interaction without Frontiers: Joint Proceedings of HCI 2001 and IHM 2001.* Springer Science & Business Media, 2011.

[Bodén 01] M. Bodén. A guide to recurrent neural networks and backpropagation, 2001.

[Boháč & Rott$^+$ 17] M. Boháč, M. Rott, V. Kovář. Text punctuation: An inter-annotator agreement study. In *International Conference on Text, Speech, and Dialogue*, pp. 120–128. Springer, 2017.

[Cettolo & Girardi$^+$ 12] M. Cettolo, C. Girardi, M. Federico. Wit3: Web inventory of transcribed and translated talks. In *Conference of european association for machine translation*, pp. 261–268, 2012.

[Che & Wang$^+$ 16] X. Che, C. Wang, H. Yang, C. Meinel. Punctuation prediction for unsegmented transcript based on word vector. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pp. 654–658, 2016.

[Chen 99] C. J. Chen. Speech recognition with automatic punctuation. In *Sixth European Conference on Speech Communication and Technology*, 1999.

[Cho & Niehues$^+$ 12] E. Cho, J. Niehues, A. Waibel. Segmentation and punctuation prediction in speech language translation using a monolingual translation system. In *International Workshop on Spoken Language Translation (IWSLT) 2012*, 2012.

[Cho & Niehues$^+$ 17] E. Cho, J. Niehues, A. Waibel. Nmt-based segmentation and punctuation insertion for real-time spoken language translation. In *INTERSPEECH*, pp. 2645–2649, 2017.

[Christensen & Gotoh$^+$ 01] H. Christensen, Y. Gotoh, S. Renals. Punctuation annotation using statistical prosody models. In *ISCA tutorial and research workshop (ITRW) on prosody in speech recognition and understanding*, 2001.

[Chung & Glass 18] Y. Chung, J. R. Glass. Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech. *CoRR*, Vol. abs/1803.08976, 2018.

[Davis & Mermelstein 80] S. Davis, P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE transactions on acoustics, speech, and signal processing*, Vol. 28, No. 4, pp. 357–366, 1980.

[Doetsch & Zeyer$^+$ 17] P. Doetsch, A. Zeyer, P. Voigtlaender, I. Kulikov, R. Schlüter, H. Ney. Returnn: the rwth extensible training framework for universal recurrent neural networks. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 5345–5349, March 2017.

[Federico & Cettolo$^+$ 12] M. Federico, M. Cettolo, L. Bentivogli, P. Michael, S. Sebastian. Overview of the iwslt 2012 evaluation campaign. In *IWSLT-International Workshop on Spoken Language Translation*, pp. 12–33, 2012.

[Hochreiter & Schmidhuber 97] S. Hochreiter, J. Schmidhuber. Long short-term memory. *Neural computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

[Huang & Seide 00] H. C.-H. Huang, F. Seide. Pitch tracking and tone features for mandarin speech recognition. In *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)*, Vol. 3, pp. 1523–1526. IEEE, 2000.

[Ittichaichareon & Suksri$^+$ 12] C. Ittichaichareon, S. Suksri, T. Yingthawornsuk. Speech recognition using mfcc. In *International Conference on Computer Graphics, Simulation and Modeling (ICGSM'2012)*, pp. 28–29, 2012.

[iws 19] *The IWSLT 2019 Evaluation Campaign.* Zenodo, Nov. 2019.

[Levy & Silber-Varod$^+$ 12] T. Levy, V. Silber-Varod, A. Moyal. The effect of pitch, intensity and pause duration in punctuation detection. In *2012 IEEE 27th Convention of Electrical and Electronics Engineers in Israel*, pp. 1–4. IEEE, 2012.

[Makhoul & Kubala$^+$ 99] J. Makhoul, F. Kubala, R. Schwartz, R. Weischedel et al. Performance measures for information extraction. In *Proceedings of DARPA broadcast news workshop*, pp. 249–252. Herndon, VA, 1999.

[Matusov & Mauser$^+$ 06] E. Matusov, A. Mauser, H. Ney. Automatic sentence segmentation and punctuation prediction for spoken language translation. In *International Workshop on Spoken Language Translation (IWSLT) 2006*, 2006.

[Mikolov & Chen$^+$ 13] T. Mikolov, K. Chen, G. S. Corrado, J. Dean. Efficient estimation of word representations in vector space. *CoRR*, Vol. abs/1301.3781, 2013.

[Panayotov & Chen$^+$ 15] V. Panayotov, G. Chen, D. Povey, S. Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210. IEEE, 2015.

[Peitz & Freitag$^+$ 11] S. Peitz, M. Freitag, A. Mauser, H. Ney. Modeling punctuation prediction as machine translation. In *International Workshop on Spoken Language Translation (IWSLT) 2011*, 2011.

[Powers 11] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation, 2011.

[Ray & Rajeswar$^+$ 15] A. Ray, S. Rajeswar, S. Chaudhury. Text recognition using deep blstm networks. In *2015 eighth international conference on advances in pattern recognition (ICAPR)*, pp. 1–6. IEEE, 2015.

[Sak & Senior$^+$ 14] H. Sak, A. Senior, F. Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, 2014.

[Sennrich & Haddow$^+$ 16] R. Sennrich, B. Haddow, A. Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, Aug. 2016. Association for Computational Linguistics.

[Szaszák 19] G. Szaszák. An audio-based sequential punctuation model for asr and its effect on human readability. *Acta Polytechnica Hungarica*, Vol. 16, No. 2, 2019.

[Tilk & Alumäe 15] O. Tilk, T. Alumäe. Lstm for punctuation restoration in speech transcripts. In *Sixteenth annual conference of the international speech communication association*, 2015.

[Ueffing & Bisani$^+$ 13] N. Ueffing, M. Bisani, P. Vozila. Improved models for automatic punctuation prediction for spoken and written text. In *Interspeech*, pp. 3097–3101, 2013.

[Vaswani & Shazeer$^+$ 17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin. Attention is all you need. *CoRR*, Vol. abs/1706.03762, 2017.

[Wang & Chen$^+$ 18] F. Wang, W. Chen, Z. Yang, B. Xu. Self-attention based network for punctuation restoration. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pp. 2803–2808. IEEE, 2018.

[Xiao-Dan Mei & Jengshyang Pan$^+$ 01] Xiao-Dan Mei, Jengshyang Pan, Sheng-He Sun. Efficient algorithms for speech pitch estimation. In *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing. ISIMP 2001 (IEEE Cat. No.01EX489)*, pp. 421–424, 2001.

[Yi & Tao 19] J. Yi, J. Tao. Self-attention based model for punctuation prediction using word and speech embeddings. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7270–7274. IEEE, 2019.