



ZURICH UNIVERSITY OF APPLIED SCIENCES

PROJEKTARBEIT

---

# Texte automatisch korrigieren

---

*Authors:*

Francis DU  
Alexandre MANAI

*Supervisors:*

Prof. Dr. Mark CIELIEBAK  
Dr. Don TUGGENER

*A Projektarbeit submitted in fulfillment of the requirements  
for the degree of Bachelor of Computer Science*

*in the*

Zurich University of Applied Sciences  
Center for Artificial Intelligence

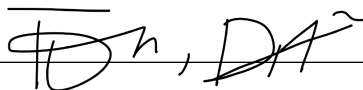
December 24, 2021

## Declaration of Authorship

Francis DU & Alexandre MANAI, declare that this thesis titled, "Texte automatisch korrigieren" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in Bachelor degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: \_\_\_\_\_



Date: 24.12.2021  
\_\_\_\_\_

ZURICH UNIVERSITY OF APPLIED SCIENCES

# *Abstract*

Computer Science

Center for Artificial Intelligence

Bachelor of Computer Science

**Texte automatisch korrigieren**

by Francis DU & Alexandre MANAI

Speech-to-Text engines have made great advances in recent years, though problems like incomplete sentences, disfluencies and grammatically incorrect sentences still persist. Especially when performing sentence boundary detection and punctuation tasks poor performance is observed which contributes to a dreadful reading experience. This thesis deals with improving readability of such automatically transcribed texts. We focus on the punctuation prediction aspect and develop an evaluation pipeline that uses punctuation predictors from 2 different implementations. Several experiments are conducted with mainly three corpora of three different readability levels to establish the boundaries of model capabilities. The result of this work suggests that appropriate punctuation placements improve the quality of synthesized texts. Finally, we propose a simple agreement algorithm of how to combine two punctuation models for optimal punctuation prediction and propose additional approaches to improve our solution in future works.

ZÜRCHER HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN

# *Zusammenfassung*

Computer Science

Center for Artificial Intelligence

Bachelor of Computer Science

**Texte automatisch korrigieren**

von Francis DU & Alexandre MANAI

Speech-to-Text Engines haben in vergangenen Jahren grosse Fortschritte gemacht, obwohl Probleme wie unvollständige Sätze, Sprachunstetigkeit und grammatisch inkorrekte Sätze immer noch bestehen. Insbesondere bei der Satzabgrenzungserkennung und Zeichensetzung kann man schlechte Leistungen erkennen, was zu einem miserablen Leseerlebnis führen kann. Diese Arbeit beschäftigt sich mit der Verbesserung der Leserlichkeit von solchen automatisch transkribierten Texten. Wir fokussieren uns dabei auf die Vorhersage von Satzzeichen und entwickeln eine Evaluierungspipeline welche Satzzeichenprädiktoren aus zwei verschiedenen Implementationen benutzt. Diverse Experimente werden durchgeführt mit hauptsächlich drei Korpora mit verschiedenen Leserlichkeitsstufen um die Grenzen der Modellfähigkeiten zu erkennen. Schliesslich schlagen wir einen einfachen Algorithmus vor, der in Kombination von zwei Zeichensetzungsmodelle optimal Satzzeichen vorherzusagen und weisen auf weitere Herangehensweisen für zukünftige Arbeiten hin um Lösung in zukünftigen Arbeiten zu verbessern.

## *Acknowledgements*

We would like to thank Prof. Dr. Mark Cieliebak and Dr. Don Tugener for their insightful help and motivational push. We learned enormously about the thoroughness of scientific procedures thanks to their guidance.

# Contents

<b>Declaration of Authorship</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Zusammenfassung</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Baseline . . . . .	1
1.2 Objectives . . . . .	2
<b>2 Theoretical foundations</b>	<b>3</b>
2.1 RNN - Recurrent Neural Networks . . . . .	3
2.1.1 Bidirectional RNNs . . . . .	5
2.1.2 LSTM - Long short-term memory . . . . .	5
Bidirectional LSTMs . . . . .	6
2.2 Attention mechanism . . . . .	7
2.2.1 Encoder . . . . .	7
2.2.2 Decoder . . . . .	8
2.3 Transformer architectures . . . . .	9
2.3.1 Input Embedding . . . . .	9
2.3.2 Positional Encoding . . . . .	10
2.3.3 Multi-head Attention . . . . .	10
2.3.4 Masked Multi-head Attention . . . . .	10
2.3.5 Feed Forward . . . . .	10
2.3.6 Linear . . . . .	10
2.3.7 Softmax . . . . .	10
2.4 Activation functions . . . . .	11
2.4.1 Sigmoid . . . . .	11
2.4.2 Tanh . . . . .	11
2.4.3 ReLU . . . . .	11
2.5 Evaluation metrics . . . . .	12
2.5.1 Important Information . . . . .	12
2.5.2 Classification Report . . . . .	12
Classification types . . . . .	12
Accuracy . . . . .	12
Precision . . . . .	12
Recall . . . . .	12
F1-score . . . . .	12
Support . . . . .	12
2.5.3 Confusion Matrix . . . . .	13

<b>3</b>	<b>Corpora</b>	<b>14</b>
3.1	ICSI Meeting Corpus . . . . .	14
3.1.1	Conclusion . . . . .	15
3.2	BBCNews Dataset . . . . .	15
3.2.1	Conclusion . . . . .	16
3.3	Interscriber transcripts . . . . .	16
<b>4</b>	<b>Punctuation models</b>	<b>17</b>
4.1	Punctuation restoration . . . . .	17
4.2	Punctuator2 . . . . .	17
<b>5</b>	<b>Interscriber tool</b>	<b>18</b>
5.1	Features . . . . .	18
5.1.1	Audio Transcription . . . . .	18
5.1.2	Editor . . . . .	18
<b>6</b>	<b>Procedure and methodology</b>	<b>20</b>
6.1	Initial analysis . . . . .	20
6.1.1	Procedure . . . . .	20
6.1.2	Data . . . . .	20
6.2	Main analysis . . . . .	20
6.2.1	Overview Pipeline . . . . .	20
	Preprocessing . . . . .	23
	Punctuation prediction . . . . .	24
	Postprocessing . . . . .	24
	Evaluation . . . . .	24
6.2.2	Experiments . . . . .	24
	Iteration 1 . . . . .	25
	Iteration 2 . . . . .	25
	Iteration 3 . . . . .	25
	Human Evaluation . . . . .	25
<b>7</b>	<b>Results</b>	<b>26</b>
7.1	Initial analysis . . . . .	26
7.1.1	Syntax result . . . . .	26
7.1.2	Error distribution . . . . .	27
7.1.3	Interpretation . . . . .	27
7.1.4	Conclusion . . . . .	28
7.2	Main analysis . . . . .	28
7.2.1	Iteration 1 . . . . .	28
	Punctuation restoration . . . . .	28
	Punctuator2 . . . . .	30
7.2.2	Iteration 2 . . . . .	31
	Punctuation restoration . . . . .	31
	Punctuator2 . . . . .	33
7.3	Iteration 3 . . . . .	35
	Punctuation restoration . . . . .	35
	Punctuator2 . . . . .	37
7.3.1	Human Evaluation . . . . .	39
<b>8</b>	<b>Encountered Issues</b>	<b>40</b>
8.1	Disfluency . . . . .	40

8.1.1	Motivation . . . . .	40
8.1.2	Work . . . . .	40
8.1.3	Issue . . . . .	40
8.2	Curl request for Punctuator2 model . . . . .	40
8.2.1	Motivation . . . . .	40
8.2.2	Work . . . . .	41
8.2.3	Issue . . . . .	41
8.3	Sentence Compression . . . . .	41
8.3.1	Motivation . . . . .	41
8.3.2	Issue . . . . .	41
8.4	General Pipeline . . . . .	42
8.4.1	Motivation . . . . .	42
8.4.2	Work . . . . .	42
8.4.3	Issue . . . . .	42
8.5	Learning the Theory . . . . .	43
8.5.1	Motivation . . . . .	43
8.5.2	Work . . . . .	43
8.5.3	Issue . . . . .	43
<b>9</b>	<b>Conclusion</b>	<b>44</b>
<b>10</b>	<b>Future Works</b>	<b>45</b>
10.1	Agreement . . . . .	45



## Chapter 1

# Introduction

In recent years research in the field of speech-to-text has made great advances. The ever-growing amount of generated data and bigger computing power have contributed massively towards this development. Engines like IBM's Watson Speech to Text<sup>1</sup>, Google Cloud<sup>2</sup> and Microsoft Azure<sup>3</sup> have been developed for commercial use and perform quite well in the English language. However, spoken and written dialogues differ greatly in structure and flow as there could be many mistakes such as correcting a sentence mid speech or stuttering and other kinds of disfluencies in the former, whereas the latter benefits from corrections and neat grammar. Therefore, generated transcripts of spoken English suffer at readability because of mistakes that seem basic for most people. Small mistakes like wrongly placed punctuation marks could disturb the reading flow of the user.

This work focuses on the aspect of readability of such automatically generated transcripts, especially of meeting transcripts where multiple people are involved. We investigate how we could alleviate the reader from a painstaking experience when perusing through a transcribed text and in a second step, using state of the art punctuation models, we conduct experiments with the goal to improve readability.

### 1.1 Baseline

As mentioned previously, spontaneous spoken speech is accompanied by irregularities such as repetitions, incomplete sentences, disfluencies and grammatically incorrect sentences. To smoothen such artifacts several aspects need to be looked at:

- Extend incomplete sentences
- Remove self-corrections and repetitions, filler words and hesitations
- Correct grammatically incorrect sentences
- Insert punctuation

Several possible approaches have already been proposed. In the domain of grammar correction, the paper of Lee and Seneff [1] investigates the problem by creating a paraphrase-generation-based approach. Additionally, this method can be extended with other paraphrasing-based methods which could remove repetitions, filler words, stutters and, furthermore, complete sentences.

---

<sup>1</sup><https://www.ibm.com/cloud/watson-speech-to-text>

<sup>2</sup><https://cloud.google.com/speech-to-text>

<sup>3</sup><https://azure.microsoft.com/de-de/>

With the idea of extension and adaptability in mind, Hori et al. [2] used a paraphrasing weighted Finite-State Transducer which deemed quite successful.

Another necessary post-processing step is punctuation prediction. Tilk and Alumäe [3] put forward a trained language model combined with detected audio splitting points capable of predicting punctuation.

Promising ideas have also been seen in the domain of language simplification. Especially an approach based on Siddhartan [4] which first identifies the structure of the sentence, then simplifies the sentence according to the introduced rules and, finally, corrects some expressions in order to make them resemble human writing.

The problem of punctuation prediction is not new, though still young. It is almost solved for formal written language but faces challenges in synthetic or automatically generated texts (Tuggener and Aghaebrahimian, 2021)[10]. Several approaches have been researched over the last couple of decades. Earlier works make use of a set of rules or regular expressions much like the system proposed by Grefenstette and Tapanainen [11]. Most recent advances however, take the approach of deep learning by Kaur and Singh [12] or naïve bayes based models by Loper and Pardo[13]. Tuggener and Aghaebrahimian [10] also propose solutions for insertion of punctuation marks into text produced by Natural Language Generation (NLG) systems. In particular, they predict correct positions for periods and other punctuation marks in general. One limitation though is that models are trained on a fixed set of corpora in which they perform well. However, when faced with out-of-domain data performance decreases significantly [10].

## 1.2 Objectives

The objective of this work is to propose a solution as how to improve readability of automatically generated text by placing punctuation marks at appropriate positions. The challenge lies in dealing with transcribed spoken language and dealing with various topic domains. Punctuation models are in the usual case trained on one specific domain where they produce stellar results, but perform poorly when dealing with another. Therefore, several experiments are conducted to analyse the performance of punctuation models on corpora containing different levels of readability. We will do so by designing a pipeline to automatically generate reports and evaluations.

In chapter 2 we cover the theoretical foundations. Chapter 3 presents the corpora used for this work and gives a short description of them. In chapter 4 and 5 the punctuation models and tools used for our pipeline are introduced and briefly explained. Chapter 6 then explains our procedure and methodology by giving an insight as to how we planned our experiments. Their results and interpretation are shown in chapter 7. Chapter 8 shows what problems we encountered during this work. To bring everything together, in chapter 9 we form our conclusion and finally, chapter 10 serves as a means to have a discussion and present possible ideas for future works.

## Chapter 2

# Theoretical foundations

In this chapter we'll cover all the necessary theoretical background needed to be able to understand and evaluate the models/procedure used.

Section 2.1 and 2.2 go in detail in the concepts of Recurrent Neural Networks, LSTMs and Attention mechanisms. In section 2.3, we talk about Transformer architectures. These sections are important to understand why we made the decisions explained in Chapter 5.

Finally, section 2.4 is dedicated to understanding the metrics used to evaluate the models.

### 2.1 RNN - Recurrent Neural Networks

Recurrent Neural Networks (RNN) is a type of Artificial Neural Network (ANN) which is altered to be able to analyse temporal/sequence data well. More specifically, it allows the previous outputs to be used as inputs. This makes it suitable for Natural Language Processing (NLP) tasks such as speech recognition.

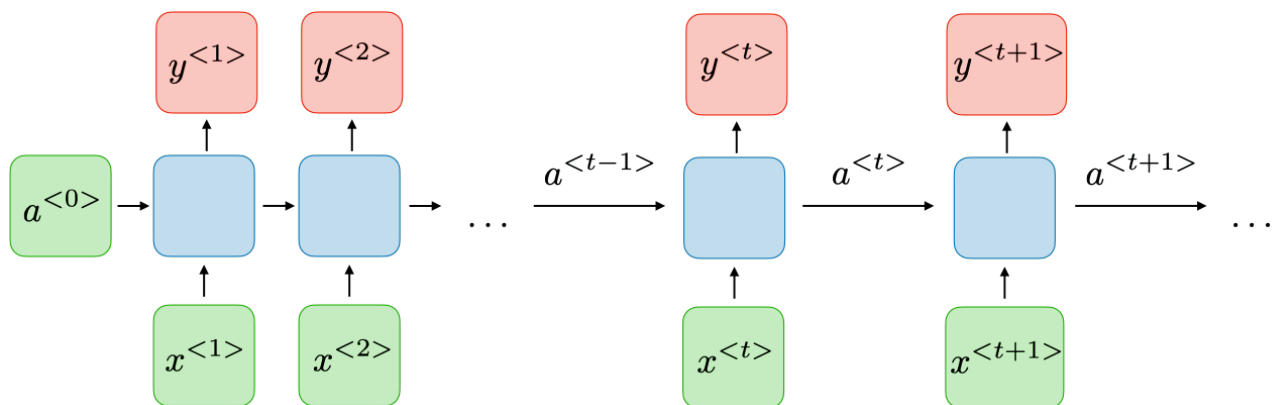
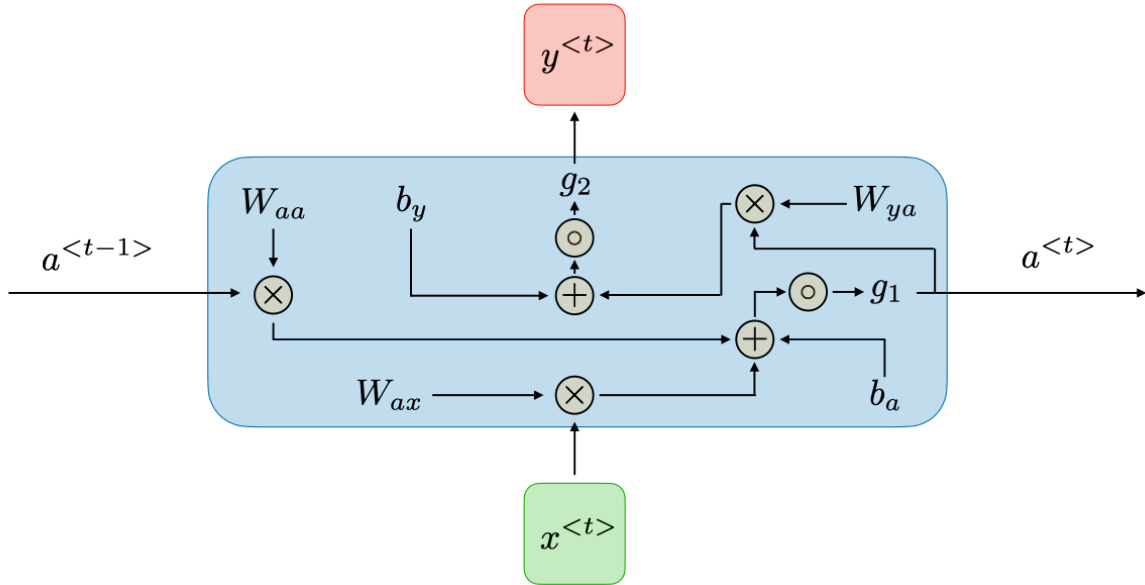


FIGURE 2.1: Architecture of traditional RNN<sup>0</sup>

The sequential nature of those neural networks is visible in Figure 2.1 where each activation  $a^{<t>}$  at timestep  $t$  is used as input for the block in layer  $t+1$ .

The block (depicted in blue in Figure 2.1) has following logic:

FIGURE 2.2: Logic of RNN block<sup>0</sup>

We can see in Figure 2.2, the dependence of  $a^{<t-1>}$  for block at  $t$ . The output  $a^{<t>}$  is the direct summation of  $a^{<t-1>}W_{aa}$ ,  $x^{<t>}W_{ax}$  and bias  $b_a$ .

Additionally, this aspect is also visible directly in the definition of  $a^{<t>}$  and  $y^{<t>}$ :

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

$$y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

with  $g_1$  and  $g_2$  being activation functions (see section 2.3)

The dependence of block  $t-1$  for block  $t$  creates a problem. Each input needs to be processed sequentially negating parallel processing which is favorable for training and predicting for Neural Networks. Additionally, the forward dependency creates a strict temporal order which gives current inputs context of previous inputs but not of future ones. [5]

Forward and backwards context might be useful in NLP, therefore a solution to the context problem has been developed called **Bidirectional RNN** (BRNN).

### 2.1.1 Bidirectional RNNs

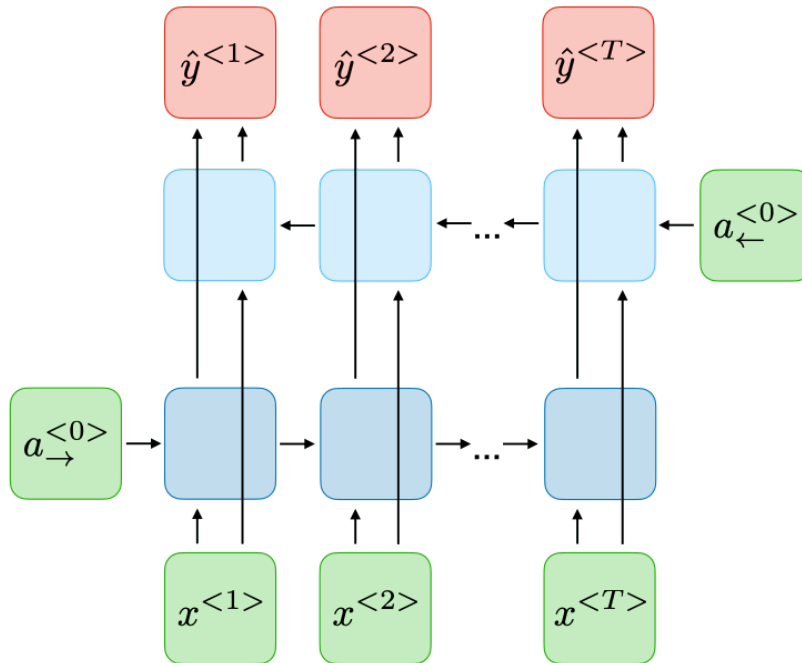


FIGURE 2.3: Bidirectional RNN architecture<sup>0</sup>

In Figure 2.3, we can observe a forward pass through time, beginning from the start of the sequence and another RNN starting from the end running backwards in time. This idea gives the model a richer representation enabling better predictions. [5]

### 2.1.2 LSTM - Long short-term memory

LSTMs were first introduced by Hochreiter and Schmidhuber [6]. Their idea was to respond to the "vanishing gradient problem" where long-term gradients might "vanish" meaning tending towards zero or "explode" meaning tending towards infinity. They found a way to remember long-term information for long periods of time.

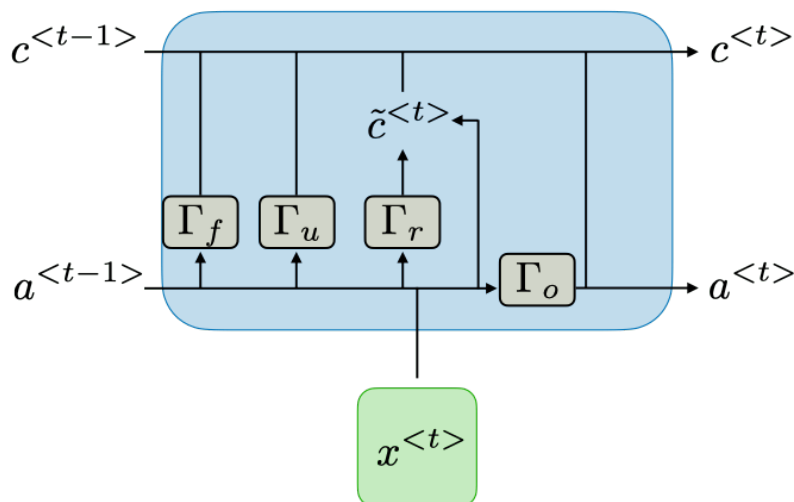


FIGURE 2.4: LSTM internal architecture<sup>0</sup>

Their implementation included different types of gates to enable memory management[9]:

- $\Gamma_f$  (Forget Gate): decides if cell  $c^{<t-1>}$  should be erased or not
- $\Gamma_u$  (Update Gate): responsible for the amount of information  $\tilde{c}^{<t>}$  that flows through
- $\Gamma_r$  (Relevance Gate): responsible if previous information  $a^{<t-1>}$  gets forgotten or not
- $\Gamma_o$  (Output Gate): responsible for how much of  $c^{<t>}$  should be revealed

Definition:

$$\Gamma = \sigma(Wx^{<t>} + Ua^{<t-1>} + b) \quad (2.1)$$

where  $W, U, b$  depend on the gate used and  $\sigma$  is the sigmoid activation function (see section 2.3)[9]

### Bidirectional LSTMs

Bidirectional LSTMs have been introduced by Graves, Fernandez and Schmidhuber [7] in the same mindset of BRNN to introduce more bilateral context.

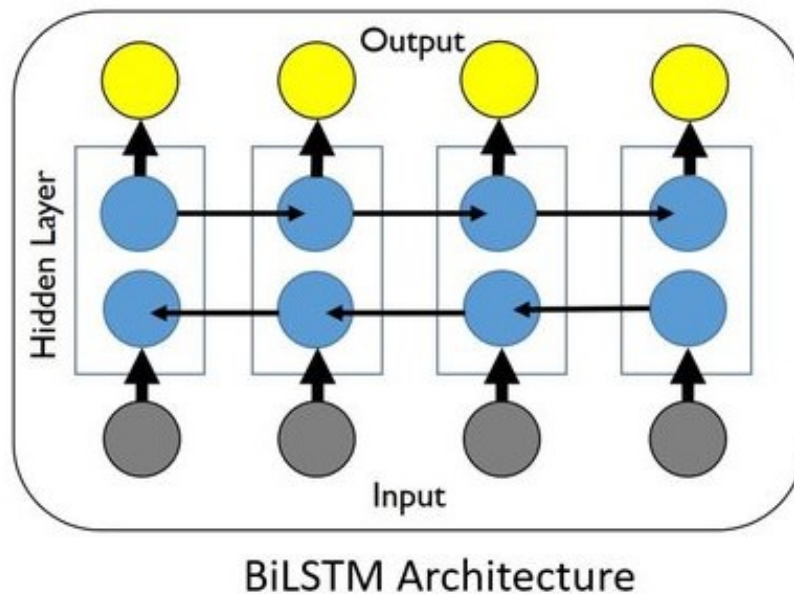


FIGURE 2.5: BiLSTM Architecture<sup>1</sup>

The cell present in Figure 2.4 is depicted as the blue circle in Figure 2.5.

## 2.2 Attention mechanism

Attention mechanisms found in neural networks are somewhat similar to the ones found in humans. They focus in high resolution on one part of the input and in low resolution on the rest.[14]

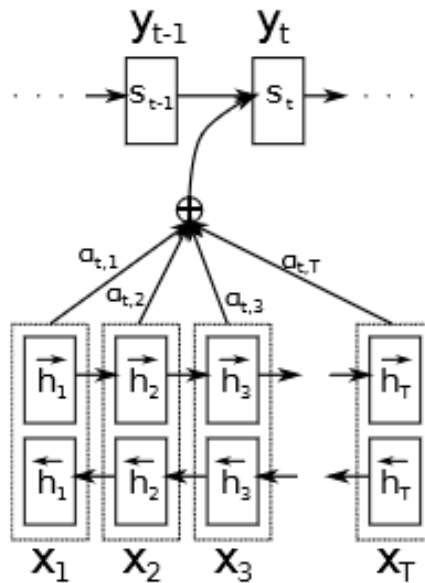


FIGURE 2.6: Attention mechanism

Following architecture (see Figure 2.6) is based on an Encoder-Decoder architecture. The encoder part is based on a Bidirectional Recurrent Network (BiRNN)[15] and the decode part emulates searching through a sentence during the decoding for a translation.

### 2.2.1 Encoder

Generally speaking, BiRNN have forward states  $\vec{h}_i$  and backwards states  $\overleftarrow{h}_i$  which is calculated for  $T$  words.

They concatenated in one  $h_i$  vector:

$$h_i = \begin{bmatrix} \vec{h}_i \\ \overleftarrow{h}_i \end{bmatrix} \quad (2.2)$$

<sup>0</sup> Graphs issued from here: <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>.

<sup>1</sup> Graphs issued from here: [https://www.researchgate.net/figure/LSTM-and-BiLSTM-Architectures\\_fig2\\_324769532](https://www.researchgate.net/figure/LSTM-and-BiLSTM-Architectures_fig2_324769532).

### 2.2.2 Decoder

In this part we try to decode the  $h_i$ 's with help of an attention mechanism. We're trying to find  $y_t = s_t$  defined in the following way:

$$s_i = f(s_{i-1}, y_{i-1}, c_i) \quad (2.3)$$

Now let's look at each component more in depth to understand the mechanism.

$$c_i = \sum_{j=1}^T \alpha_{ij} h_j \quad (2.4)$$

Each weight  $\alpha_{ij}$  is computed by:

$$c_i = \frac{\exp(e_{ij})}{\sum_{k=1}^T \exp(e_{ik})} \quad (2.5)$$

where

$$e_{ij} = a(s_{i-1}, h_j) \quad (2.6)$$

$a$  is an alignment model which scores how well the inputs around  $h_j$  and output at  $i$  are similar.

$\alpha_{ij}$  is a probability that the word  $y_i$  is aligned/translated from the word  $x_j$ .

$e_{ij}$  is also called energy and denotes the importance of  $h_j$  to the previous layer  $s_{i-1}$



## 2.3 Transformer architectures

Transformers are deep-learning models which adopt a mechanism of self-attention. Compared to RNN's, previously seen, transformers don't process data in sequential order rather thanks to the attention mechanism which provides context for all positions in the input data. For NLP, for example, it wouldn't have to start analysing the start of the sentence, but it could start at the end. Additionally, it provides more parallelism than RNN's do.[16]

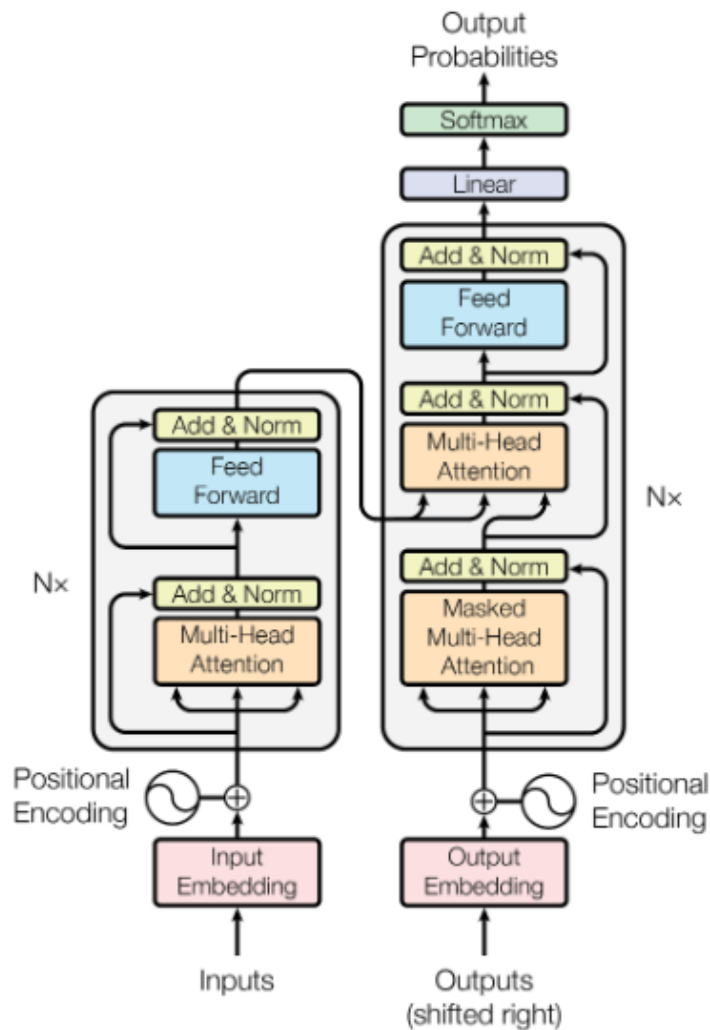


FIGURE 2.7: Transformer Architecture

Transformers are essentially based on a Encoder/Decoder architecture with the encoding part on the left side and the decoding part on the right side (see Figure 2.7). Let's look at each component to understand its workings:

### 2.3.1 Input Embedding

This unit of the model is responsible for passing a word into an embedding space. Therefore creating a vector representation of the word to be suitable as input for the model.

We have unsupervised learning methods that are able to achieve that, for example, GloVe<sup>1</sup>

### 2.3.2 Positional Encoding

Solely the Input Embedding creates a vector representation of the word which is stripped of his positional information in the sentence. That's why we have an additional "Positional Encoding" unit which encodes this positional information(context) into the vector.

### 2.3.3 Multi-head Attention

This is the block that provides the self-attention. For an NLP task such as translation, for example, each word is compared to every other word in the sentence. An attention vector is computed which holds a value for each word in the sentence. That value represents relevance of one word to all the other words in the sentence. It, therefore, catches contextual information about the sentence.

### 2.3.4 Masked Multi-head Attention

This unit is essentially similar to "Multi-head Attention", but the difference lies in the comparisons made for each word relative to the other words of the sentence. In the "Multi-head Attention" unit every word is compared, but in this one we only compare the word to words that appear before it and not after.

### 2.3.5 Feed Forward

This unit takes all attention vectors computed in the layer before and feeds it to a feed forward neural network which has the task to bring to a dimension which is usable for the next step.

### 2.3.6 Linear

This unit, in the case of translation, brings the input vector to the dimension of the whole vocabulary of the target language.

### 2.3.7 Softmax

Finally, the "Softmax" unit brings the vector to a probability distribution making it interpretable to humans. In the case of translation, this model would give out the word with highest probability to appear next and this until we have the fully translated input sentence.

---

<sup>1</sup>More Information: <https://nlp.stanford.edu/projects/glove/>

## 2.4 Activation functions

### 2.4.1 Sigmoid

Definition:

$$g(z) = \frac{1}{1 + e^{-z}} \quad (2.7)$$

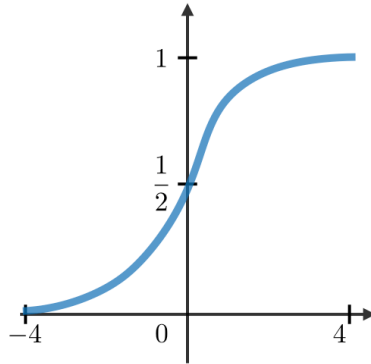


FIGURE 2.8: Sigmoid activation function

### 2.4.2 Tanh

Definition:

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \quad (2.8)$$

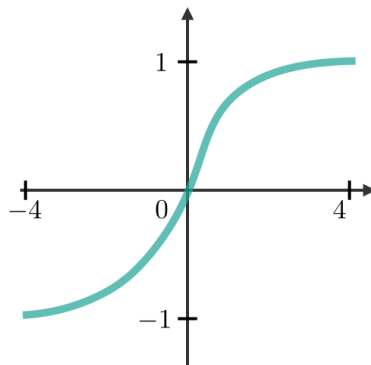


FIGURE 2.9: Tanh activation function

### 2.4.3 ReLU

Definition:

$$g(z) = \max(0, z) \quad (2.9)$$

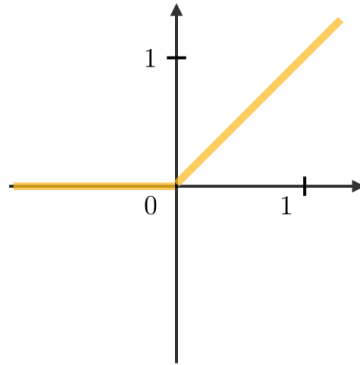


FIGURE 2.10: ReLU activation function

## 2.5 Evaluation metrics

In this section, we'll look at different metrics and the mediums we used to present them with.

### 2.5.1 Important Information

In our classification tasks, the empty space is denoted as "0".

### 2.5.2 Classification Report

#### Classification types

True Positive =  $TP$

True Negative =  $TN$

False Positive =  $FP$

False Negative =  $FN$

#### Accuracy

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

#### Precision

$$\text{Precision} = \frac{TP}{TP+FP}$$

#### Recall

$$\text{Recall} = \frac{TP}{TP+FN}$$

#### F1-score

$$F1 = \frac{2*Precision*Recall}{Precision+Recall} = \frac{2*TP}{2*TP+FP+FN}$$

#### Support

Represents the amount of occurrences of a certain classification prediction

### 2.5.3 Confusion Matrix

		Predicted label	
		p	n
True label	p'	True Positive	False Negative
	n'	False Positive	True Negative

To have a better understanding of the nature of the mistakes compared to the classification report, we'd look into the Confusion Matrix.

In the confusion matrix, we can see the relation between the labels predicted by the model and the actual ground-truth labels. We can observe which labels have been wrongly predicted and by which making the models behavior more understandable.

Essentially, the confusion matrix shows the ways your classification model is confused when it makes predictions.<sup>2</sup>

---

<sup>2</sup>Quote from: <https://machinelearningmastery.com/confusion-matrix-machine-learning/>

## Chapter 3

# Corpora

In this chapter we will present all the datasets used for conducting our experiments. Section 3.1 will show the structure of the ICSI dataset. In section 3.2 we elaborate on the BBCNews dataset and finally, in section 3.3 we present the interscriber output.

### 3.1 ICSI Meeting Corpus

The ICSI Meeting Corpus consists of over 70 hours of meeting recordings. It was built by transcribing meetings that occurred at the International Computer Science Institute (ICSI) in Berkeley, California. It contains audio, word-level transcripts of meetings and various metadata on participants, meetings and hardware[19].

Transcripts / Metadata: All transcripts of every meeting are saved as xml files. They include the following metadata:

Per word (tag `<w>`):

- Id
- Start time
- End time
- Type of word

Per vocalsound (tag `<vocalsound>`):

- Id
- Start time
- End time
- Description

Per disfluency maker (tag `<disfmarker>`):

- Id
- Start time
- End time

An example can be seen in Listing 1

```

1 <w nite:id="Bdb001.w.1,266" starttime="339.926" endtime="340.256" c="W
  ">could</w>
2 <disfmarker nite:id="Bdb001.disfmarker.51" starttime="" endtime=""/>
3 <w nite:id="Bdb001.w.1,267" starttime="340.256" endtime="340.426" c
  ="W">all</w>
4 <w nite:id="Bdb001.w.1,268" starttime="340.426" endtime="" c="W">you
  </w>
5 <w nite:id="Bdb001.w.1,269" starttime="" endtime="340.586" c="W">'d
  </w>
6 <w nite:id="Bdb001.w.1,270" starttime="340.586" endtime="340.756" c
  ="W">have</w>
7 <w nite:id="Bdb001.w.1,271" starttime="340.756" endtime="340.876" c
  ="W">to</w>
8 <w nite:id="Bdb001.w.1,272" starttime="340.876" endtime="341.276" c
  ="W">change</w>
9 <w nite:id="Bdb001.w.1,273" starttime="341.276" endtime="341.556" c
  ="W">is</w>
10 <w nite:id="Bdb001.w.1,274" starttime="341.556" endtime="341.936" c
  ="W">the</w>
11 <w nite:id="Bdb001.w.1,275" starttime="341.936" endtime="341.936" c
  ="CM">,</w>
12 <vocalsound nite:id="Bdb001.vocalsound.42" starttime="" endtime=""
  description="breath"/>
13 <wnite:id="Bdb001.w.1,278" starttime="342.469" endtime="343.009" c
  ="W">um</w>
14 <w nite:id="Bdb001.w.1,279" starttime="343.009" endtime="343.009" c
  ="CM">,</w>

```

LISTING 3.1: Sample ICSI

### 3.1.1 Conclusion

The extracted texts stem from speech and include interruptions, stutters, sentence repetition and other types of disfluency. These fit perfectly to our use case. We expect our models to encounter problems with inserting the punctuations at the correct place.

## 3.2 BBCNews Dataset

This dataset consists of 2225 documents from the news site BBC containing articles in five topical areas from 2004-2005[20]. These are texts written by many different journalists in different styles. For every article there is one text file with the headline at the first line followed by the article. The article itself is divided randomly into multiple lines with empty lines in between. See Listing 2 as example.

```

1 Ink helps drive democracy in Asia
2
3 The actual technology behind the ink is not that complicated. The ink
  is sprayed on a person's left thumb.
4
5 Widely circulated articles compared the use of ink to the rural
  practice of marking sheep
6
7 David Mikosz works for the IFES, a non-profit organisation that
  supports the building of democracies.

```

LISTING 3.2: Sample BBCNews

### 3.2.1 Conclusion

These are texts written in English with correct punctuations and sentence structure. Though they are not transcripts of spoken language they serve as a means to evaluate our punctuation models under optimal conditions. Therefore, we expect our models to be able to replicate the punctuations with a good performance.

### 3.3 Interscriber transcripts

The Interscriber transcripts used for our experiments consist of transcribed audio files that were downloaded from the internet. They can range from interviews to monologues and we assess the quality of sentence to be in the middle of the ICSI corpus and the BBC dataset. We expect our models to perform moderately over these transcripts.



## Chapter 4

# Punctuation models

In this chapter we present the different punctuation models we use in this work and explain why we chose to use them. We narrowed down our choices to two models.

### 4.1 Punctuation restoration

The implementation of this model is based on the paper by Tanvirul et. al[17] and is able to restore punctuation for the English language. They fine tuned a transformer<sup>1</sup> based architecture for the task. While they provide means to train the model on custom data, pretrained models are also available. In this example, we used the RoBERTa-large model which is pretrained on a large corpus of English data. The transformer encoder is followed by an LSTM<sup>2</sup> and a final linear layer that predicts punctuation. For our task, this model presents itself as a good choice for conducting experiments as they themselves obtain state-of-the-art results and neatly provide code and a pretrained model. Their implementation is available in this repository<sup>3</sup> that also offers examples on how to use it.

### 4.2 Punctuator2

This is a punctuator implementation that is based on the paper by Ottokar Tilk and Tanel Alumäe[18]. Their model uses a bidirectional recurrent neural network<sup>4</sup> architecture with an attention mechanism in order to punctuate text. An online repository that provides some documentation on usage can be found here<sup>5</sup>. They offer the possibility to train the models on own data, but also provide pretrained models. These have been trained on English TED talks<sup>6</sup> data which consists of 2.1M words and the English EuroparlV7<sup>7</sup> corpus which consists of 40M words. For this task, this punctuator seemed like a good choice due to its usability right out of the box and available pretrained models.

---

<sup>1</sup>See chapter 2.3

<sup>2</sup>See chapter 2.1.2

<sup>3</sup><https://github.com/xashru/punctuation-restoration>

<sup>4</sup>see Chapter 2.1.3

<sup>5</sup><https://github.com/ottokart/punctuator2>

<sup>6</sup><https://www.idiap.ch/en/dataset/ted>

<sup>7</sup><https://www.statmt.org/europarl/>

## Chapter 5

# Interscriber tool

In this chapter we introduce Interscriber<sup>1</sup>, an auto-transcription tool by Spinning-Bytes AG, which is a joint spin off of the Swiss Federal Institute of Technology and the Zurich University of Applied Sciences.

## 5.1 Features

### 5.1.1 Audio Transcription

The website features automatic transcription of more than 15 different audio formats. Via a new project or an import project button the tool is able upload an audio file and transcribe it within minutes. Though it is able to combine several speech processing engines by Google, IBM, etc. they also offer a confidential mode where data never leaves servers from Switzerland. However, the engine for confidential mode was developed in-house and doesn't perform as well as other engines from the big companies.

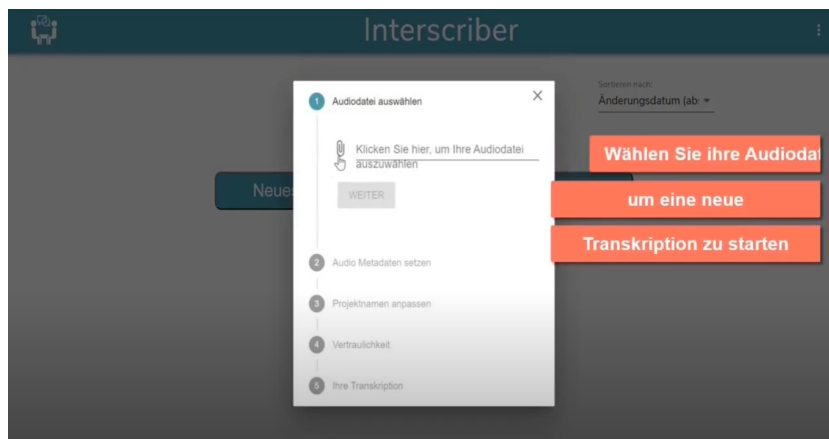


FIGURE 5.1: Interscriber Upload Tool<sup>0</sup>

### 5.1.2 Editor

After transcribing audio the site offers a powerful editor. On the upper left the audio track can be seen and played at a desired timestamp. The current word is then

<sup>1</sup>See <https://interscriber.com/de/>

<sup>1</sup><https://spinningbytes.com/>

highlighted in the text. It additionally offers following features:

On the mid left:

- Assign phrases to other speakers
- Change speaker names

In the text:

- Auto-merge sentences
- Correct errors in text
- Export the text for editing in word

The buttons on the upper right:

- Export the text as a word document
- Export as an Excel sheet

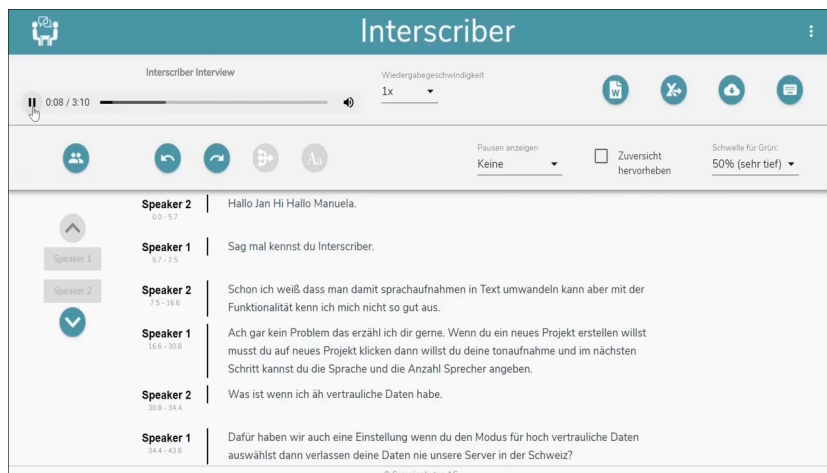


FIGURE 5.2: Interscriber Edit Tool<sup>0</sup>

<sup>0</sup> Graphs issued from here: [https://www.youtube.com/watch?v=sN\\_N9js8vp8&ab\\_channel=SpinningBytes](https://www.youtube.com/watch?v=sN_N9js8vp8&ab_channel=SpinningBytes).

## Chapter 6

# Procedure and methodology

### 6.1 Initial analysis

This section will concentrate on the initial analysis we did to be able to filter feasible objectives out of the broad given baseline.

Essentially, we had four major aspects to consider:

- Extend incomplete sentences
- Remove self-corrections and repetitions, filler words and hesitations
- Correct grammatically incorrect sentences
- Insert punctuation

The result of this analysis gave us an aspect to focus on.

#### 6.1.1 Procedure

The procedure used was relatively straight forward.

We let speech files run through Interscriber<sup>1</sup>, collect the output, proceed to identify the occurrence of each artifact and, finally, interpret the results and decide which artifact to focus on.

#### 6.1.2 Data

We used two 5 minute recordings of the meetings we had for our Projektarbeit including four speakers and, to diversify, we took a 10 minute recording of an interview with clear voices and separations between two speakers.

### 6.2 Main analysis

In this section we present the procedures and methodology of our main evaluation pipeline. We elucidate on the elements of our pipeline and all the artifacts that are generated and used.

#### 6.2.1 Overview Pipeline

The whole evaluation process is depicted in Figure 6.1. It can be seen that it consists of 3 stages:

---

<sup>1</sup>See how we run in through in Chapter 5

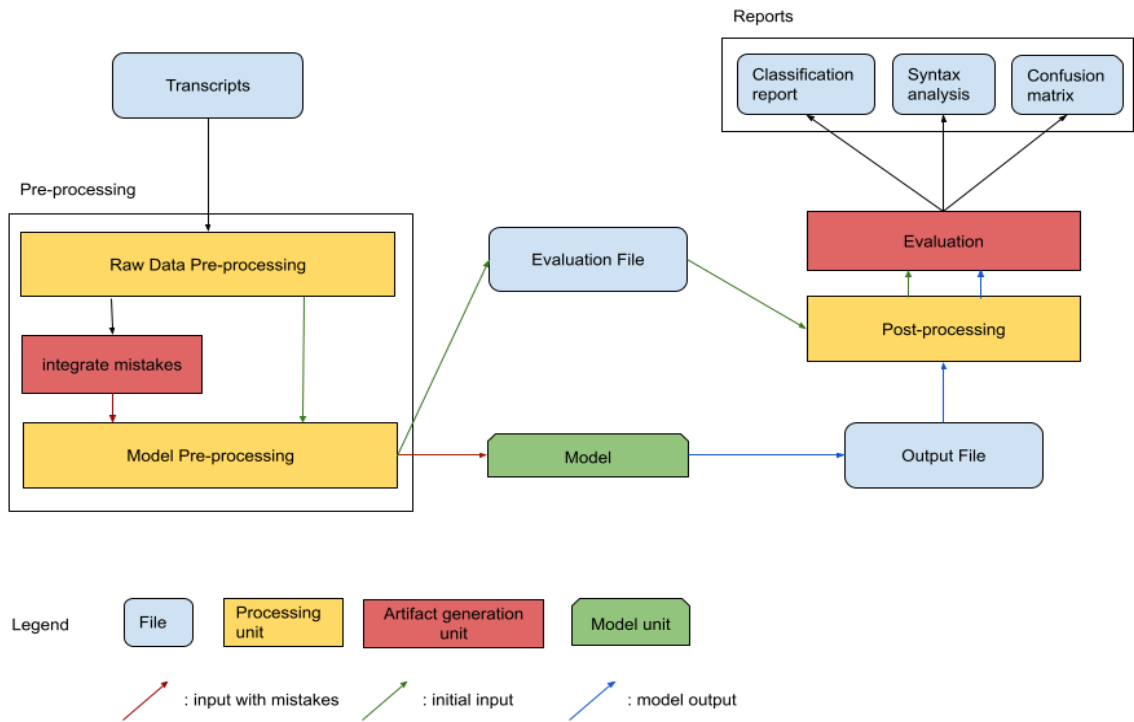


FIGURE 6.1: Evaluation Pipeline

1. Pre-processing:

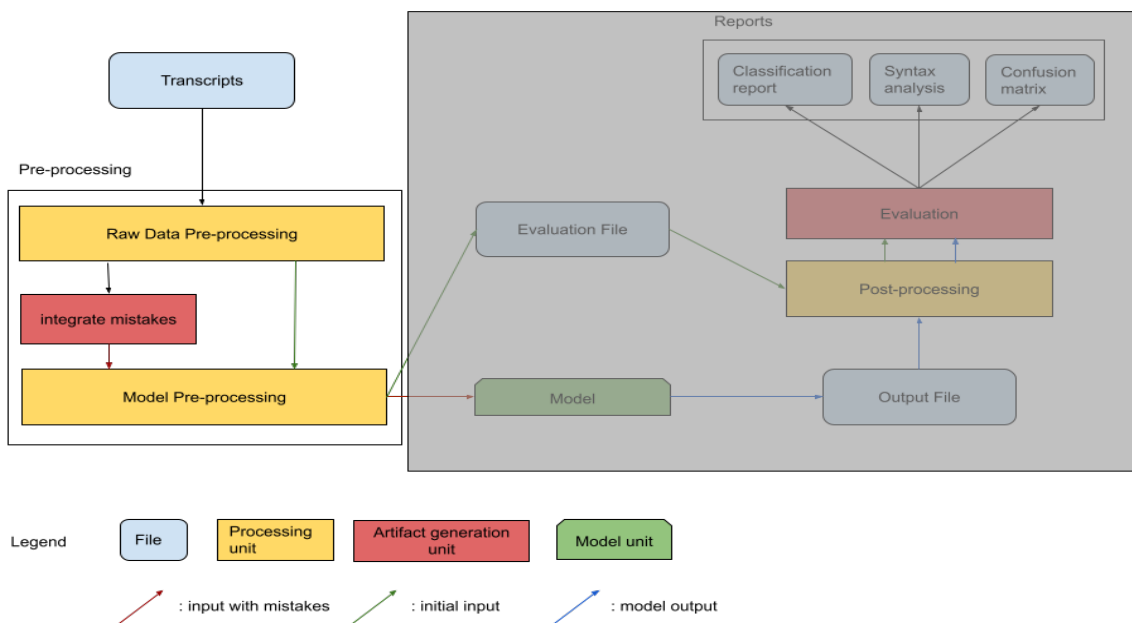


FIGURE 6.2: Architecture pre-processing

Within this stage all the raw data are processed and restructured in such a way,

that our punctuation models can take them as input. Depending on the type of punctuation model and corpus used, we preprocess differently. At the end, two files are generated. One serves as input for our models while the other is an evaluation file that serves as the ground truth for calculating scores / validation at the end of the whole pipeline.

## 2. Punctuation:

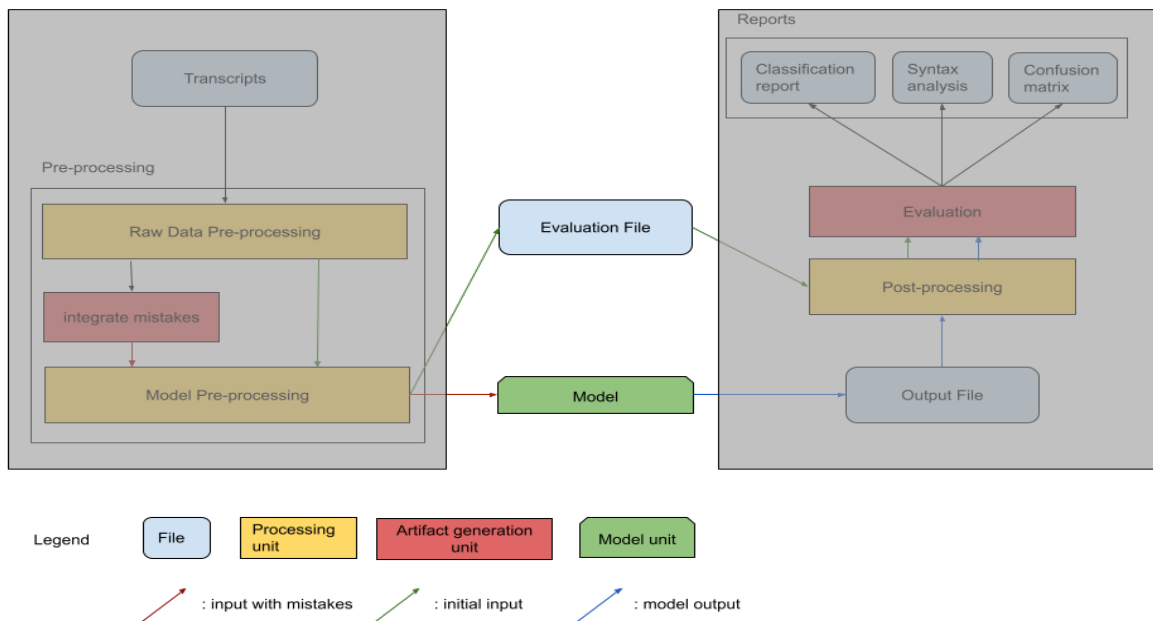


FIGURE 6.3: Architecture punctuation

In this part of the pipeline we feed our preprocessed data to the punctuation models. Outputs with restored punctuation marks are generated which are used for evaluation and analysis.

## 3. Postprocessing and analysis:

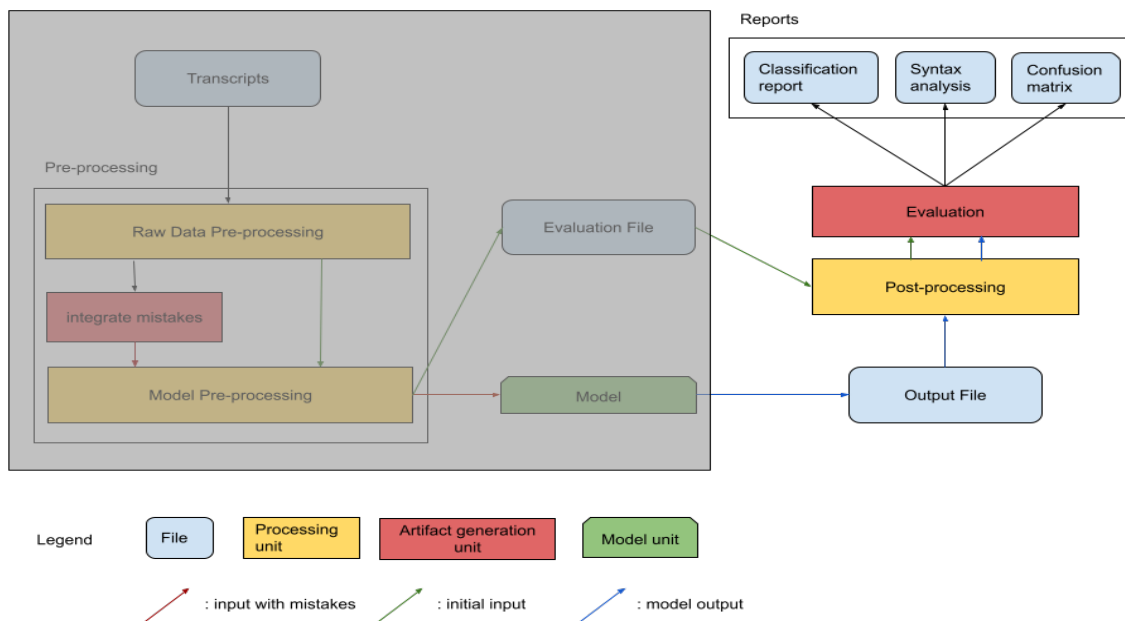


FIGURE 6.4: Architecture postprocessing

In postprocessing the generated outputs are further cleaned so that they can be evaluated by comparing them to the ground truth. Finally, in this last step reports are produced for analysis.

## Preprocessing

During this stage we gather the raw data and see how we need to modify the texts so that they can be accepted by our punctuation models. Depending on the corpus different measures had to be taken:

- ICSI Corpus:  
First of all, all words had to be extracted from the XML files. They are saved in a text file. In a second step, we remove empty lines. If the pipeline goes through `punctuator2`, we also preprocess following words in a set manner for model input:

1. gonna → going to
2. gotta → got to
3. wanna → want to

Finally, we output 2 copies, namely the evaluation file in `evalFiles`<sup>2</sup> folder and the test file as input for the `punctuator` in the `data` folder<sup>3</sup>.

- BBCNews dataset:  
This dataset already comes in text format so we only do preprocessing for

<sup>2</sup><https://github.zhaw.ch/manaiale/PA/tree/pipeline/evalFiles>

<sup>3</sup>[https://github.zhaw.ch/manaiale/PA/tree/pipeline/punctuation\\_models/punctuation-restoration/data](https://github.zhaw.ch/manaiale/PA/tree/pipeline/punctuation_models/punctuation-restoration/data)

the model. The preprocessing part involves removing hyphen and redundant white spaces as well as duplicate punctuation marks. Same as in the ICSI corpus, we remove empty line spaces so that it can be accepted by the punctuator.

- Interscriber dataset:  
Not much preprocessing has to be done by texts generated by the Interscriber tool. Since we only used a small dataset, manually copying and pasting it to a text file was sufficient for generating an experiment input and evaluation file.

### Punctuation prediction

In the main part of our pipeline the punctuation models take the preprocessed files as input. All punctuations are removed, predicted and restored by the model. All experiments were conducted by the two models:

- Punctuation Restoration<sup>4</sup>
- Punctuator2<sup>5</sup>

### Postprocessing

In postprocessing the files produced by our punctuation models are further processed to increase validation. Generated text from the previous stage is given as input for the postprocessing. Depending on the punctuation model used, different kind of errors that are generated in the output are cleaned. Double empty spaces or faulty modifications to words are corrected. The corrected texts are then analysed and used for calculating model performance with the metrics explained in chapter 2.4.

### Evaluation

In this step, model performance is measured. We use accuracy, recall, F1-score and support to generate a classification report. Also, a confusion matrix is produced in order to see how the model decided to set different punctuation marks compared to the ground truth. We use functions from the sklearn<sup>6</sup> module to generate these reports. Lastly, a docx document is generated using the docx<sup>7</sup> module to manually compare the result with the evaluation file. In there, punctuation mistakes are color coded to show what kind of errors occurred in order to better understand our statistics.

## 6.2.2 Experiments

In this section we elucidate on the procedures of our experiments. We use three different datasets of different levels of readability and conduct experiments with them. In the first Iteration, we use the best readable dataset, namely the BBCNews dataset since it is written text by journalists. For Iterations two and three we use the ICSI and Interscriber datasets in order to do an analysis of less readable text.

---

<sup>4</sup><https://github.com/xashru/punctuation-restoration>

<sup>5</sup><https://github.com/ottokart/punctuator2>

<sup>6</sup><https://scikit-learn.org/stable/>

<sup>7</sup><https://python-docx.readthedocs.io/en/latest/>



### **Iteration 1**

In Iteration one, the BBCNews dataset is run twice through the pipeline with a different punctuator each run. The purpose of this is to see how the models perform with correctly hand written texts. The outputs generated are used to analyse each run. We look at each report individually by analysing the scores, classification reports and text.

### **Iteration 2**

In Iteration 2, we concatenate 20 pre-processed ICSI files(see Chapter 3 Section 1) run them through both models, collect the output and analyse it comparatively to Iteration 1's results. We use the result we got during Iteration 1 to guide our search and analysis in Iteration 2.

### **Iteration 3**

Iteration 3 is the final Iteration. It builds on the discoveries of both previous Iterations to draw conclusions on an interview audio file which was run through Interscriber. We'll proceed with the same analysis done in both other Iterations and consequently compare all three to each other.

### **Human Evaluation**

We also do experiments involving human evaluation. It is important to see how people actually perceive our generated outputs. For this we take the interscriber output. We select randomly 5 paragraphs of different lengths and their 3 different versions of punctuation. They are processed to be all lower cased. In the end, we have 5 blocks containing three identical texts each with different punctuations from punctuation restoration, punctuator2 and Interscriber. Within these blocks we shuffle the texts randomly and survey people on which text is the most readable within the blocks. In the end, we gather all the points together.

## Chapter 7

# Results

This chapter elaborates and discusses the results from the experiments described in chapter 6.

### 7.1 Initial analysis

For the syntax analysis, following color coordination was used:

- **this**: Segmentation mistake
- **this**: Punctuation mistake
- **this**: Capitalization mistake
- **this**: Special/unforeseen mistake
- **this**: Disfluencies

#### 7.1.1 Syntax result

1 **Aufgefallen** dass er **schweizerdeutsch** geredet hat **starken** **Dialekt** und  
 ah dann transkribiert **Ich bin perfekt also praktisch fehlerfrei** .

FIGURE 7.1: Text snippet from the recording from a Teams meeting

1 Also **Ich** mochte mich mal ein bisschen vom Innenminister **losen weil**  
 2 hier spricht die **kulturstaatsministerin** **F** seine  
**begrifflichkeit** **Wir sind nicht Burka** hatte ich nie gewählt  
**aber die Frage die** .

FIGURE 7.2: Text snippet from the interview

### 7.1.2 Error distribution

File	Segmentation mis.	Punctuation mis.	Capitalization mis.	Special mis.	Disfluencies
Meeting_1	29	39	10	17	14
Meeting_2	74	55	30	30	17
Interview_1a	6	54	29	15	0
Interview_1b	5	86	49	33	0

TABLE 7.1: Error distribution per Input File

Total Mistakes	% Segmentation	% Punctuation	% Capitalization	% Special	% Disfluencies
595	0,191596639	0,393277311	0,198319328	0,159663866	0,057142857

TABLE 7.2: Overall Mistake percentages

Total Mistakes	% Segmentation	% Punctuation	% Capitalization	% Special	% Disfluencies
315	0,326984127	0,298412698	0,126984127	0,149206349	0,098412698

TABLE 7.3: Meetings Mistake percentages

Total Mistakes	% Segmentation	% Punctuation	% Capitalization	% Special	% Disfluencies
280	0,039285714	0,5	0,278571429	0,171428571	0,010714286

TABLE 7.4: Interview Mistake percentages

### 7.1.3 Interpretation

We'll do a distinction between the Meeting recordings and the Interview recordings due to their strong differences in nature.

Meetings have bad microphone quality, jumpy noise, speakers cut each other, and there is a mix of English and German and many disfluencies which normally appear during free speech.

Interviews have, contradictingly, have a clearer microphone input, speakers cut each other less, talk in longer sentences and use an elaborate vocabulary.

Thus, we'll list 2 of the most recurrent mistakes per File Type:

- Meetings:
  1. Segmentation mistakes because this audio input possesses more speakers which are speaking over each other often
  2. Punctuation mistake due to unclear sentences and people cutting each other
- Interviews:
  1. Punctuation mistakes happening because of difficult and long sentences
  2. Capitalization mistakes caused by a difficult and extensive vocabulary

### 7.1.4 Conclusion

Segmentation being clearly a relevant artifact isn't, unfortunately, in the scope of our work. Segmentation mistakes are most likely due to a wrong SpeechToText translation which isn't included in our Projektarbeit.

Additionally, we can see that for both file types, punctuation mistakes are extremely recurrent and make the text difficult to read for human beings.

Therefore, we revised the objectifs to correct punctuation mistakes.

## 7.2 Main analysis

### 7.2.1 Iteration 1

Our experiments in Iteration 1, described in chapter 6 show the following results:

#### Punctuation restoration

	precision	recall	f1-score	support
!	0.00	0.00	0.00	4
,	0.50	0.69	0.58	209
.	0.83	0.85	0.84	323
0	0.99	0.97	0.98	5842
:	0.00	0.00	0.00	8
?	0.86	0.67	0.75	9

TABLE 7.5: Report generated by using Punctuation restoration on BBCNews dataset

Initially, it can be seen that the model performs placing a dot quite well with a recall of 0.85. Even the precision and f1-score stay consistent. When it comes to commas though, it performs worse with a recall of just 0.58.

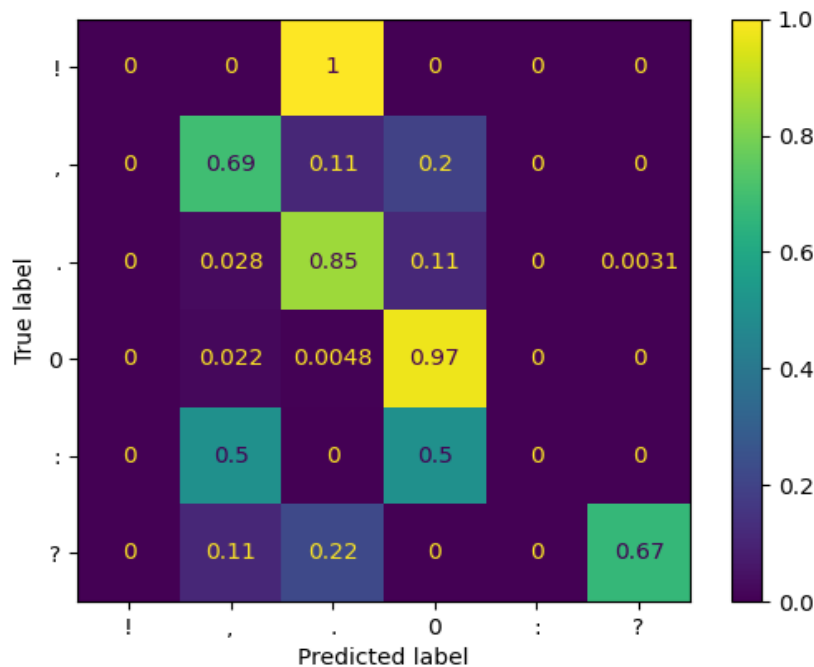


FIGURE 7.3: Punctuation restoration classification report BBC

When looking at the classification report, it can be seen that 11 per cent of dots did not get predicted. Also, when analysing the commas it becomes clear that 11 per cent of them were predicted as dots whereas 20 per cent are missing.

For the syntax analysis we show a few sentences and highlight some of the typical mistakes that were made. Next to these the ground truth is displayed in brackets:

```

1 Stephen Toulouse, a Microsoft security manager, said the flaws were
  known about .[]and although the firm...
2 These were considered to be less critical.[,] however,[.] If not
  updated, either automatically or manually, ...
3 To be infected,[] users must open up the attachment travelling with
  the message,[.] which bears the code...

```

FIGURE 7.4: Text snippet from punctuation restoration output on BBCDataset

**Conclusion** Generally, we can see that the model performs quite well on written texts such as news articles. It produces a relatively high f1-score when it comes to predicting dot placement, but shows some weakness when placing commas. One typical mistake that can be easily removed by postprocessing is the dot before an "and". Also, the only mistake when it comes to commas is when they are predicted as dots and mistaking commas for dots does not hinder readability in a massive way in our opinion. In a few cases, commas are placed after introductory clauses which are intuitively correct, even though in the ground truth they are often omitted.

**Punctuator2**

	precision	recall	f1-score	support
!	1.00	0.75	0.86	4
,	0.47	0.89	0.62	209
.	0.66	0.95	0.78	323
0	1.00	0.94	0.97	5842
:	0.53	1.00	0.70	8
?	1.00	0.78	0.88	9

TABLE 7.6: Report generated by using Punctuator2 on BBCNews dataset

Even though this model has a high recall when placing a dot, it has only an f1-score of 0.78. Also, the commas do not show a good result by having an f1-score of 0.62.

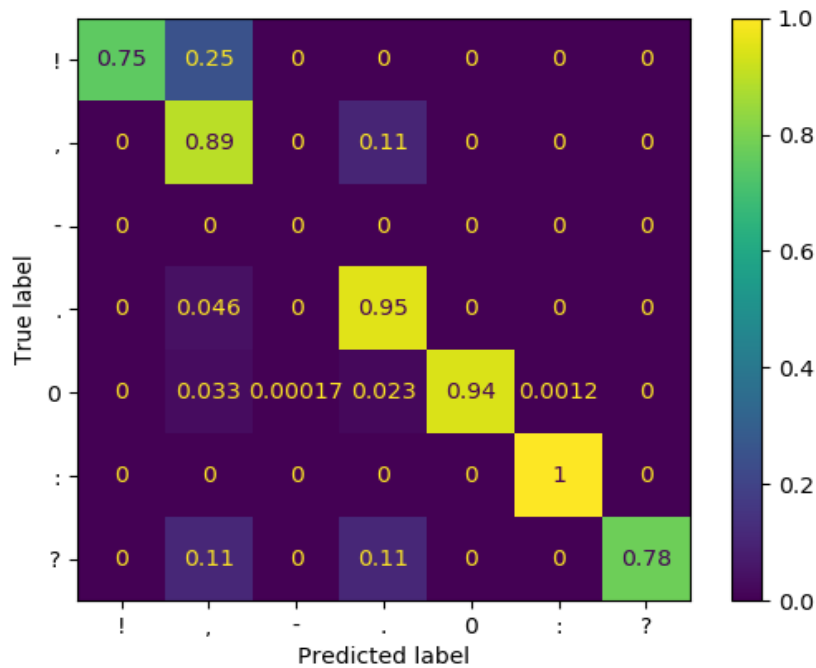


FIGURE 7.5: Punctuator 2 classification report BBC

When looking at the classification report, it can be seen that 11 per cent of dots did not get predicted. Also, when analysing the commas it becomes clear that 11 per cent of them were predicted as dots whereas 20 per cent are missing.

When doing a syntax analysis like in the previous section we observe following patterns:

```

1 The Kyrgyz Republic,[] a small, mountainous state of the former
   Soviet republic,[] is using invisible, ink and ultraviolet readers...
2 If.[] The ink shows under the UV light the voter will not be allowed
   to enter the polling.[] Station.

```

FIGURE 7.6: Text snippet from punctuator2 output on BBCDataset

**Conclusion** The punctuator2 model performs moderately well on the used dataset. There are no dots or commas that were predicted as empty spaces. When looking at dots we see that they are predicted as commas sometimes. Most common mistakes are doubled punctuation marks and random dots followed by capitalized letters since it starts a new sentence. These problems considerably decrease experience when reading the text. This outcome could be derived from the fact, that this model was mainly trained on parliament texts and ted talks. Considering the fact that the used dataset consists of news articles covering technical and sports topics the results are understandable.

## 7.2.2 Iteration 2

Our experiments in Iteration 2, described in chapter 6 show the following results:

### Punctuation restoration

	precision	recall	f1-score	support
!	0	0	0	28
,	0.33	0.73	0.45	3039
.	0.71	0.55	0.62	3111
?	0.52	0.51	0.51	453

TABLE 7.7: Report generated by using Punctuation Restoration on x20 ICSI dataset

Certain labels were omitted for reasons of weak importance and information holding:

- Label "0" was omitted because it presented with near perfect results 0.95 minimum precision and for an enormous support. In our context, this result isn't important. We'd much more focus on ending or segmenting punctuations.
- Label " : " which had a support of 2 wasn't strong of significance

We see a small Precision for commas which could lead to believe that this result is bad, but further contextual analysis is necessary, because cases of swapping of commas and dots could still yield a grammatically correct and readable output even though this score might lead to think otherwise.

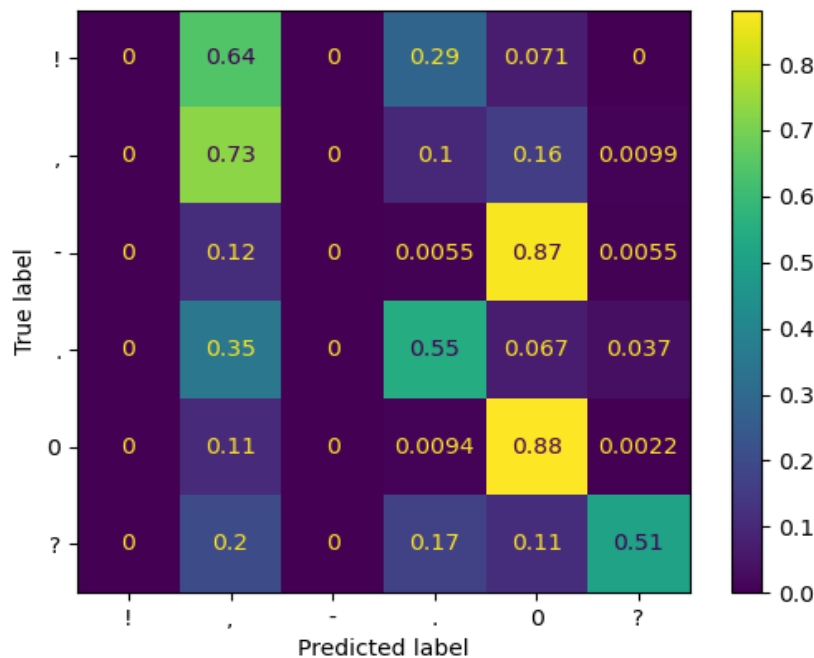


FIGURE 7.7: Punctuation Restoration Confusion matrix x20 ICSI

Observing the Confusion Matrix several results can be labeled as worrisome and others not. For example, that "!" have been predicted as "," for 64% is problematic because a stopping punctuation has been predicted as a comma for most occasions. Then again, values such as 29% of "!" have been predicted as "." aren't a problem because both can be used interchangeably in most times and still be readable.

Consequently, we'll list all worrisome results related by the Confusion Matrix: (we'll use "→" as "predicted as")

- "!" → "," for 64%: we can see in the text snippets 1 & 2 of the Syntax analysis that the usage of "," instead "!" is reasonable and logical.
- "," → "." for 10%: text snippets 3 & 4 in Figure 7.8 show that "." have been put logically after or before words, such as "Right" or "but", which could signal the start of another sentence, especially in spoken language. Mostly, looking at the whole output, "." were put before words like "because", "I mean" and "and" which makes sense grammatically and stays perfectly readable.
- "." → "," for 35%: The text snippet 5 is recurrent where "," are predicted before "Hmm" or after "Yeah". This scenario is rather a question of writing style than a question of right or wrong making this prediction acceptable.
- "?" → "," for 20%: Predictions that we can see in text snippet 6 happen most of the time for those predictions. These mistakes are clearly unproblematic. The text is still correct and perfectly readable.



Syntax analysis<sup>1</sup>:

```

1 ...should, Andreas was saying, Yeah, Ach,[!] Well then, But then...
2 ...intentions. So I thought, " Mmm,[!] Maybe for a deep...
3 ...to take pictures of. Or very rarely.[,] but you usually...
4 ...different type.[,] Right? Right? Yep, Right.[,] ...
5 ...have question mark because it's extracted. Yeah,[.] Hmm. ...
6 ... that have these features, OK,[?] and then you'd like...

```

FIGURE 7.8: Text snippets from punctuation restoration for Iteration 2

**Conclusion** Thanks to the analysis, we can observe that most mistakes are inoffensive. These predictions seem to appear around words such as *"because"*, *"I mean"*, *"and"* and *"Hmm"* which are very frequent in spoken language. But, generally, predictions done by the punctuation restoration model in Iteration 2 seem to be very promising.

### Punctuator2

	precision	recall	f1-score	support
!	0	0	0	28
,	0.23	0.51	0.32	3039
.	0.15	0.11	0.13	3111
?	0.3	0.14	0.19	453

TABLE 7.8: Report generated by using Punctuator2 on x20 ICSI dataset

Certain labels were omitted for reasons of weak importance and information holding:

- Label "0" was omitted because it presented with near perfect results 0.95 minimum precision and for an enormous support. In our context, this result isn't important. We'd much more focus on ending or segmenting punctuations.

We see a small metrics overall indicating an sub-optimal model, but then again it needs to be seen in context to agree or disagree with the results from the classification report.

<sup>1</sup>For the syntax analysis we show a few sentences and highlight some of the typical mistakes that were made. Next to these the ground truth is displayed in brackets

		Predicted label				
		!	,	.	?	0
True label	!	0.0	1.0	0.0	0.0	0.0
	,	0.0	0.51	0.153	0.007	0.330
	.	0.0	0.37	0.109	0.013	0.507
	?	0.0	0.272	0.052	0.143	0.532

FIGURE 7.9: normalized Confusion Matrix for punctuator2 model on x20 ICSI for Iteration 2

Observing the Confusion Matrix (Figure 7.9), we can filter out several instances: (we'll use "→" as "predicted as")

- "!" → "," for 100%: We can see in text snippet 1 of Figure 7.10, which is the only occurrence of this misclassification, that both punctuations are interchangeable keeping the meaning of the sentence intact. A necessary step to keep a comfortable reading is to post-process the capitalization of "And", but that is a rather simple extra step.
- "," → "." for 15%: The times this misclassification happened (see text snippet 2 & 3 for example) the choice taken by the model is reasonable, keeping it readable.
- "." → "," for 37%: Text snippets 4 & 5 summarize all the mistakes done in this case. We can observe that in both cases the text stays fluid, even though the context of text snippet 5 seems to be broken, because the listing of numbers was intended for a phone number in that case and the model didn't break up with a dot at the right spot making the listed number senseless. A possible solution to this problem is to have a human post-process these rare situations which shouldn't be too much of a hassle.
- "." → "0" for 50%: We can deduce from text snippets 6 & 7 that this model likes to omit punctuation before junction words such "and" & "so" which makes sense and keeps the flow and meaning of the text the same. Whereas, in text snippet 8, we can see that the omission of "." destructures the text completely making it extremely unenjoyable to read and syntactically wrong furthermore.
- "?" → "," for 27%: This misclassification (text snippet 9 & 10) deemed to be problematic, because a clear sentence structure of a question was at hand, but

the model still did a mistake. Additionally, questions only read themselves in one way for humans and when it doesn't fit, it stands out immediately.

- "?" → "0" for 53%: Similarly to the misclassification described above for text snippet 11.

Syntax analysis<sup>2</sup>:

```

1 ...we did diff Exactly,[!] And, that's how Yeah But, you...
2 ...and so forth. Uh.[,] so, um, if...
3 ...Mm, hmm, Ah.[,] well, For...
4 ...data. OK Yeah,[.] Right,[.] Time time times...
5 ...nine, nine, zero,[.] seven, seven, nine...
6 ...for this sort of stuff [.] And. so the only question...
7 ...if you want to [.] So that, um...
8 ...particular items that it can take [.] Sure [.] If. you...
9 ...features in right,[?] Mmm. And...
10 ...what's the advantage of doing that versus just putting it into
    this format,[?] ...
11 ...Are? they ready to go or [?] ...

```

FIGURE 7.10: Text snippets from punctuator2 for Iteration 2

**Conclusion** Punctuator2 in Iteration 2 had underwhelming metrics, but each misclassification needed to be seen case by case to make sure of it. Seeing the Syntax analysis, we observed that for cases of '"," → "."' and '"," → ","' the mistakes kept the flow and meaning of the text intact. But in cases such as '"," → "0"' and '","? → ","', it made the predicted text unenjoyable to read and even incorrect at times. These created problems that are also not easily post-processed. Punctuator2 seems, therefore, to be difficult to use for texts similar to ICSI.

### 7.3 Iteration 3

Our experiments in Iteration 3, described in chapter 6 show the following results:

#### Punctuation restoration

	precision	recall	f1-score	support
!	0	0	0	1
,	0.43	0.78	0.55	49
.	0.68	0.51	0.58	75
?	0.33	0.67	0.44	3

TABLE 7.9: Report generated by using Punctuation Restoration on an Interview run through Interscriber

<sup>2</sup>For the syntax analysis we show a few sentences and highlight some of the typical mistakes that were made. Next to these the ground truth is displayed in brackets

Certain labels were omitted for reasons of weak importance and information holding:

- Label "0" was omitted because it presented with near perfect results 0.95 minimum precision and for an enormous support. In our context, this result isn't important. We'd much more focus on ending or segmenting punctuations.
- Label " : " which had a support of 2 wasn't strong of significance

We can see promising metrics for the Precision of ".", all Recalls and the f1-score of "." and ",". Let's look at it more in detail with the Confusion Matrix and Syntax analysis to deduce if the model is applicable for data similar to an interview output of Interscriber.

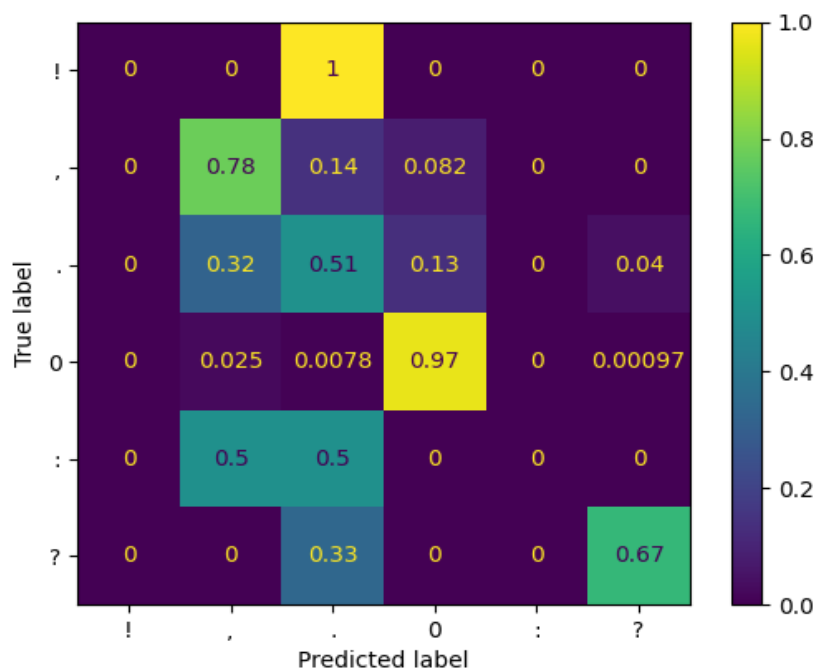


FIGURE 7.11: Punctuation Restoration Confusion matrix Interview Interscriber

Worrisome results from the Confusion Matrix are listed here: (we'll use "→" as "predicted as")

- ", " → "." for 14%: We have here another occurrence of the model where the model cuts a sentence short at a joining word like "so" (see text snippets 1 & 2 in Figure 7.12). The created sentences stay fully readable and keep their flow which is essential for the reader.
- "." → "," for 32%: In text snippets 3 & 4, we observe that the model, in this case, decided to place a "," making the sentence still understandable and fully readable. A post-processing step still needs to be taken to correct the capitalization mistakes.
- "." → "0" for 13%: We can understand from text snippets 5 & 6 that the "." was misplaced to later in the text. After analysing the text more, we summarize that

this displacement of the "." still makes the text perfectly understandable and comfortable to read.

Syntax analysis<sup>3</sup>:

```

1 ...you move into the way of another.[,] so the amount of rain...
2 ...the number of meters You travel.[,] so to stay...
3 ...depend at all on its slant,[.] Then no matter...
4 ...drops from the side,[.] And you'll get...
5 ...the number of meters [.] You travel. so to stay...
6 ...has to lead the world [.] Getting to zero greenhouse...

```

FIGURE 7.12: Text snippets from punctuation restoration for Iteration 3

**Conclusion** We can conclude that in Iteration 3, punctuation restoration performed extremely well. The text keeps its context and flow intact. The readability is surprisingly good. This model seems very promising for this use-case.

### Punctuator2

	precision	recall	f1-score	support
!	0	0	0	1
,	0.35	0.51	0.42	49
.	0.57	0.32	0.41	75
?	0.0	0.0	0.0	3

TABLE 7.10: Report generated by using Punctuator2 on Interview in Interscriber

Certain labels were omitted for reasons of weak importance and information holding:

- Label "0" was omitted because it presented with near perfect results 0.95 minimum precision and for an enormous support. In our context, this result isn't important. We'd much more focus on ending or segmenting punctuations.

Knowing the fact that the support is low, we have rather small results suggesting an inadequate model for this task. Then again, the context needs to be analysed to be certain.

This model has the specificity to capitalize the next word after having predicted a "." at some position.

<sup>3</sup>For the syntax analysis we show a few sentences and highlight some of the typical mistakes that were made. Next to these the ground truth is displayed in brackets

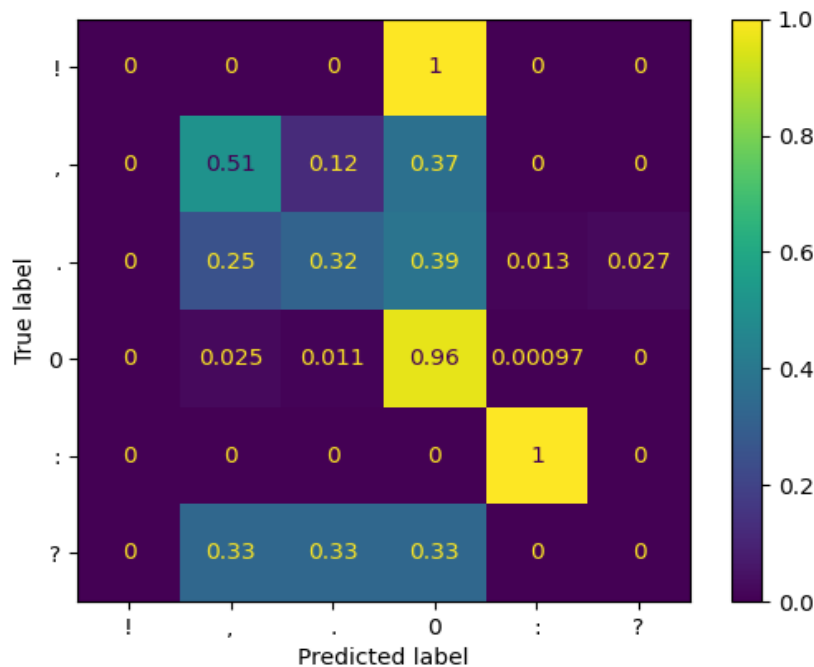


FIGURE 7.13: Punctuator2 Confusion matrix Interview Interscriber

Observing the Confusion Matrix (Figure 7.13), we should observe these misclassifications:

(we'll use "→" as "predicted as")

- ", " → "." for 12%: Both classifications are interchangeable and perfectly readable. (see Text snippets 1 & 2 in Figure 7.14)
- "." → ", " for 25%: In text snippet 3 & 4, we can find predictions which are plausible and surely still readable even though one small step of post-processing needs to be made.
- "." → "0" for 39%: Text snippet 5 is one part of the mistakes happening of that type, but in that case the text stays readable. Whereas in text snippet 6, the flow of the sentence gets broken and the sentence simply doesn't make sense anymore.
- "? " → ", " for 33%: This instance (text snippet 7) is interesting, because the model fully reconstructed the sentence with the surrounding words and context making it still understandable and logical in the reading of the text.

Syntax analysis<sup>4</sup>:

```

1 ...more educated people than ever[,] We have a generation...
2 ...made in a different way[,] The steel in the...
3 ...change, new ideas[,] That's way greater...
4 ...Europe every year[,] Exactly The...
5 ...has to lead the world[,] Getting to zero...
6 ...has ever faced[,] Absolutely...
7 ...low hanging fruit,[?] Passenger cars, Part...

```

FIGURE 7.14: Text snippets from punctuator2 for Iteration 3

**Conclusion** This model seems to do well in certain instance, but does very poorly on others such as `'".' → "0"` creating illogical sentences which make the text difficult to read.

### 7.3.1 Human Evaluation

In this section we show how humans evaluate our outputs, though only a small sample can be evaluated on a small group of people. In our survey 40 people participated. Results can be seen in in the following chart:

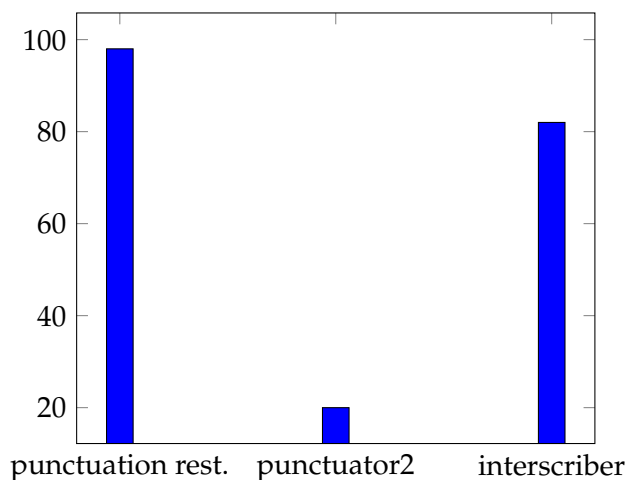


FIGURE 7.15: Results from human evaluation

**Conclusion** Much like our previous analyses we observe that the punctuation restoration model is seen superior as punctuator2. It is perceived better than the interscriber and shows the best results by getting voted 98 times as the best readable text within a block. The interscriber tool received 82 votes and punctuator2 only 20. This survey serves as an first impression and illustrates our previous analysis in a positive way. However, the small amount of people, namely 40, is not enough to come to a definitive conclusion. Additionally, this experiment would have to be conducted with many more text examples than just 5 paragraphs. Nevertheless, it gives us a convincing first impression.

<sup>4</sup>For the syntax analysis we show a few sentences and highlight some of the typical mistakes that were made. Next to these the ground truth is displayed in brackets

## Chapter 8

# Encountered Issues

### 8.1 Disfluency

In this section we talk about our thoughts and goals when considering disfluency removal as a means to improve readability.

#### 8.1.1 Motivation

During our meetings, one of the methods for improving reading quality that we considered was disfluency removal. Intuitively, it made sense that removing stutterings and word / sentence repetitions could result in a cleaner text.

#### 8.1.2 Work

For our work, we tried an initial analysis using the deep disfluency <sup>1</sup> module. We did a small experiment on a few sentences and produced some promising results. However, scoring proved to be a difficult task as we didn't know what kind of metric should be used to evaluate readability after disfluency removal.

#### 8.1.3 Issue

Disfluency removal in itself is a complex problem that required much more research and time. In the end, we decided to leave this topic for future works as we focused more on punctuation restoration for our task.

### 8.2 Curl request for Punctuator2 model

#### 8.2.1 Motivation

During the inserting of the punctuator2 model into our pipeline, we had several options to go with. We could clone the repository or use an in-built API curl request where we could send our text to punctuate and it would come back to us punctuated, which was rather appealing to us so we could avoid the inconvenience of having to install libraries etc... We simply had some pre-processing steps to take, but it seemed rather simple and faster to implement.

---

<sup>1</sup>[https://github.com/clp-research/deep\\_disfluency](https://github.com/clp-research/deep_disfluency)



### 8.2.2 Work

To be able to insert this curl request into our pipeline we had several pre-processing steps to follow:

- text needed to be lower cased
- concatenated
- formatted into utf-8 adding every ASCII breaking points
- finally, add 'text=' in front of the desired text to punctuate

Having formatted the input text, we introduced the curl request into our bash script and it worked. We received a punctuated text in return.

### 8.2.3 Issue

During the evaluation of the input, we observed a discrepancy between the predicted labels and the ground-truth labels in size. After long hours of debugging, we found out that their model would change words such "gotta", "wanna" and "gonna" into their written form "goingto", "wantto" and "goingto". These permutations wrongly created predictions labels in our algorithm.

To remedy to that problem, we cloned the repository and changed parts of the source code to fit our needs. After some time, it worked and we could get a perfect output that fits our evaluation pipeline.

## 8.3 Sentence Compression

This section discusses our ideas and problems encountered with the method of sentence compression.

### 8.3.1 Motivation

Reducing a long sentence to a smaller size without significant information loss was one approach we deemed appropriate for an improved reading experience. The goal was to see what impact such a shortening of a sentence could have on the readability of text and whether or not it still conveyed the same meaning before compression.

### 8.3.2 Issue

One issue we encountered were with available tools for sentence compression. First of all, the number of github repositories available for testing this task were limited and if available, offered minimalistic documentation. Almost none of these offered publicly available pretrained models. Also, we encountered dependency problems where programs and modules had compatibility issues. We did some small tests using sentence simplification from this github page<sup>2</sup>, but since it was not the focus of our task we decided to close this topic and leave it for future works due to much more required time for research.

---

<sup>2</sup><https://github.com/garain/Sentence-Simplification>

## 8.4 General Pipeline

### 8.4.1 Motivation

The general pipeline was an idea that we thought of during the same time we were building the evaluation pipeline. The general pipeline was a first draft that conceptualized the communication of Interscriber and our evaluation pipeline.

### 8.4.2 Work

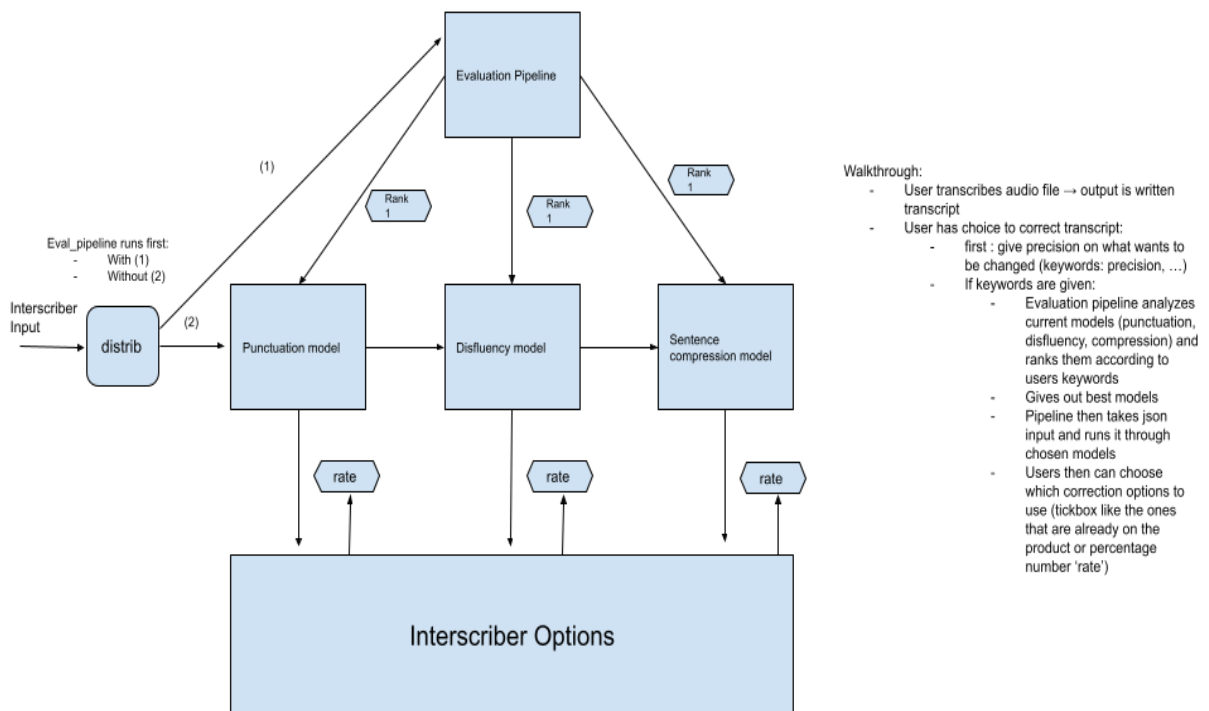


FIGURE 8.1: General Pipeline Idea

### 8.4.3 Issue

Unfortunately, this work turned out to be out of scope of our Projektarbeit. So we dropped the idea, but it could maybe be used as a first idea later on.

## **8.5 Learning the Theory**

### **8.5.1 Motivation**

We, both, are very interested in Machine Learning and want to understand how things function before we blindly use them. We wanted to develop a real understanding of the models, we were using, so we could evaluate and understand better where mistakes could come from.

### **8.5.2 Work**

It was extremely instructive, but also very difficult to keep track and understand everything that we used. We spent several hours each week, especially at the beginning, looking at lectures online, reading chapter of books, reading papers and experimenting with the code to be able to understand our models better.

### **8.5.3 Issue**

It's not an issue per say, because we really wanted to understand, but was a very challenging part of our Projektarbeit that we wanted to put forward.

## Chapter 9

# Conclusion

As described in our objective in chapter 1.2 we aimed to improve readability of transcriptions of spoken English by appropriate punctuation detection. Thus, we analysed two punctuation models on different text that vary in quality. As expected they performed best in written language, generating recall scores around 0.8 when punctuating dots. In other texts with worse quality due to speech transcription we observed lower scores. However, scores alone are not the only factor to be considered when evaluating models in NLP domains. By manually reading the outputs of our predictors we observed that mistakes did not significantly distort context and worsen readability. Except of a few basic mistakes we noticed better placed punctuation. In the end, having generated text files with more appropriate punctuation positioning, we observed an increase in quality and perceived readability. Our survey with 40 people confirmed our findings, though much more needs to be done in a larger scale.

In a next step training the models with other domain datasets could prove to be helpful in this task. Adding other punctuators into our evaluation pipeline could also prove to be useful. Involving other approaches like disfluency removal, sentence compression and fixing incomplete sentences could further enhance readability.

## Chapter 10

# Future Works

In this chapter we discuss how future works could improve on our learnings in improving readability. An algorithm based on the punctuation models used in this work could be built.

### 10.1 Agreement

The idea is to develop an agreement between both models punctuation restoration and punctuator2. We hope that by taking the decisions of both into consideration that we could get better results in detecting appropriate places for punctuation and therefore improve readability. Following algorithm is proposed by comparing outputs from the models. Let out1 be the output of punctuation restoration and out2 be the output of punctuator2. We then do the following:

---

**Algorithm 1** An agreement algorithm for two punctuator models

---

```

for p1 in out1, p2 in out2 do
  if p1 equals p2 then
    Choose p1
  else if p1 equals dot and p2 equals comma then
    Choose p1
  else if p1 equals comma and p2 equals dot then
    Choose p2
  else if p1 equals empty space and (p2 equals dot or comma) then
    Choose empty space
  else if p2 equals empty space and (p1 equals dot or comma) then
    Choose empty space
  end if
end for

```

---

With this algorithm we cover three cases. If both agree on the prediction, we simply punctuate like suggested. When they are in a disagreement where they cannot agree between dot and comma we simply choose the predictor that suggests a dot. However, if one of them predicts empty space while the other one predicts a punctuation mark we simply place an empty space, as there is likely to be a false positive.

# Bibliography

- [1] John Lee, Stephanie Seneff "Automatic Grammar Correction for Second-Language Learners" Conference Paper (Jan. 2006)
- [2] Takaaki Hori, Daniel Willett, and Yasuhiro Minami "Paraphrasing spontaneous speech using weighted finite-state transducers" (Apr. 2003)
- [3] Ottokar Tilk, Tanel Alumae "LSTM for Punctuation Restoration in Speech Transcripts" (Jun. 2015)
- [4] Advait Siddharthan "An architecture for a text simplification system" Language Engineering Conference (Dec. 2002)
- [5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville "Deep Learning", Chapter 10.2 (2015)
- [6] Sepp Hochreiter, Jürgen Schmidhuber "Long Short-Term Memory" (1997)
- [7] Alex Graves, Santiago Fernandez, and Jürgen Schmidhuber "Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition" (Jan. 2005)
- [8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville "Deep Learning", Chapter 10.10 (2015)
- [9] Afshine Amidi, and Shervine Amidi from Stanford University, "CS230 - Deep Learning" Lecture, Chapter GRU/LSTM (2020)
- [10] Tuggener Don, Aghaebrahimian Ahmad The Sentence End and Punctuation Prediction in NLG Text (SEPP-NLG) Shared Task 2021 (Sep, 2021)
- [11] Gregory Grefenstette, Pasi Tapanainen "What is a word, what is a sentence? problems of tokenization", (1997)
- [12] Jagroop Kaur and Jaswinder Singh "Deep neural network based sentence boundary detection and end marker suggestion for social media text" (2019)
- [13] Roque Lopez and Thiago A. S. Pardo. Experiments on sentence boundary detection in user-generated web content. In Computational Linguistics and Intelligent Text Processing, pages 227–237, Cham. Springer International Publishing, (2015)
- [14] Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio "Neural machine translation by jointly learning to align and translate" (2015)
- [15] Schuster, M. and Paliwal, K. K. "Bidirectional recurrent neural networks" (1997)
- [16] Ashish Vaswan, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin "Attention is all you need" (Dec 2017)
- [17] Tanvirul Alam, Akib Khan, Firoj Alam "Punctuation Restoration using Transformer Models for High-and Low-Resource Languages", (2020)

- 
- [18] Ottokar Tilk, Tanel Alumäe "Bidirectional Recurrent Neural Network with Attention Mechanism for Punctuation Restoration", (2016)
- [19] Adam Janin , Don Baron , Jane Edwards , Dan Ellis, David Gelbart Nelson Morgan, Barbara Peskin , Thilo Pfau Elizabeth Shriber, Andreas Stolcke and Chuck Wooters "The ICSI Meeting Corpus" California, International Computer Science Institute, (2003)
- [20] D. Greene and P. Cunningham. "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering", Proc. ICML (2006).

# List of Figures

2.1	Architecture of traditional RNN . . . . .	3
2.2	Logic of RNN block . . . . .	4
2.3	Bidirectional RNN architecture . . . . .	5
2.4	LSTM internal architecture . . . . .	5
2.5	BiLSTM Architecture . . . . .	6
2.6	Attention mechanism . . . . .	7
2.7	Transformer Architecture . . . . .	9
2.8	Sigmoid activation function . . . . .	11
2.9	Tanh activation function . . . . .	11
2.10	ReLU activation function . . . . .	12
5.1	Interscriber Upload tool . . . . .	18
5.2	Interscriber Edit Tool . . . . .	19
6.1	Evaluation Pipeline . . . . .	21
6.2	Architecture pre-processing . . . . .	21
6.3	Architecture punctuation . . . . .	22
6.4	Architecture postprocessing . . . . .	23
7.1	Text snippet from the recording from a Teams meeting . . . . .	26
7.2	Text snippet from the interview . . . . .	26
7.3	Punctuation restoration confusion matrix BBC . . . . .	29
7.4	Text snippet from punctuation restoration output on BBCDataset . . . . .	29
7.5	Punctuator 2 confusion matrix BBC . . . . .	30
7.6	Text snippet from punctuator2 output on BBCDataset . . . . .	31
7.7	Punctuation Restoration Confusion matrix x20 ICSI . . . . .	32
7.8	Text snippets from punctuation restoration for Iteration 2 . . . . .	33
7.9	normalized Confusion Matrix for punctuator2 model on x20 ICSI for Iteration 2 . . . . .	34
7.10	Text snippets from punctuator2 for Iteration 2 . . . . .	35
7.11	Punctuation Restoration Confusion matrix Interview Interscriber . . . . .	36
7.12	Text snippets from punctuation restoration for Iteration 3 . . . . .	37
7.13	Punctuator2 Confusion matrix Interview Interscriber . . . . .	38
7.14	Text snippets from punctuator2 for Iteration 3 . . . . .	39
7.15	Results from human evaluation . . . . .	39
8.1	General Pipeline Idea . . . . .	42



# List of Tables

7.1	Error distribution per Input File . . . . .	27
7.2	Overall Mistake percentages . . . . .	27
7.3	Meetings Mistake percentages . . . . .	27
7.4	Interview Mistake percentages . . . . .	27
7.5	Report generated by using Punctuation restoration on BBCNews dataset	28
7.6	Report generated by using Punctuator2 on BBCNews dataset . . . . .	30
7.7	Report generated by using Punctuation Restoration on x20 ICSI dataset	31
7.8	Report generated by using Punctuator2 on x20 ICSI dataset . . . . .	33
7.9	Report generated by using Punctuation Restoration on an Interview run through Interscriber . . . . .	35
7.10	Report generated by using Punctuator2 on Interview in Interscriber . .	37

# List of Abbreviations

<b>NLP</b>	<b>Natural Language Processing</b>
<b>RNN</b>	<b>Recurrent Neural Network</b>
<b>BiRNN</b>	<b>Bidirectional Recurrent Neural Network</b>
<b>RNN</b>	<b>Recurrent Neural Network</b>
<b>LSTM</b>	<b>Long Short Term Memory</b>
<b>BiLSTM</b>	<b>Bidirectional Long Short Term Memory</b>
<b>ICSI</b>	<b>International Computer Science Institute</b>
<b>BBC</b>	<b>British Broadcasting Corporation</b>
<b>IBM</b>	<b>International Business Machines</b>