Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

School of
Engineering

Bachelor Thesis

ZHAW School of Engineering

Centre for Artificial Intelligence

The Influence of Audio Length on the Performance of Swiss-German Speech Translation Models

Authors
Niklas Rijk van der Heide
Felix Matthias Saaro

Supervisor
Prof. Dr. Mark Cieliebak

Secondary Supervisor
Dr. Jan Deriu

June 9th, 2023

Declaration of Originality

By submitting this Bachelor's thesis, the undersigned student confirms that this thesis is his/her own work and was written without the help of a third party. (Group works: the performance of the other group members are not considered as third party).

The student declares that all sources in the text (including Internet pages) and appendices have been correctly disclosed. This means that there has been no plagiarism, i.e. no sections of the Bachelor thesis have been partially or wholly taken from other texts and represented as the student's own work or included without being correctly referenced.

Any misconduct will be dealt with according to paragraphs 39 and 40 of the General Academic Regulations for Bachelor's and Master's Degree courses at the Zurich University of Applied Sciences (Rahmenprüfungsordnung ZHAW (RPO)) and subject to the provisions for disciplinary action stipulated in the University regulations.

Winterthur, 09.06.2023	[illegible]
City, Date	Felix Saaro
Winterthur, 09.06.2023	[illegible]
City, Date	Niklas Rijk van der Heide

Abstract

Speech Translation models designed to convert spoken Swiss-German to written German have been in existence for some time. While these models generally perform well, their performance in various scenarios remains poorly understood. In this thesis, we explore the influence of audio length on the performance of Swiss-German speech translation models and identify the necessary factors for achieving better performance on longer audio segments.

To achieve this, we examined four speech translation models from different institutions. A model from the Zurich University of Applied Sciences (ZHAW), one from the University of Applied Sciences Northwestern Switzerland (FHNW), a model from Microsoft, as well as a model from OpenAI called Whisper.

We conducted eight different experiments using a Swiss-German corpus collected by the ZHAW and FHNW. In the experiments, the audio length was augmented in various ways. From there, we found that while the ZHAW, FHNW and Microsoft models showed a tendency to perform worse on longer duration, extending the duration by adding silence did not influence on the performance. Changing the playback speed has a negative influence on the ZHAW, Microsoft and Whisper models, both when speeding segments up or slowing them down. The FHNW model exhibited extraordinary robustness to changes in playback speed, as the results when accelerated by a factor of 1.25 were nearly identical to the results when the playback speed was not altered. The biggest influence on performance was when adding more than one sentence to a segment. Without a segmentation of the input audio the ZHAW, FHNW and Microsoft models performed badly, indicating that segmentation should be introduced as soon as more than one sentence appears in an audio recording. Training a model specifically on multi-sentence segments showed promising results, on single sentence segments and multi-sentence segments as well as in scenarios where sentences are split while segmenting the audio recordings. Comparing a sentence-based segmentation, which is considered ideal for models trained on single sentence segments, to a fixed-window segmentation with an overlap showed an almost identical result.

Examining the models on a real-life recording showed that the ZHAW (lowercase) and ZHAW (multi-sentence) models perform considerably worse than the FHNW, Microsoft and Whisper models. Indicating that more investigation is required to fully understand what makes a speech translation model work well in real-life scenarios.

Zusammenfassung

Speech-Translation Modelle, die gesprochenes Schweizerdeutsch in geschriebenes Deutsch übersetzten, existieren schon seit einiger Zeit. Obwohl diese Modelle im Allgemeinen gut funktionieren, ist ihre Leistung in verschiedenen Szenarien noch wenig erforscht. In dieser Arbeit untersuchen wir den Einfluss der Audiolänge auf die Leistung von Schweizerdeutsch-Sprachübersetzungsmodellen und identifizieren die erforderlichen Faktoren für eine bessere Leistung bei längeren Audiosegmenten.

Um dieses Ziel zu erreichen, haben wir vier Sprachübersetzungsmodelle von verschiedenen Institutionen untersucht. Ein Modell von der Zürcher Hochschule für Angewandte Wissenschaften (ZHAW), eines von der Fachhochschule Nordwestschweiz (FHNW), ein Modell von Microsoft sowie ein Modell von OpenAI namens Whisper.

Wir haben acht verschiedene Experimente mit einem Schweizerdeutsch-Korpus durchgeführt, der von der ZHAW und FHNW erstellt wurde. In den Experimenten wurde die Audiolänge auf verschiedene Weise verändert. Dabei stellten wir fest, dass die Modelle von der ZHAW, FHNW und von Microsoft dazu neigten, bei längerer Dauer schlechter abzuschneiden, während die Verlängerung der Dauer durch Hinzufügen von Stille keine Auswirkung auf die Leistung hatte. Die Wiedergabegeschwindigkeit zu verändern scheint bei den Modellen der ZHAW, von Microsoft und bei Whisper einen negativen Einfluss zu haben, sowohl beim Beschleunigen als auch beim Verlangsamen der Segmente. Das Modell der FHNW zeigte ein ausserordenliche Robustheit gegenüber einer Veränderung der Wiedergabegeschwindigkeit, so waren die Resultate beim Beschleunigen mit einem Faktor 1.25 nahezu identisch mit den Resultaten, bei denen die Wiedergabegeschwindigkeit nicht verändert wurde. Der grösste Einfluss auf die Leistung ergab sich, wenn mehr als ein Satz zu einem Segment hinzugefügt wurde. Ohne eine Segmentierung der Aufzeichnung schnitten die Modelle von ZHAW, FHNW und Microsoft schlecht ab, was darauf hinweist, dass eine Segmentierung eingeführt werden sollte, sobald mehr als ein Satz in einer Aufzeichnung erscheint. Das Training eines Modells speziell auf Mehrsatzsegmenten zeigte vielversprechende Ergebnisse für Einsatzsegmente, Mehrsatzsegmente und Szenarien, in denen Sätze beim Segmentieren der Tonaufzeichnungen aufgeteilt werden. Ein Vergleich zwischen einer satzbasierten Segmentierung, die als ideal betrachtet wird, und einer Fixed-Window-Segmentierung zeigte nahezu identische Ergebnisse.

Die Untersuchung der Modelle anhand einer realen Aufnahme zeigte, dass das ZHAW- und das Mehrsatzmodell deutlich schlechter abschneiden als die Modelle von FHNW, Microsoft und Whisper. Dies deutet darauf hin, dass weitere Untersuchungen erforderlich sind, um vollständig zu verstehen, was ein Speech-Translation Modelle in realen Szenarien gut funktionieren lässt.

Preface

This thesis has provided us with a valuable opportunity to delve into the field of speech translation for Swiss German. We were privileged to test remarkable models and witness their performance firsthand. This thesis has greatly contributed to our personal and professional growth. We eagerly anticipate the future of speech translation in Swiss German and the advancements it holds.

We extend our heartfelt gratitude to our supervisors, Prof. Dr. Mark Cieliebak and Dr. Jan Deriu, for their support and insightful weekly discussions throughout the duration of this thesis. Additionally, we would like to express our appreciation to the InIT for providing the necessary infrastructure to conduct our experiments. Special thanks are also due to Yanick Schraner from the FHNW for granting us access to their API, allowing us to perform experiments with the FHNW model. Lastly, we would like to thank Ellen Saaro and Marcel Saaro for their time, proofreading, and invaluable advice on the structure of this thesis.

Contents

Chapter 1

Introduction

The demand for audio transcription arises in various scenarios, such as recording meeting notes, generating subtitles, or aiding individuals with hearing impairments to participate in conversations or lectures. Initially, the focus in speech processing primarily revolved around widely used languages like English or Spanish. However, a recent shift towards new architectures, like transformers, has enabled training on a significantly larger number of languages, even those without a lot of labeled data or without a written form. Notably, Whisper, a language model developed by OpenAI, incorporates 98 languages during its pre-training phase [1]. Microsoft's speech-to-text technology supports an impressive 75 languages encompassing different dialects, totaling support for 487 locales [2]. Significant advancements have been made in the field of speech transcription, including the improvement of transcribing Swiss-German.

According to [3] Swiss-German is not dividable into a few dialects, instead it is considered a dialect continuum. While Swiss-German can be used in written form, there does not exist a set of rules like in standard German. Thus when transcribing Swiss-German it needs to be translated to standard German in the process. The limited availability of annotated Swiss-German data, combined with its comparatively low linguistic significance and the absence of a large speaker population proficient in Swiss-German, classify the language as a low-resource one within the context of speech processing according to [4]. Transcribing low-resource languages successfully can be done using frameworks, such as wav2vec 2.0, which is pre-trained on a large amount of data and can then be fine tuned on a specific language like Swiss-German [5]. There exist a handful of models that can transcribe Swiss-German to standard German, like the ZHAW (lowercase) model described in Chapter 3.1.1 or the Whisper model developed by OpenAI (Chapter 3.1.5).

This thesis conducts different experiments on the existing models to find out what the influence of audio duration is on the performance of these speech translation models. The thesis intends to draw conclusions regarding effective training strategies that can enhance the overall performance of these models.

1.1 Literature Review

Natural Language Processing (NLP) research exists in the context of AI, which emerged from Linguistics and aims to empower computers with text, speech, and language comprehension. Within NLP, speech-to-text and speech translation systems transform spoken language into written text. The field has in recent years experienced a transition from statistical techniques to neural networks, which has greatly improved the applications and research on low-resource languages, by lowering the amount of labeled data needed to create automatic speech recognition (ASR) models.

[6] focused on transcribing over 200 languages with safety and high quality as their main objective. They developed a model based on sparsely gated mixture of experts, incorporating novel data mining techniques specifically designed for low-resource languages. They proposed human translated datasets such as FLORES-200 and NLLB-SEED, the later provides bitext for 43 languages. The results demonstrated a significant 44% BLEU improvement compared to previous state-of-the-art translation systems. The social impact of this work is substantial, benefiting various stakeholders, including low-resource language communities by enabling access to content outside of their language.

In their work on multilingual language models for automatic speech recognition, [7] address the challenges of creating separate models for low-resource languages, which include limited training data, high computing costs during training, and inefficiency in hosting due to sparse traffic. Instead, they propose a general and scalable approach based on transformers that surpasses traditional multilingual transformer models. Their method involves developing large-scale locale-group transformer neural language models (NNLMs) to support ASR in low-resource languages. Furthermore, they demonstrate that fine-tuning these models with local-grouping techniques enhances the creation of improved monolingual models for low-resource languages.

In their study on speech translation, [8] explore various segmentation strategies for audio streams to achieve a balance between accuracy and latency. These segmentation approaches include segmenting based on word numbers (such as the win4 four-word window), comma-separated chunks, and conjunction-word based segments. The authors found that techniques like vocal tract length normalization (VTLN) and constrained model adaptation (CMA) were effective in improving accuracy, and retaining partial translations also contributed to increased accuracy. They introduced a novel segmentation method using conjunction-separated sentence chunks, which yielded high accuracy and low latency comparable to comma-separated sentence chunks. The study also revealed that noise in sentence and punctuation detection did not significantly impact the results. Additionally, synchronizing buffers across different pipeline components was highlighted as a requirement for efficient processing.

[9] conducted a study investigating various audio segmentation approaches for continuous audio transcription. They compared human annotated segmentation (gold) with two segmentation methods: (1) SHAS proposed by [10] and (2) a fixed-window segmentation with and without overlap between the windows. The study concluded that combining fixed-window segmentation with an overlap, a method for merging segments, and a context-aware ST model yielded promising results.

In their work on voice activation detection and audio segmentation, [11] highlight the challenge of mismatch between training data and real-world runtime environments for speech models. While systems are typically trained on manually segmented corpora, they are often presented with longer audio in practical scenarios that require segmentation. The study compares voice activation detection (VADs) to fixed-length and hybrid segmentation methods, proposing a hybrid solution that achieves improved results without compromising latency. The research emphasizes that the length of audio segments plays a crucial role in obtaining accurate outcomes. The effectiveness of segmentation approaches depends on the structural similarity between source and target languages, and segments should neither exceed the maximum length of training samples nor be excessively short. The proposed hybrid methods consider audio content and target segment length, with a focus on the latter rather than relying solely on detecting pauses, similar to the approach by [12]. The authors suggest splitting at the longest pause within the interval or at the maximum length as two variations of their proposed hybrid methods.

1.2 Outline

The thesis begins by introducing the motivation and objectives of the study in Chapter 1. This also includes a literature review to highlight related works. In Chapter 2 a comprehensive theoretical background is summarized, covering relevant concepts from previous research related to speech translations. In Chapter 3 and 4, the tools and data used to conduct the experiments are introduced. Chapter 5 outlines the pipeline utilized to conduct the experiments, explaining the experimental setup as well as each step in the speech translation pipeline. Chapter 6 is dedicated to the experiments that were conducted. Each experiment contains an introduction with a hypothesis, a section that explains the methodology that is used to conduct the experiment, a result section as well as a discussion section in which the results are discussed. Finally, Chapter 7 combines and summarizes the key findings from the experiments and gives an outlook of future research that could be conducted to further understand speech translation models.

Chapter 2

Theory

This chapter aims to introduce the theoretical foundations that are relevant to this thesis. The explanations provided should increase the comprehensibility of the subsequent chapters and lead to a better understanding for the reader.

2.1 N-Gram

N-Grams are a concept in computational linguistics to describe a sequence of n items in a sequence. These items can consist of letters, syllables, or words. They provide a standardized way of modeling text structures that can then be used to create statistical and machine learning based models.

It is worth noting how certain n-grams are given special names (Table 2.1) such as "unigram" when n equals 1. For n > 3 n-grams are commonly referred to with the name of the number n followed by "-gram", for example, "seven-gram".

N	**Title**	**Example**
1	unigram	Hello
2	bigram	Hello everyone
3	trigram	Hello my frined
4	four-gram	How is everyone doing
5	five-gram	I am doing great thanks

Table 2.1: Title and examples for n-grams with n from one to five.

Over the years n-grams have been utilized for many different use cases. In 1994 an n-gram based approach to categorize large amounts of text based documents [13]. By calculating the n-gram probabilities of the training data, profiles for each classification were created and used to predict the classification of test articles. A similar approach was used by [14] to create profiles for different composers. By measuring the probability of each possible n-gram of n chords appearing in a sequence, profiles for each composer were created and later used to identify a composition's style.

Sentiment analysis is a part of computational linguistics that involves determining the sentiment or emotional tone expressed in a piece of text. It can be used to identify whether a text expresses positive, negative, or neutral sentiment, and often involves techniques such as machine learning and natural language processing to classify and analyze text data. There exist freely available datasets, such as SENTIWORDNET 3.0 [17], that have collected sentiment scores for a large number of words. [18] proposes an automatic procedure for creating a general-purpose n-gram lexicon from an existing unigram lexicon. This new n-gram lexicon would ascribe sentiment score to n-grams for n > 1. With this new approach, the authors were able to improve the performance of statistical sentiment analysis methods.

2.2 Transformer

Transformer based models replace previous recurrent neural networks (RNN) and convolutional neural networks (CNN) for processing sequential data since the introduction by [15] in 2017. This architecture is based on the attention mechanisms that previous best performing RNNs used but with a simplified network architecture (Figure 2.1) that is also more parallelizable and requires significantly less time to train.

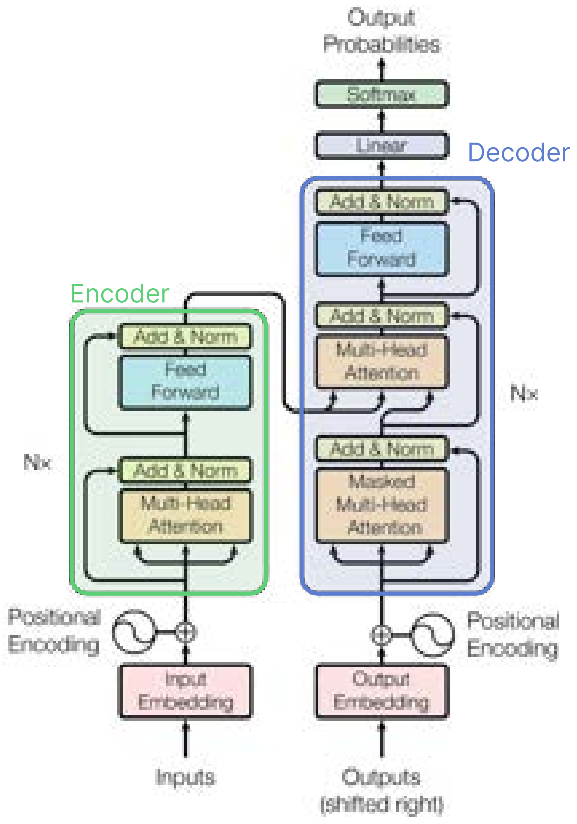

Figure 2.1: Illustration of the transformer architecture as described by [15] with additional highlighting of the encoder and decoder.

Nowadays transformers are widely used in language models such as BERT introduced by [16]. BERT stands for Bidirectional Encoder Representations from Transformers and is designed to pre-train deep bidirectional representations from unlabeled data. It can be used in a wide range of language understanding tasks, such as chatbot systems or summarization tasks, and can be easily fine-tuned to solve these tasks.

2.3 Language Model

Language modelling is one of the major approaches advancing language intelligence. It aims to mimic the probability distribution of words found in human-written text using statistical and machine learning algorithms. Over time, language models have gone through different stages of development [19].

As early as 1998, statistical language models emerged, utilizing statistical learning techniques to analyze and predict word probabilities within text [20]. This approach laid the foundation for further advancements in language modelling. Following statistical models were neural language models, employing the power of neural networks to achieve more accurate language generation and prediction [21]. Another significant breakthrough came with the introduction of pre-trained language models [22]. These models employed large, unlabeled datasets in a pre-training stage to learn general language representations. Following pre-training, the models underwent fine-tuning on specific tasks, allowing them to excel in a wide range of natural language processing tasks. The latest milestone in language modelling is the development of large language models. These models incorporate extensive computational resources and vast amounts of training data to achieve unprecedented language understanding and generation capabilities. Notable examples include OpenAi's GPT-3 and GPT-4 [23], Google's BERT [16], and Meta AI's LLaMA [24].

In [25] the relevant use cases of natural language are listed as speech transcription, speech translation, speech synthesis, question answering and information retrieval as well as chatbots and dialogue systems. While all these topics are certainly interesting and worth pursuing, only the relevant topics are described in more detail.

2.3.1 Speech Transcription

Speech transcription, also known as speech-to-text (STT), is the process of converting spoken words into written text and is used in a variety of different use cases. Speech transcription can be a useful tool for people with cognitive or physical impairments. For these people, it may be advantageous to have a written transcript of something that was spoken. Additionally, speech transcription is used to automatically generate subtitles for audio and video content. This can help people with hearing impairments and lessen the manual labour of transcribing audio by hand.

In [26] the authors reviewed literature on the impact of speech transcription on students and their performance. It was found, that text transcriptions were able to help students with hearing impairments focus on learning and taking notes and kept them from having to focus on reading lips or watching a sign language interpreter. The authors also analysed papers testing the use of speech-to-text for students during online classes. In the experiments, students were able to simultaneously hear the teacher and read a real time transcript of what was spoken. The test showed how using the transcript of the lecture as a learning tool was able to significantly improve the performance of the students using it.

Speech transcription, however, is not without its challenges. One prevalent issue is transcribing long audio recordings. Language models are typically trained on short audio samples due to the computational expense of training on larger input samples. This stems from using matrices to represent user input data. As of October 2022, even the most advanced algorithms for speech transcription still exhibit a time complexity of $O(n^{2.37188})$ [27], emphasizing the computational challenge associated with training on larger input samples due to the use of matrices. For that reason, longer audio recordings need to be transcribed in segments, and the individual transcriptions then have to be properly merged together. Another challenge is the complex nature of real-life audio. Spontaneous speech can be inconsistent, hectic and overlapping with the speech of other participants. All these issues, combined with poor audio quality, can make an audio recording difficult to accurately transcribe.

2.3.2 Speech Translation

When translating text from one language to another, language models are already widely used. These tools make it possible for people who do not comprehend the language in which a text is written to access it easily. Using Google Translate alone, 143 billion words are translated every day [28]. Additionally, language models can aid the conventional translation process by generating a first rough translation on which experts can further build upon.

Speech translation (ST) is the process of taking spoken language of one language and translating it to a different language.

This thesis focuses on the speech translation of Swiss-German audio to German text. Speech translation is used in international communication as it allows people to freely communicate even if they do not share a common language. Tourists visiting a foreign country can have conversations with local residents and business meetings can be held without the need of a translator being present.

According to the Swiss national Bureau of Statistics, more than 60% of all Swiss people speak the German dialect Swiss-German [29]. There exists however no standardised written form of Swiss-German and spoken word is notated in German as German is one of the four official languages [29]. This makes tools transcribing Swiss-German audio to German text especially useful. Recapp.ch, a commercially available tool to create those kinds of transcripts, is already being used by multiple institutions of the Swiss government [30].

2.4 Beam Search Decoding

Beam search is one of many different search algorithms used to find the most likely output sequence given a collection of probabilities for different items. When choosing a search algorithm the pros and cons of each one need to be evaluated. While there are algorithms that always generate the most likely sequence of text, this certainty comes at a massive cost of computing time. Beam search tries to strike a balance between pursuing every possible combination of words and not considering too few options [31]. The earliest mention of beam search was in 1977 [32] and has since then been reintroduced for sequence-to-sequence [33] and neural network translation [34] models.

Beam search maintains a set of the most promising output sequences, known as the beam, and explores different possible continuations of each sequence by considering multiple candidates at each step. It gradually prunes the less promising candidates based on a scoring function until the final output sequence is determined.

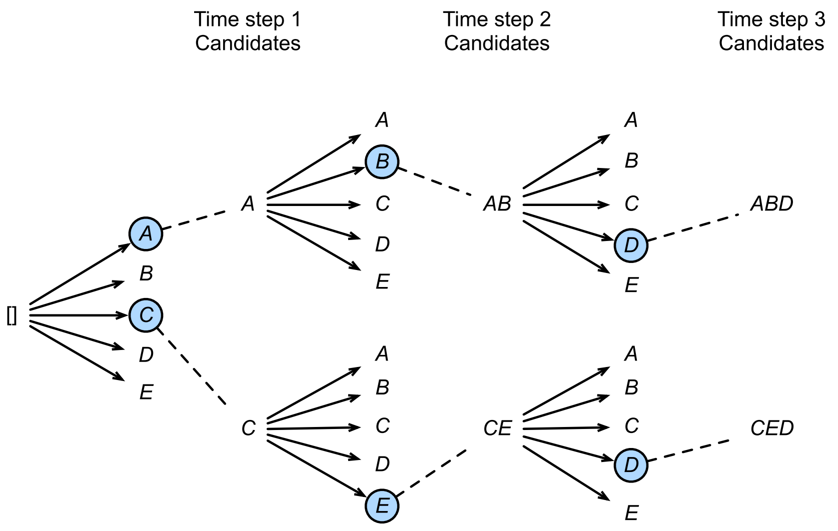

Figure 2.2: Illustration of beam search using two beams to find the most probable combination of characters from [31].

In the book [31], the example of Figure 2.2 is given. In it, the beam search algorithm is performed with *beamsize* $k = 2$. At each time step, the next elements with the highest probability given the previous elements of the beam are added. From the possible elements, A, B, C, D, E, it is A and C that have the highest probability at the start of the two beams at time step 1. Next, B has the highest probability given A as the previous element, and E is selected given C. At time step 2, the two beams are now AB and CE. The same step is repeated to get the final beam sequences of ABD and CED. From these two, the one with the higher overall probability can now be picked.

2.5 Connectionist Temporal Classification

When training language models to predict sequences, a problem arises. Generating proper alignment of labels on the training data. Connectionist Temporal Classification or CTC for short, is a technique that allows recurrent neural networks to be used for training sequence-to-sequence models without requiring aligned input-output pairs [35].

[36] provides a formal definition of the alignment process implemented by CTC, which is described in the next section.

Let $X = [x_1, x_2, x_3, ..., x_T]$ be the input sequence which in the case of audio transcription would be the raw waveforms. Let $Y = [y_1, y_2, y_3, ..., y_U]$ be the resulting output sequence, in this case, the resulting transcript. Note that T can be different from U and so the two sequences can be of varying lengths. A language model can then be trained using a loss function p, this loss function should take in X and Y and return the probability that Y is the corresponding transcript given the sequence X, $p(Y|X)$. To use gradient descent while training a model, this loss function must be differentiable. To then use the model to evaluate a new input sequence X and receive transcript Y^*, another function needs to be defined, $Y^* = \underset{Y}{argmax}\ p(Y|X)$. CTC provides cost-efficient solutions for both of these functions.

When given an input sequence X originating from an audio signal and aligning it with an output sequence Y there are a few steps best explained in this example, also shown in Figure 2.3.

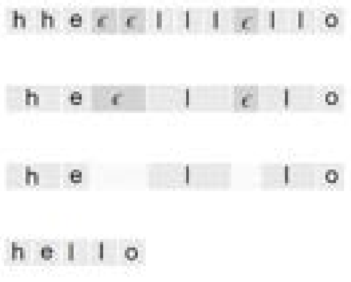

Figure 2.3: CTC alignment process from longer input sequence X to shorter output sequence Y [36].

Given the input sequence $X = [h, h, e, \epsilon, \epsilon, l, l, l, \epsilon, l, l, o]$, ϵ representing no audible letter, the first step is to merge repeating characters resulting in the sequence $[h, e, \epsilon, l, \epsilon, l, o]$. In the next step, any ϵ items are removed, resulting in the expected sequence $[h, e, l, l, o] = Y$.

2.6 wav2vec 2.0

Vectorization is a fundamental process, particularly in the field of artificial intelligence (AI) and data analysis. It is the process of converting data from its initial form into numerical vectors in a latent space, which can then be effectively utilized by a wide range of algorithms. [37] introduced a framework in 2019 in which during an unsupervised pre-training the hidden representations are learned without the need for transcribed audio. It is then fine-tuned on a smaller set of transcribed audio. This addresses the limitation of supervised learning for ASR systems, where a large amount of labeled data is required. While this amount of data exists for common languages, like English or Spanish, it might not be available for languages such as Swiss-German. In 2020, [37] introduced an updated version, called wav2vec 2.0. Which outperforms the previous state-of-the-art semi-supervised methods while offering a conceptually simpler approach. During both unsupervised training and fine-tuning, spans of the latent speech representation are masked and the model is evaluated based on its ability to predict the hidden information. CTC (Chapter 2.5) is used to calculate the loss during training.

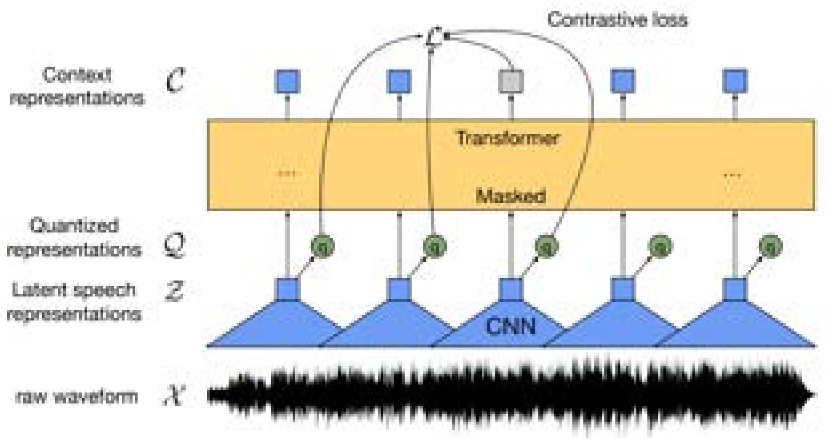
Contrastive loss
$\mathcal{L}$
Context representations $\mathcal{C}$
Transformer
Masked
Quantized representations $\mathcal{Q}$
Latent speech representations $\mathcal{Z}$
CNN
raw waveform $\mathcal{X}$

Figure 2.4: Illustrating the wav2vec 2.0 framework showing the process of creating a context representation from the raw waveform using a convolutional neural network [37].

Figure 2.4 shows, the raw audio stream X is initially fed into a multi-layer convolutional feature encoder, which produces latent speech representations Z. A transformer then takes the latent speech Z as input and generates context representations C, capturing the context of the input audio.

2.7 Measuring Performance

The transcript, which is created by an ST model, is used to evaluate the performance of a model. Evaluating the performance is necessary to compare different models and draw a conclusion from them. The performance is defined in this thesis as the quality of the translated transcript. It does not include factors such as inference time or CPU utilization.

To measure a transcript produced by an ASR system the word error rate (WER) is commonly used according to [38]. To measure a translation produced by an ST system other metrics can be used such as the BLEU or ROUGE score.

Depending on what is important to the scenario a different score can be more important, for example when creating subtitles the order of the word translated must stay the same. While in meetings notes the meaning of the words is more important. It is important to note that each performance metric has different strengths and weaknesses and to examine the performance of a model it is crucial to compare different metrics and decide depending on the goal which metric is most important.

2.7.1 WER

According to [38] WER or word error rate is a simple metric by which transcripts of ASR systems can be assessed. The WER describes the Levenshtein distance on a word, instead of a character level. To calculate the WER, a few variables need to be known.

- S: the number of substitutions
- D: the number of deletions
- I: the number of insertions,
- C: the number of correct words,

The WER is then calculated as follows.

$$WER = \frac{S+D+I}{S+D+C}$$

The WER is simple to calculate and understand, however, its simplicity is also limiting in the value it provides when evaluating transcriptions. The biggest drawback is how the meaning of the transcription is not taken into account. A model might transcribe a sentence perfectly but use a synonym in the assumed truth or restructures the sentence. In both of these cases, the WER increases significantly. The score is between 0 and 1, 0 meaning no errors, and 1 meaning all words are incorrect.

It is important to note that if a transcript is very short, a single mistake can have a bigger impact on the WER, than on a longer transcript.

2.7.2 BLEU

BLEU or bilingual evaluation understudy, was first described by [39] in 2002. The goal was to create a new metric for scoring machine generated translations that more closely resembles human evaluations. BLEU has since become one of the most popular tools to evaluate translations. It was even awarded the "Test-of-Time Paper" award by the North American Association for Computational Linguistics in 2018 [40]. At the core of the BLEU score is the idea of n-gram precision. For each possible n-gram in the generated text, the number of occurrences in the reference is counted and adjusted accordingly. After combining the precision value a score between 0 and 1 that is commonly also scaled to values between 0 and 100.

Although the BLEU score can theoretically reach up to 100, this score is necessary for a translation to be "good". Table 2.2 gives an overview of the meaning of different BLEU scores according to [41].

BLUE Score	**Interpretation**
< 10	almost useless
10-19	hard to understand underlying meaning
20-29	underlying meaning is understandable, with grammatical errors
30-40	understandable to good translation
40.50	high quality translation
50-60	very high quality, adequate, and fluent translations
> 60	quality often better than human translations

Table 2.2: Interpretation of BLEU scores by [41].

BLEU's popularity makes it a must have metric when evaluating and comparing ST models. It is however not exempt from drawbacks. When matching n-grams to calculate precision, exact matches are expected and the severity of the miss translations is ignored. A model translating the word *uncharacteristic* to *uncharacteristically* would be scored the same as a model incorrectly translating *uncharacteristic* to *pineapple* as both translations are not perfect. Additionally, translations that contain either too many or too few words are punished with a lower score even if they are still clear and understandable to humans. Furthermore, the underlying context of a translation is ignored when evaluating the BLEU score. A translation could be almost perfect, containing the intended meaning accurately, but still receive a low score simply because it does not use the same words and sentence structure as the reference text. This limitation fails to acknowledge that language is a dynamic and flexible system, and different linguistic expressions can effectively convey the same message.

In 2018, [42] addressed one of the problems with the BLEU score. The BLEU score is a parameterized metric, which means that it can vary a lot depending on the parameters used during the evaluation. [42] introduced a unified way to evaluate BLEU scores, making the BLEU score comparable across different projects.

2.7.3 ROUGE

ROUGE or Recall-Oriented Understudy for Gisting Evaluation is a metric similar to BLEU (Chapter 2.7.2). ROUGE was introduced by [43] in 2004 and has since gained popularity in the field of natural language processing and text summarizing. It addresses the limitations of previous evaluation methods by focusing on the recall of important information rather than the precision of n-gram matches.

The core idea behind ROUGE is to calculate the recall of n-gram units between the generated summary and the reference summaries. The recall is then used to compute the ROUGE score, which represents the quality of the generated summary. ROUGE evaluates the quality of summaries based on how well they capture the important content from the reference summaries.

Like any metric, ROUGE has its weaknesses. One limitation is that it solely focuses on the recall of n-grams, which means that it does not capture the fluency and coherence of the generated summaries. A summary could achieve a high ROUGE score by simply copying phrases from the reference summaries without producing a coherent and meaningful summary. Additionally ROUGE, just like BLUE and WER, relies on the generated text being similar to the provided reference. The generated text might convey the correct information with words differing from the reference material and therefore receive a low score.

2.7.4 SemDist

Semantic distance (SemDist) measures how similar two transcripts are to one another. According to [44] the semantic similarity is defined as the cosine coefficient between two vectors. Those vectors are the representations of the reference and hypothesis of an ST model. The distance is calculated defined as:

$$\text{SemDist(A,B)} = 1 - \frac{A \cdot B}{\|A\|\|B\|}$$

Where A is the reference and B the hypothesis created by the ST model. The SemDist is a value between 0 and 1. 0 implies that two transcripts are identical in meaning and 1 means that the vector representations are orthogonal.

The SemDist addresses the issues with other performance metrics where the meaning of a transcript is not taken into account. With this metric only the meaning is relevant and not the sentence structure or the words used. This is useful when a transcript does not have to be word by word, for example when transcribing a meeting.

Chapter 3

Tools

To allow anyone to recreate the experiments (Chapter 6) conducted in this thesis more easily, this chapter describes the most important tools, how and why they were used.

3.1 Language Models

At the core of every experiment (Chapter 6) conducted in this thesis lays a language model (Chapter 2.3) used to transcribe some given input. This chapter serves as an introduction to all the language models used, providing an overview of how they were created, their key features and their capabilities.

3.1.1 ZHAW Model (lowercase)

The ZHAW (lowercase) model is a wav2vec 2.0 (Chapter 2.6) based language model created by the Center for Artificial Intelligence at the Zurich School of Applied Sciences (ZHAW) during the writing of this thesis. It has not yet been officially published. The model was trained with single sentence segments from the STT4SG training dataset 4.2. All input characters were converted to lowercase, and special characters such as dots and commas were removed to address issues observed in previous models, such as struggles with capitalization of output words and the inclusion of incorrect special characters.

3.1.2 ZHAW Model (multi sentence)

The ZHAW (multi-sentence) model shares the same architecture as the ZHAW (lowercase) model (Chapter 3.1.1), both developed by the same researchers. A previously created model trained on single sentences without removing or altering any special and upper case characters was used as a base to further train the ZHAW (multi-sentence) model. However, instead of training on individual sentences, the ZHAW (multi-sentence) model was trained on sequences of one to three sentences. The purpose of this approach was to improve the model's performance on sentence boundaries and inputs containing multiple sentences.

For the training of both the base model and the multi-sentence training, the STT4SG test data (Chapter 4.2) was employed. The number of parameters remained largely unchanged from the ZHAW (lowercase) model (Chapter 3.1.1). Nevertheless, the training process incurred higher computational and time costs, as explained in (Chapter: 2.3.1).

3.1.3 FHNW

In 2022 the University of Applied Sciences and Arts Northwestern Switzerland (FHNW) introduced new language models capable of transcribing Swiss-German audio to German text [45]. One of the authors of the aforementioned study, Yanick Schraner from FHNW, graciously provided a demo API for this research thesis's experiments.

According to correspondence with Yanick Schraner, the demo API employs a variant of the XLS-R 1B model described in [45], which has not yet been published. The language model used in the API is based on FHNW Transformer [45] and incorporates wav2vec 2.0 (Chapter 2.6) with sequence-to-sequence decoding, differing from the CTC decoding (Chapter 2.5) utilized by the models described in [45]. In addition to the training data used for the previous models, namely the STT4SG-350 corpus [46] and the Swiss Parliament Corpus [47], the model was further trained using the common_voice_11 corpus [48]. The API provides two modes of operation. The first mode, referred to as FHNW (short), directly transcribes audio samples that are less than 20 seconds in length. For audio files longer than 20 seconds, the second mode, known as FHNW (long), must be used. FHNW (long) internally segments the input audio, transcribes the segments separately, and returns the merged transcripts.

3.1.4 Microsoft

As part of the Microsoft Azure's Speech Service line of products, they offer a service to transcribe audio[1]. The input audio may include words spoken in a different language than the resulting transcript. In a blog post by Microsoft [49] it is described, how a transformer based architecture has been to translate over 100 languages as of October 2021. One of the supported languages [50] is Swiss-German, This makes Microsoft's translator a fitting test subject for this thesis. One of the downsides of using the commercial product provided by Microsoft is the lack of public information about detailed architecture, pre- and postprocessing and the training data, used to train the model. It has to be assumed that Microsoft did not use any of the test data, used in the experiments described in this thesis.

Similar to the FHNW model (Chapter 3.1.3), the Microsoft API provides two possible methods of transcribing a piece of audio. Recordings with a duration of under 30 seconds [51] can be directly transcribed in one step. For inputs exceeding the 30 second limit, Microsoft internally splits the audio into segments, transcribes them separately and returns the merged transcript. The details of these steps are not publicly available.

[1]Available under: `https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service`

3.1.5 Whisper

At the end of 2022, OpenAI introduced their new language model called Whisper [1]. Its primary function was to approach human level robustness and accuracy on English speech recognition. It is based on an encoder-decoder Transformer architecture with the large-v2 version utilized in this paper boasting a staggering 1550 Million parameters. To train the model, a vast amount of data consisting of 680,000 hours was used, with a specific preprocessing step of segmenting the test data into 30 second segments. By employing large-scale weak supervision techniques, the researchers successfully developed a model that outperforms previous fully supervised models on various standard benchmarks. The training process encompassed four scenarios. Firstly, the primary function of English to English transcription, where the model was to predict the English transcript of audio containing spoken word English. Secondly, any-to-English transcription, where the input audio contains any of the 96 other languages included in the training data, and the model is made to predict the correct English transcript. Thirdly, non-English transcription, transcribing audio containing a non-English language and transcribing it to text of that language. And lastly, the model was trained to generate no transcript in cases where there were no audible spoken words in the provided input.

Notably, the training data for multilingual speech recognition, made up of 117'113 hours of non-English audio, included 13,344 hours of German, but reportedly no Swiss-German. While this means the model was not specifically trained to transcribe Swiss-German audio to German text, initial tests have demonstrated its capability to produce usable results, warranting its inclusion in this thesis.

3.1.6 SemDist Model

To calculate the SemDist between two sentences a sentence-transformers model was used [52]. Bidirectional Encoder Representations from Transformers or BERT for short, has been a successful language representation model used for improving many natural language processing tasks [16]. From BERT, Sentence-BERT or SBERT was developed in August of 2019 [52]. SBERT uses siamese and triplet network structures to create better sentence embeddings. These embeddings can be compared using cosine-similarity to calculate a Semantic Similarity Score or Semantic Distance (Chapter 2.7.4).

3.2 Python Libraries

To avoid compatibility errors when recreating the experiments described in this thesis, this chapter includes short descriptions and the use cases for important Python packages utilized (Table 3.1).

Package	**Version**	**Use Case**
numpy	1.22.1	Mathematical calculations.
sentence-transformers	2.2.2	Run SemDist language model (Chapter 3.1.6).
openai-whisper	20230314	Run whisper model (Chapter 3.1.5).
librosa	0.9.2	Analysing audio files.
screbleu	2.0.0	Calculate BLEU scores (Chapter 2.7.2).
rouge-score	0.1.2	Calculate ROUGE scores (Chapter 2.7.3).
jiwer	2.3.0	Calculate WER of samples (Chapter 2.7.1).
pydub	0.25.1	Cutting, merging and manipulating audio files.
pyctcdecode	0.3.0	CTC decoding (Chapter 2.5).
Levenshtein	0.20.9	Calculating Levenshtein distances (Chapter 5.3).
kelm	github.com[2]	Manage CTC language model.
azure.cognitiveservices.speech	1.26.0	SDK for Microsoft azure API (Chapter 3.1.4).
supabase	2.2.2	Database to store results.

Table 3.1: Python packages used, their versions and use cases.

[2]Available under: `https://github.com/kpu/kenlm/archive/master.zip`

Chapter 4

Data

This chapter gives an overview of the corpora that were used for the experiments in Chapter 6. It describes what each dataset contains and how it was collected.

4.1 Manual Transcription

To compare ST models in a real-life scenario, one audio sample was manually transcribed (Appendix A.1). The audio used was a weather report produced by TeleZüri and uploaded to their YouTube Channel[1]. The report is a two minute recording from 2014 spoken by a single person, the former Miss Schweiz Dominique Rinderknecht. Her dialect is from the eastern region of Switzerland and the spoken text is most likely scripted. Each word is clearly pronounced and there are no background noises present. The advertisements present at the start and end of the uploaded video were removed manually.

[1]Available under: `https://www.youtube.com/watch?v=51ieImUZaco`

4.2 STT4SG-350

The STT4SG-350 (Speech-to-Text for Swiss German) corpus was collected in 2022 by [46] and contains 343 hours of Swiss-German audio segment, each annotated and labeled with dialect, canton, zip code, age and gender of the speaker.

The data was obtained with the help of a web app where Swiss-German speakers were shown a German sentence, asked to record and translate it to Swiss-German.

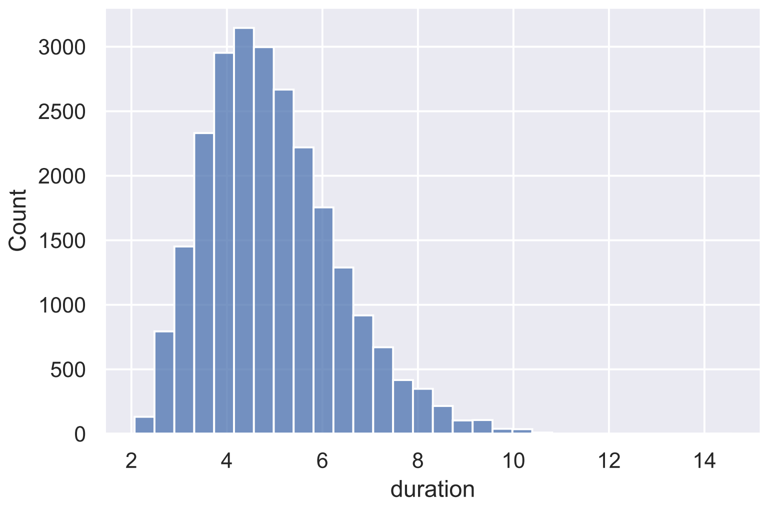

Figure 4.1: Distribution of the audio length within the STT4SG-350 test corpus.

The test set of the corpus consists of 35 hours in 7 different dialects. Figure 4.1 shows that each audio sample is between 2 and 15 seconds with most samples being around 4 to 6 seconds in length.

Chapter 5

Experiment Setup

This chapter describes the process used to prepare run and analyze the experiments conducted. This should help the reader better understand the process used to generate the results and make it easier to reproduce the results described in this thesis.

5.1 Definitions

To properly describe the experiment setup the following terms need to be defined:

- **input files**

  Input files are all the files that are given as the input of the experiment. *Input files* can be the audio files that are to be transcribed, the text files containing the reference transcription or JSON files containing metadata about the audio file. This metadata could be any information used to test a hypothesis. For example, if the impact of the gender of the speaker was to be tested, this information would be included in the metadata file. The *input files* are linked by having the same file name excluding the file type.

- **run**

  A *run* is the process of running one or more sets of input files through the experiment pipeline. A *run* is always part of an experiment.

- **experiment**

  An *experiment* is a collection of one or more runs that aim to evaluate a hypothesis.

- **segment**

  In the case that an input audio file cannot be evaluated by the selected model in one piece, it needs to be cut into smaller pieces. These pieces are called *segments.* If the input audio does not need to be cut, it becomes a single *segment.*

- **segmentation**

  The definition, of how an input audio file is to be cut into segments is saved in a *segmentation* file. Each input file will have a corresponding *segmentation*.

5.2 Pipeline Steps

To run an experiment, the pipeline steps shown in Figure 5.1, are completed.

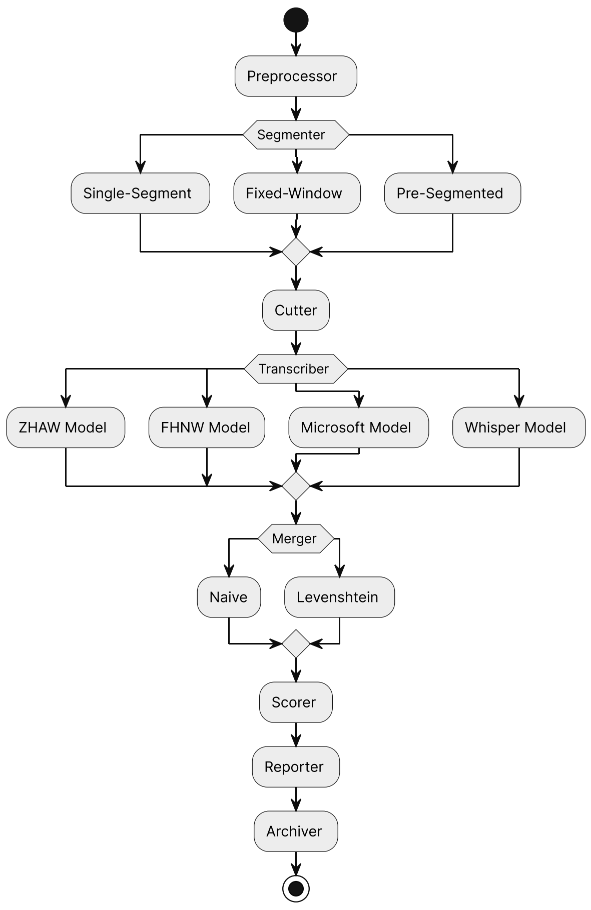

Figure 5.1: Illustration of the pipeline including the individual steps and their different implementations.

1. **Preprocessor**

   The preprocessor's job is to make sure the audio input files that are used as the input to the experiment are in the correct form to be transcribed by the model used. Preprocessing includes changing the audio format and setting the frame rate to the required 16'000 Hz.

2. **Segmenter**

   Some ST models only transcribe audio that is up to a certain length. If an input audio file is longer than this maximum length, it has to be cut into segments, transcribed in parts and merged again. It is the segmenters job to decide how each audio file is to be cut. In the pipeline, three different options exist to do this.

   (a) **Single Segment** forces one segment even if the length exceeds the maximum length that the ST model can transcribe.

   (b) **Fixed-Window** segments the audio into fixed length sizes with a defined overlap. This overlap can also be 0 seconds.

   (c) **Pre-Segmented** offers the option to manually segment the audio before and include a file in the pipeline, input that defines the segmentation already.

   For each audio input file, the segmenter creates a corresponding JSON file including an array of all the details for every segment of that file.

```
{
  "index": 0,
  "start_time_in_s": 0,
  "duration_in_s": 0,
  "input_file": "<path to the audio file>"
}
```

3. **Cutter**

   The cutter cuts the input audio file into the segments defined by the segmenter. The input audio files are copied, cut into the defined segments and stored in a dedicated folder.

4. **Transcriber**

   An ST model creates a transcript for each audio segment. These transcriptions are grouped for each original audio input file and saved into a JSON file. The index matches the index in the JSON object generated by the segmenter for the respective audio segment. The output of the transcriber looks like this:

```
[
  {
    "index": 0,
    "text": "<transcript>"
  }
]
```

5. **Merger**

   If the input audio files are cut into segments and transcribed separately, the generated transcript for this one audio file needs to be merged again. Two merging algorithms are offered in this pipeline:

   (a) **Naive** concatenates each audio segment. This approach can achieve good results if the input audio file is cut at optimal places but might struggle with segments created with fixed sizes.

   (b) **Levenshtein** this approach is described in Chapter 5.3.

6. **Scorer**

   To evaluate the performance of the transcriber the generated transcriptions are scored using different metrics such as the BLEU, ROUGE, WER and SemDist. The scores are then collected in a JSON file.

   ```
   [
     {
       "type": "<name of the metric>",
       "value": 0,
       "file": "<path to the transcript>"
     }
   ]
   ```

7. **Reporter**

   To save the results of a pipeline run a reporter sends the results to a database that can be accessed from another device.

8. **Archiver**

   The archiver cleans up the files generated by the experiment and stores them locally in a separate folder.

5.3 Levenshtein Merger

In certain experiments, the input audio may be too long to be evaluated by a model as a whole. In that case, the audio is cut into smaller segments with a fixed size and a set amount of overlap between segments. Each segment is then evaluated separately, and the resulting transcripts need to be merged again. To create the possibility of merging two transcripts by considering the similarity between consecutive segments, this algorithm was developed.

To better explain the process, the following two hypothetical transcriptions were created:

1. *this is an algorithm based on the minimum average*
2. *based on the minimum average Levenshtein distance of two sentences*

It is easy to see what the merged sentence should be *this is an algorithm based on the minimum average Levenshtein distance of two sentences*. However, the automation of this process calls for a more quantifiable way of achieving this result.

The basic idea of this algorithm is to look at each possible way to overlap the two transcriptions and calculate a score for each possible number of words overlapped. This score should represent how similar the overlapped words are to each other. The overlap with the best score is most likely to be the actual overlap of the two transcriptions. With the best overlap known the two segments can then be combined into one.

In the field of linguistics and computer science, there are many different ways to calculate the similarity of two strings [53]. One particular metric is the Levenshtein distance [54]. The Levenshtein distance is the minimum number of edits needed to convert one string into the other or vice versa. An edit can be the deletion, insertion or substitution of one character in a string, each associated with a cost of 1. The combined cost of all edits needed represents the Levenshtein distance. The fewer edits are needed to change one string into the other, the more similar they are to each other.

As an example, here is the minimum number of edits needed to turn *nothing* into *something*.

1. substitute **n** with **s** → ***s****othing*
2. insert **m** → *so****m****thing*
3. insert **e** → *som****e****thing*

The Levenshtein distance is now calculated by adding the costs of all steps together.

$$distance = Cost_{substitution} + Cost_{insertion} + Cost_{insertion}$$

$$distance = 1 + 1 + 1 = 3$$

To use the similarity of two words to find the optimal overlapping of two sentences, every possible overlapping is analyzed. For each possibility, the Levenshtein distance of every word to its overlapping opposite is calculated and averaged. This average Levenshtein distance is used as a score of how good this overlapping is. An example overlapping of three words is shown in Table 5.1.

Sentence 1	algorithm	based	on	the	minimum	average		
Sentence 2				based	on	the	minimum	average
Levenshtein Distance				4	6	6		
Average Distance					5.33			

Table 5.1: Calculating the average Levenshtein distance of two sentences with three words overlapping.

This calculation is then repeated for each possible number of overlapping words as shown in Table 5.2. Now it is clear to see how overlapping the two sentences with five words is the optimal solution just like predicted. This process still results in optimal overlap even if the two transcripts contain incorrectly transcribed words.

Overlap	Average Distance	this	in	an	algorithm	based	on	the	minimum	average
9	5.67	based	on	the	minimum	average	levenshtein	distance	of	two
8	6.25		based	on	the	minimum	average	levenshtein	distance	of
7	6.14			based	on	the	minimum	average	levenshtein	distance
6	6.33				based	on	the	minimum	average	levenshtein
5	0					based	on	the	minimum	average
4	5.5						based	on	the	minimum
3	5.33							based	on	the
2	7								based	on
1	6									based

Table 5.2: Calculation of the average Levenshtein distance for every possible number of words overlapping for two example sentences.

Assume the two sentences contain incorrectly transcribed words and are now as follows:

1. *this in a algorithm base on they minimum over*
2. *based of the min average levelstein difference on two senses*

The same process still concludes with the result that the overlap of five words is the optimal solution (Table 5.3). Now the minimum average distance is no longer 0 but the correct overlap is still the one with the lowest score.

Overlap	**Average Distance**	**this**	**in**	**a**	**algorithm**	**base**	**on**	**they**	**minimum**	**over**
9	5.67	based	of	the	min	average	levelstein	difference	on	two
8	5.75		based	of	the	min	average	levelstein	difference	on
7	5.71			based	of	the	min	average	levelstein	difference
6	5.83				based	of	the	min	average	levelstein
5	2.2					based	of	the	min	average
4	5						based	of	the	min
3	4.67							based	of	the
2	5								based	of
1	4									based

Table 5.3: Calculation the average Levenshtein distance for every possible number of words overlapping for two example sentences containing mistranscriptions.

The process of finding the optimal overlap of two transcripts can be formally defined as follows:

Let A be a series of words $[a_1, a_2, \dots, a_n]$ and B be a series of words $[b_1, b_2, \dots, b_m]$ representing two consecutive segmentation's with some overlap.

Let $lev(a, b)$ be a function that returns the Levenshtein distance between the words a and b.

Let $avg\_lev(A, B, v)$ be a function returning the average Levenshtein distance when overlapping v elements of A and B.

$$avg\_lev(A,B,v) = \begin{cases} \frac{1}{o}\sum_{i=1}^{v} lev(a_{n-v+i}, b_i), & \text{if } 1 \leq v \leq \min(n,m) \\ \text{undefined}, & \text{otherwise} \end{cases}$$

And finally, let $best\_overlap(A, B)$ be a function returning the number of overlapping words resulting in the lowest average Levenshtein distance for any A and B.

$$best\_overlap(A,B) = \underset{1 \leq v \leq \min(n,m)}{argmin} \; avg\_lev(A,B,v)$$

After the optimal overlap has been determined, the two sentences need to be merged into one text. This is achieved by looking at the overlapping words and using the first half of the first sentence and the second half of the second sentence (Table 5.4).

Words used						2		2						
Sentence 1	this	is	an	algorithm	based	**on**	**the**	minimum	average					
Sentence 2						on	the	**minimum**	**average**	levenshtein	distance	of	two	sentences
Merged	this	is	an	algorithm	based	on	the	minimum	average	levenshtein	distance	of	two	sentences

Table 5.4: Merging two sentences with an even number of words overlapping.

In case the number of overlapping words is uneven, one more word is taken from the first sentence (Table 5.5).

Words used						3			2					
Sentence 1	this	is	an	algorithm	**based**	**on**	**the**	minimum	average					
Sentence 2					based	on	the	**minimum**	**average**	levenshtein	distance	of	two	sentences
Merged	this	is	an	algorithm	based	on	the	minimum	average	levenshtein	distance	of	two	sentences

Table 5.5: Merging two sentences with an odd number of words overlapping.

When cutting audio files into segments with a set size, the resulting audio may include partial words at the start and end of the segment. These partial words have a low likelihood of being correctly transcribed at the end and start of two consecutive segments. To avoid these like mistranscriptions from influencing the average Levenshtein distance a set number of words are removed at the start and end of each segment before calculating the optimal overlap, and are added back again before merging.

Chapter 6

Experiments

The experiments in this chapter investigate the influence of audio length on the performance of Swiss-German ST models. It starts by comparing the performance of different models on a set of test sentences with various lengths (Chapter 6.1). After that, the length of the audio segments in the test set is increased by adding some extra padding (Chapter 6.2). To further modify the audio length the segments are sped up and slowed down (Chapter 6.3). To extend the duration, multiple sentences are combined into one audio segment (Chapter 6.4) and then padding between two sentences is introduced (Chapter 6.5). With the findings from experiments, a new model is trained and tested on the experiments 1 to 4 (Chapter 6.6). To further compare the merging methods, which are required for long audio recordings, the ST models are subjected to an experiment in which the theoretical limit of the merging methods are compared (Chapter 6.7). Finally, the models are tested on a real-life audio recording which requires segmentation before transcribing and merging after it (Chapter 6.8).

6.1 VARLEN - Variable Length

This experiment aims to examine the relationship between the duration of an audio segment and the performance of individual models. There is reason to believe that the quality of transcripts tends to decrease as the duration of the audio segments increase. This can be explained by the distribution of the audio segment's duration within the corpora that were used to train some of the ST models that are under observation. This experiment is designed to gain further insights into the behavior of ST models across different audio segment durations.

6.1.1 Hypothesis

Speech translation models perform better on shorter audio segments.

6.1.2 Method

For this experiment, the test set of the STT4SG corpus (Chapter 4.2) was chosen because it has not been used during the training of any of the ST models. From the test set 10'000 audio segments of lengths between 1 and 15 seconds were sampled, translated and evaluated. Each audio segment contains exactly one sentence and no segmentation has been applied to the audio samples because all ST models under observation are developed to work with audio samples up to 15 seconds. This experiment is conducted on the ZHAW (lowercase), FHNW (short), Microsoft (short) and Whisper models. To evaluate the performance of the ST models the BLEU and ROUGE scores were calculated as well as the WER and SemDist.

6.1.3 Results

For each ST model, the performance metrics are plotted using four scatter plots arranged in a grid of two by two. The plots for the BLEU and ROUGE scores are plotted in the first row and in the second the WER and SemDist are plotted. For this experiment, a scatter plot is created for each performance metric. Each audio segment is represented by a dot. The location of the dot depends on the performance metric using the y-axis and the duration using the x-axis.

To get a better understanding of the distribution within one axis a histogram is shown on top of each plot for the x-axis. It shows the distribution of the audio segment's duration. And on the right side of the plot, a histogram is shown that highlights the distribution of the performance metric. In addition to the histograms, a linear regression line has been fitted to the scatter plot to reveal any tendencies and help to prove or disprove the hypothesis.

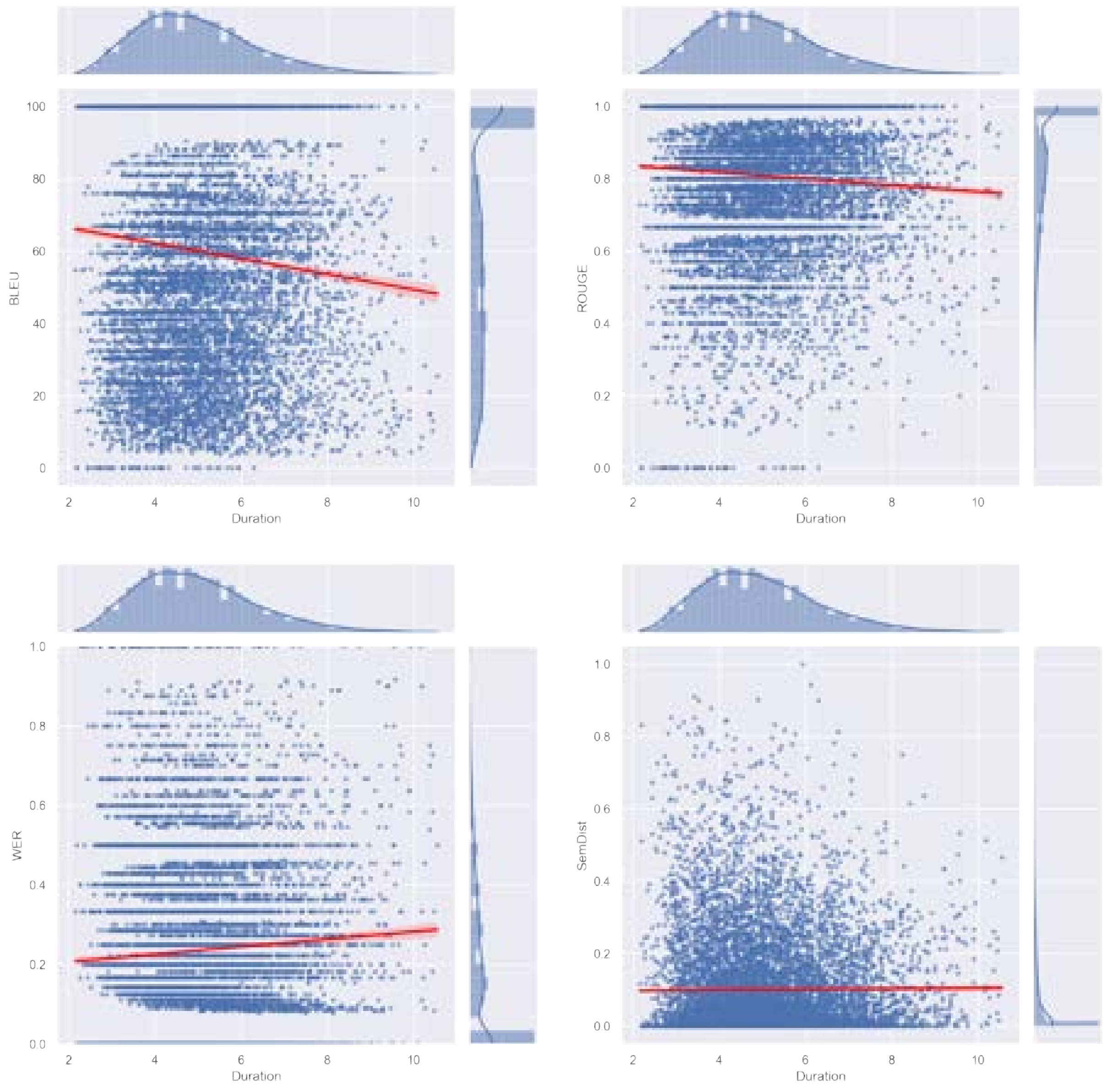

Figure 6.1: Comparison between the duration of audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the ZHAW (lowercase) model.

In Figure 6.1 the ZHAW (lowercase) model shows a decreasing BLEU and ROUGE score on longer audio segments. The WER increases for longer audio segments. The duration seems to have no effect on the SemDist.

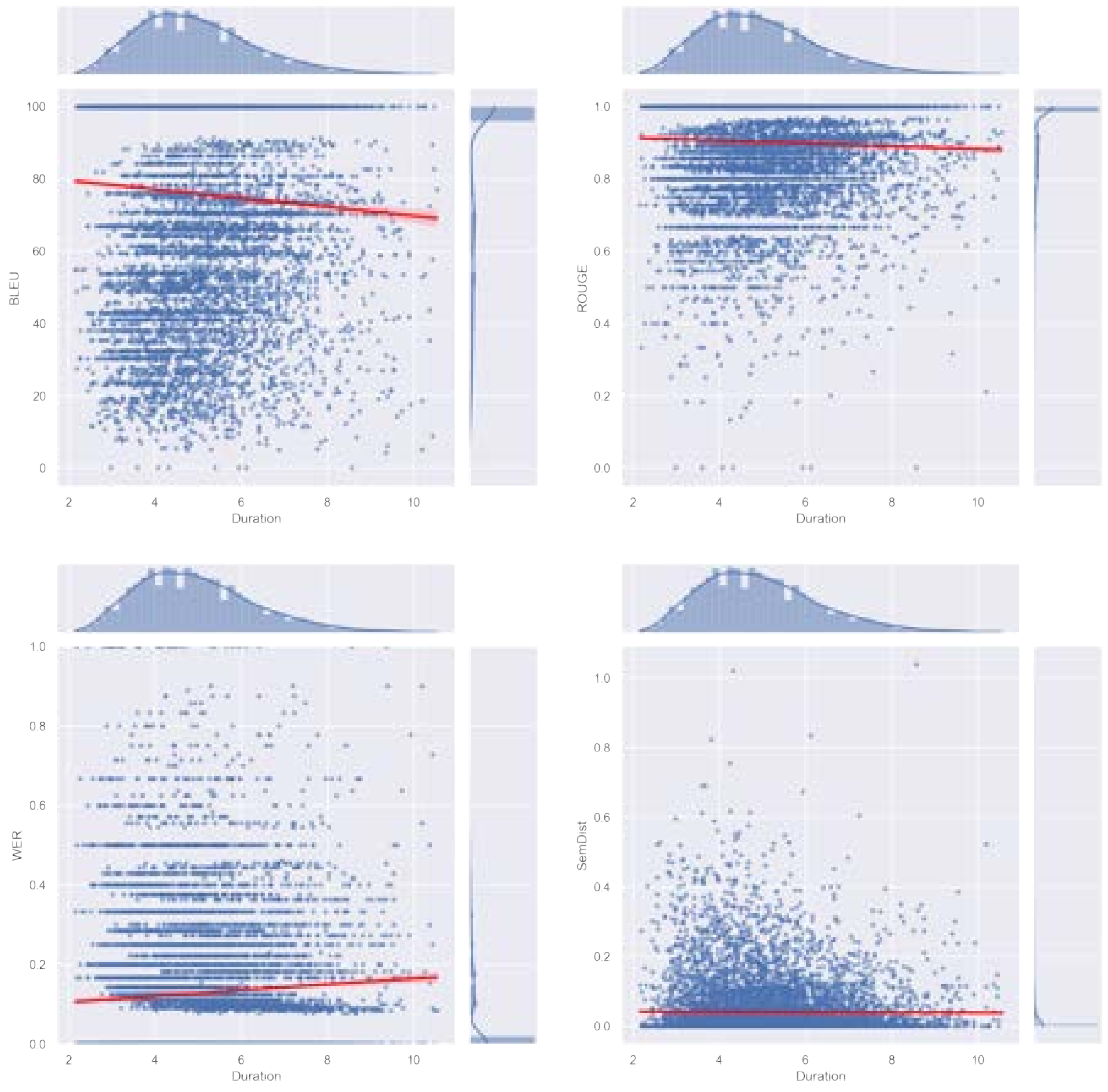

Figure 6.2: Comparison between the duration of audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the FHNW model.

In Figure 6.2 the FHNW model also shows a decreasing BLEU and ROUGE score on longer audio segments. Compared to the ZHAW (lowercase) model in Figure 6.1 it shows an overall better performance. Similar to the WER and SemDist, the WER increases for longer audio segments and is compared to the ZHAW (lowercase) model lower and the SemDist seems to be stable.

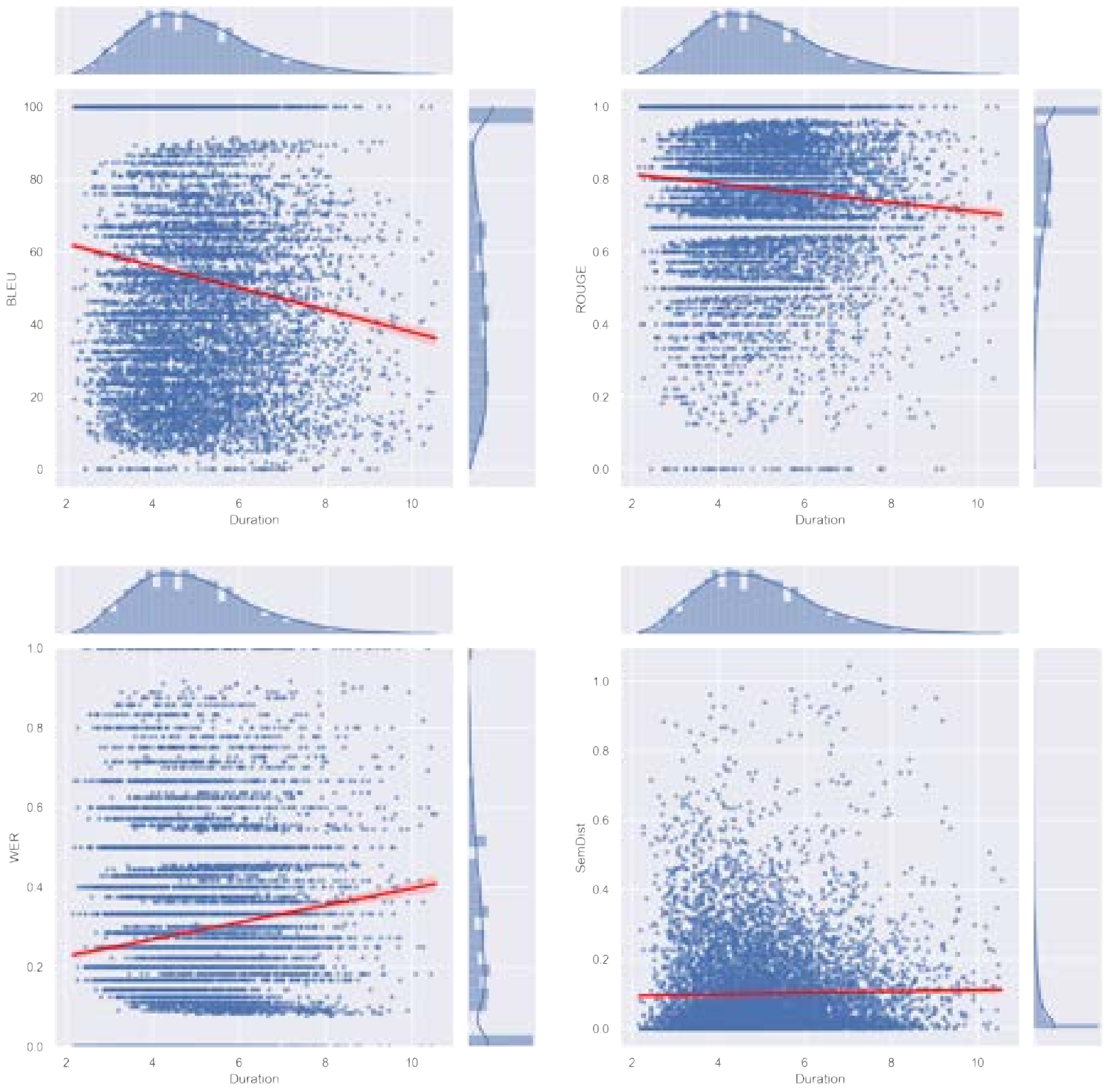

Figure 6.3: Comparison between the duration of audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the Microsoft model.

In Figure 6.3 the Microsoft model shows a decreasing BLEU and ROUGE score on longer audio segments. The WER increases for longer audio segments however the duration seems to have little to no effect on the SemDist.

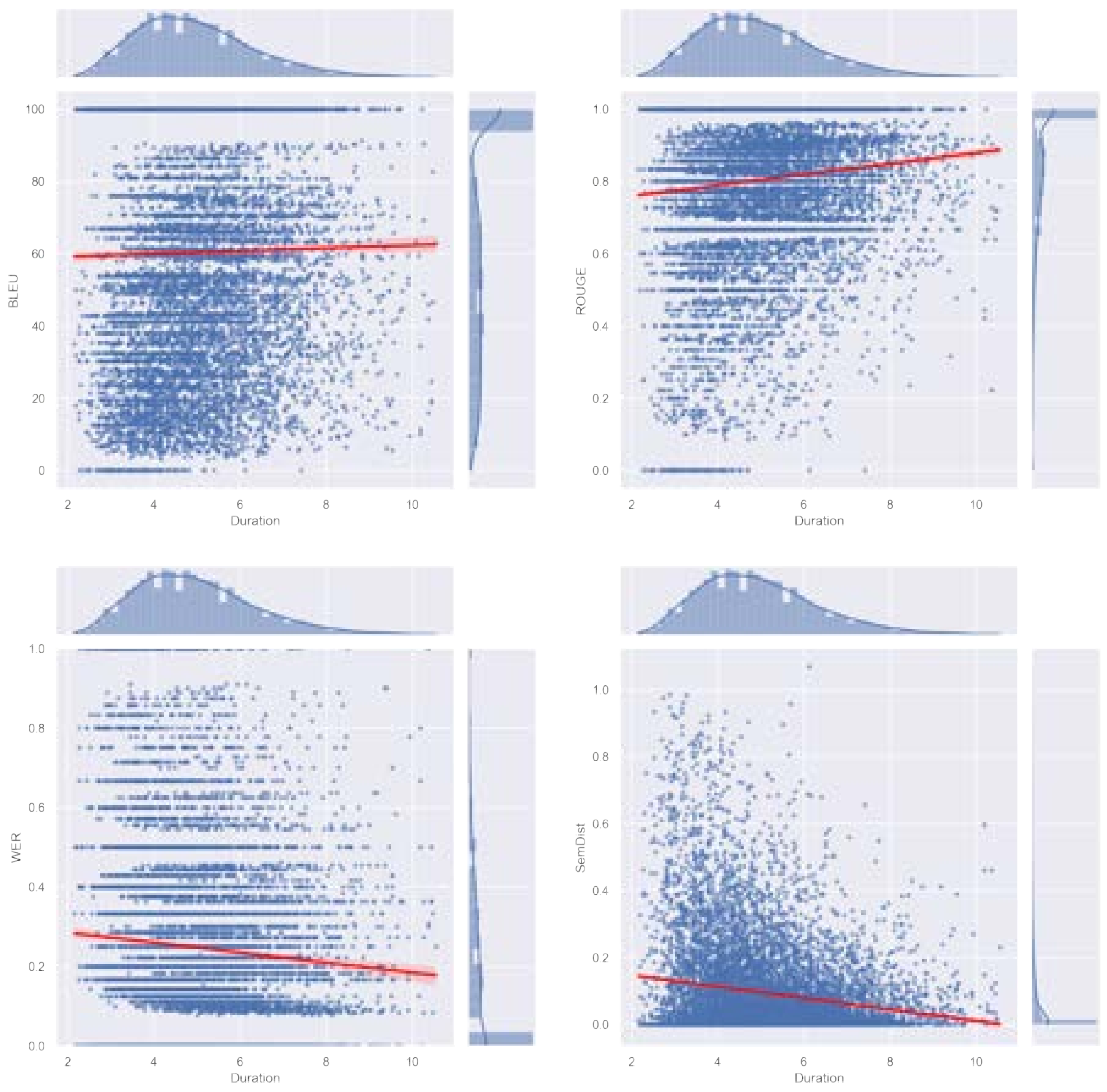

Figure 6.4: Comparison between the duration of audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the Whisper model.

In Figure 6.4 the Whisper model shows a slightly improving BLEU and ROUGE score on longer audio segments. The WER and SemDist decrease on longer audio segments.

Comparing the average scores across the different ST models (Table 6.1) shows that the FHNW model performs the best in this scenario with one sentence per audio segment. The ZHAW (lowercase) models and Whisper score alike while the Microsoft model seems to perform the worst in this experiment.

Model	**Avg. BLEU**	**Avg. ROUGE**	**Avg. WER**	**Avg. SemDist**
ZHAW (lowercase)	60.09	0.81	0.23	0.1
FHNW	75.86	0.9	0.13	0.04
Microsoft	53.13	0.78	0.29	0.1
Whisper	60.33	0.8	0.25	0.1

Table 6.1: Comparison of the average BLEU, ROUGE, WER and SemDist performance scores on the ZHAW (lowercase), FHNW, Microsoft and Whisper model.

6.1.4 Discussion

In this experiment, the relationship between the audio duration and the model performance was examined. The results show decreasing BLEU and ROUGE scores and increasing WER on longer audio segments for the ZHAW (lowercase), FHNW and Microsoft models. The SemDist on the ZHAW (lowercase), FHNW and Microsoft models showed no significant change on longer audio segments.

The Whisper model shows an increasing BLEU and ROUGE score as well as a decreasing WER and SemDist on samples with a longer duration. In this scenario the behavior can be explained by the training process of the Whisper Model, explained in Chapter 3.1.5. Where the training data is split into 30 second segments.

The hypothesis, that speech translation models perform better on shorter audio segments, is accepted for the ZHAW (lowercase), FHNW and Microsoft models. However, for the Whisper model, it cannot be verified with this experiment.

The audio samples have a length between approximately 1 and 15 seconds. It would be interesting to test single sentences that are longer than 15 seconds, however, this is not available in the dataset that was used to conduct the experiment. Since longer audio segments are not available, an experiment (Chapter 6.2) has been designed in which audio segments are artificially increased by adding silence at the beginning and the end of an audio segment.

6.2 PADDIN - Added Padding

In this experiment, the relation between the amount of padding added to the beginning and end of an audio segment and the performance of individual ST models is examined. The results of a previous experiment (Chapter 6.1) showed that the ZHAW (lowercase), FHNW and Microsoft model performance decreased on longer audio segments. This experiment aims to find out if artificially increasing the duration of audio segments influences the performance of the ST models. With this, it can be determined whether the duration is decisive for the decreasing performance or some other factor.

6.2.1 Hypothesis

The amount of added padding to an audio segment does not influence the performance of speech translation models.

6.2.2 Method

For this experiment, the test set of the STT4SG corpus (Chapter 4.2) was chosen because it has not been used during the training of any of the ST models. 10'000 audio segments of various lengths were sampled from the test set and grouped into ten categories. To each category, a specific amount of padding consisting of silence between 0 and 4'500 ms was added. The padded audio segments were then translated and evaluated by the chosen ST models. This experiment is conducted on the ZHAW (lowercase), FHNW (short), Microsoft (short) and Whisper models. Each audio segment contains exactly one sentence and no segmentation has been applied to the audio samples because all ST models under observation are developed to work with audio samples up to 15 seconds. To evaluate the performance of the ST models the BLEU and ROUGE scores were calculated as well as the WER and SemDist.

6.2.3 Results

For each ST model, the performance metrics are plotted using four groups of box plots arranged in a grid of two by two. The plots for the BLEU and ROUGE scores are plotted in the first row and in the second the WER and SemDist are plotted.

Each box plot represents one of ten categories of padding added. The first category shown in blue is the reference case to which no padding has been added. In each other category, a specific amount of padding between 0 and 4'500 ms has been added. The categories are sorted by the amount from smallest to largest.

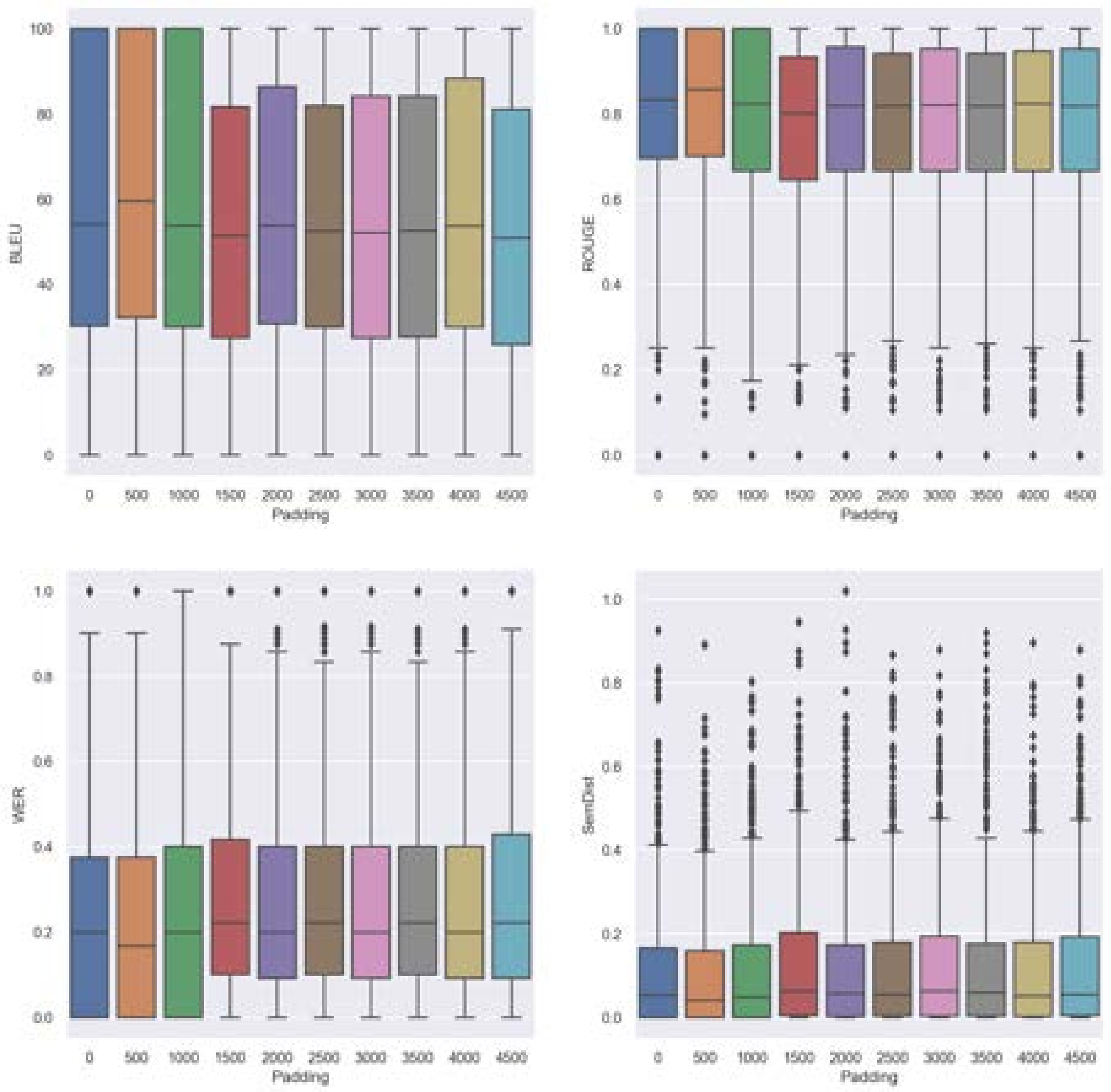

Figure 6.5: Comparison between the amount of padding added to audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the ZHAW (lowercase) model.

In Figure 6.5 the ZHAW (lowercase) model shows that the padding has no significant influence on the BLEU and ROUGE score. However, we see a small difference between padding below 1'000 ms and above. The WER shows a similar trend, below 1'000 ms padding it is consistently lower than above. The padding seems to have no significant effect on the SemDist.

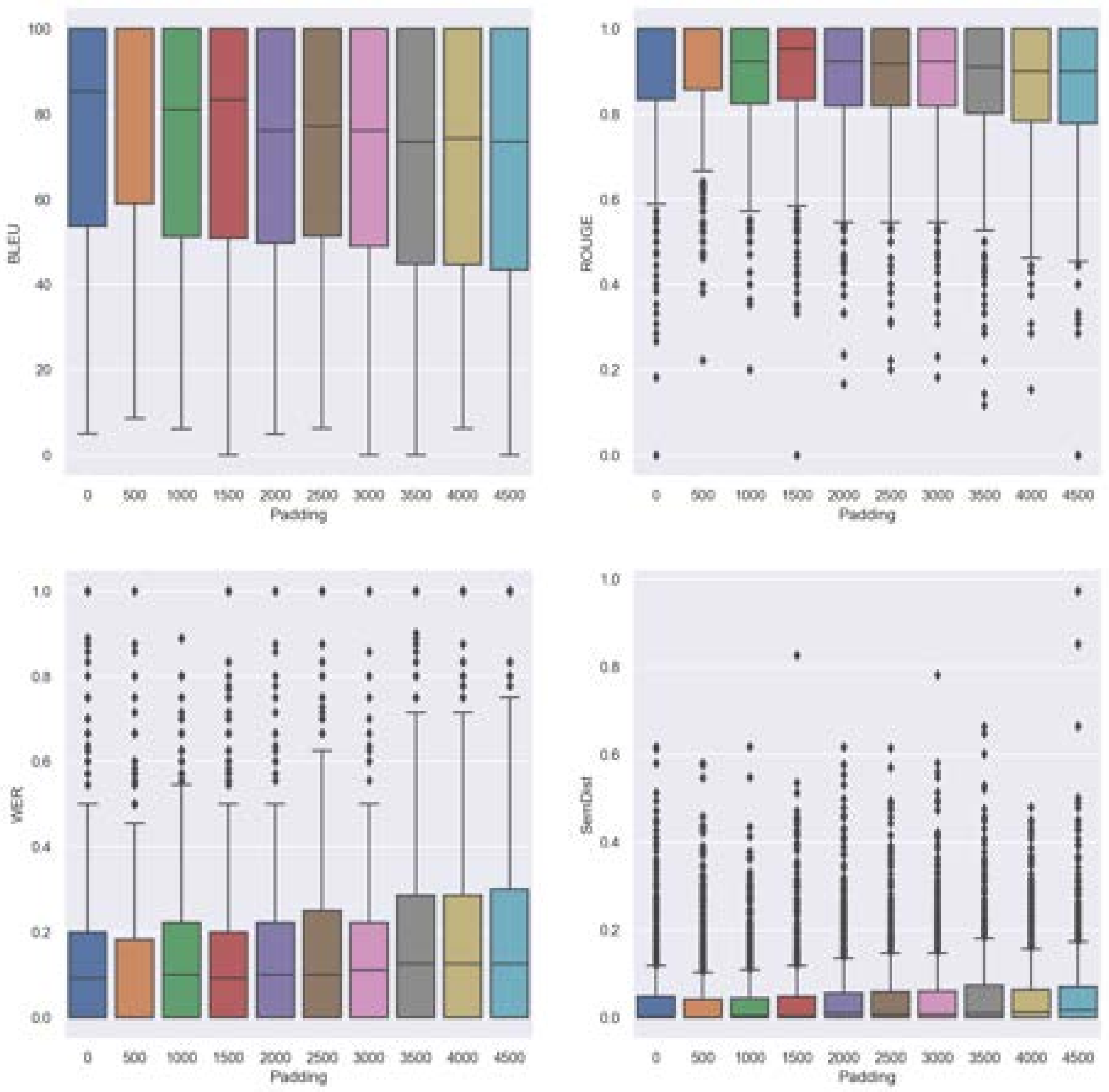

Figure 6.6: Comparison between the amount of padding added to audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the FHNW model.

In Figure 6.6 the FHNW model shows that the padding has a small influence on the BLEU and ROUGE score. Longer padding tends to have slightly lower BLEU and ROUGE scores. The WER also increases if more padding is added. However, the padding has very little influence on the SemDist.

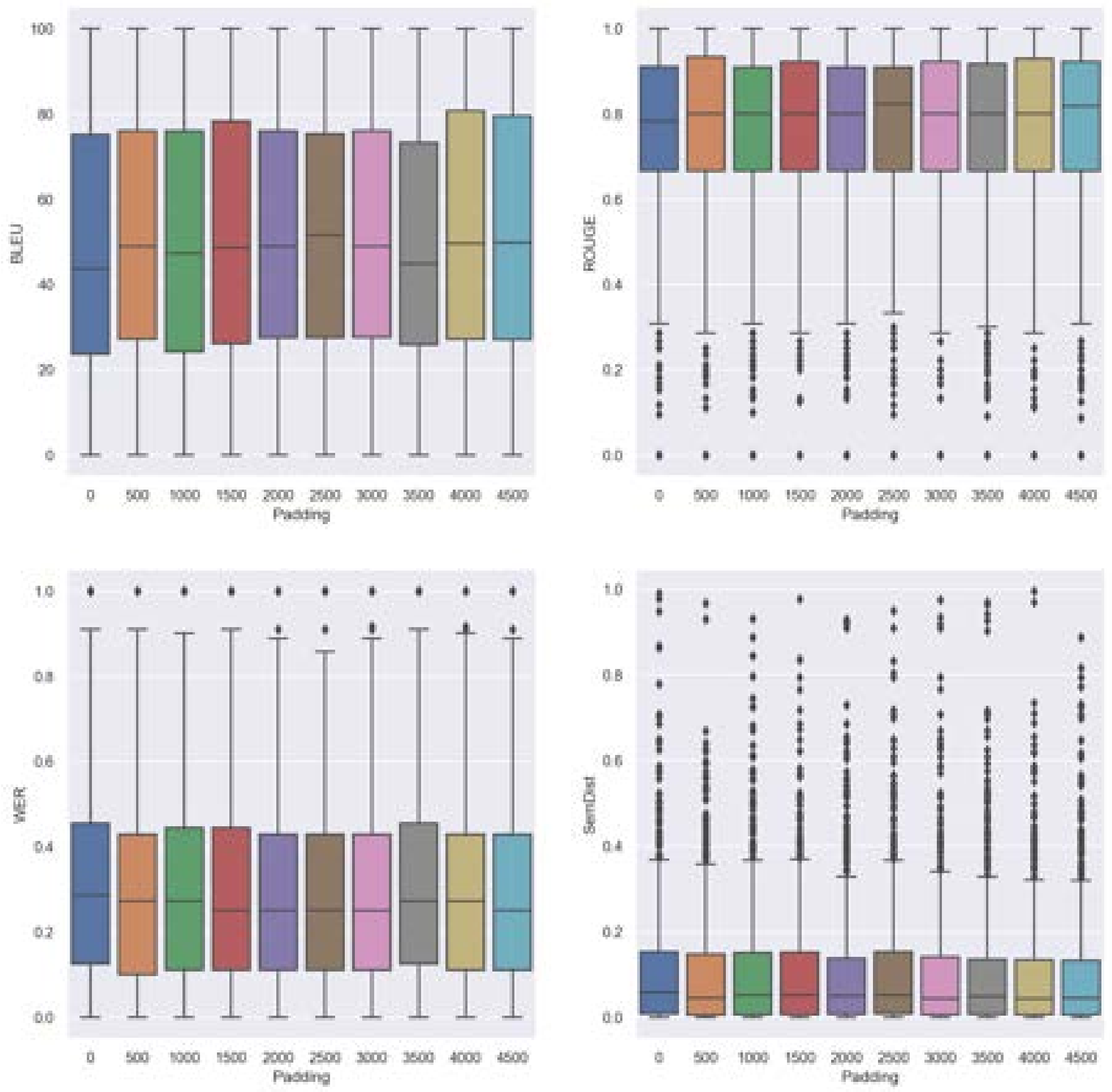

Figure 6.7: Comparison between the amount of padding added to audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the Microsoft model.

In Figure 6.7 the Microsoft model shows that the padding does not influence any of the scores.

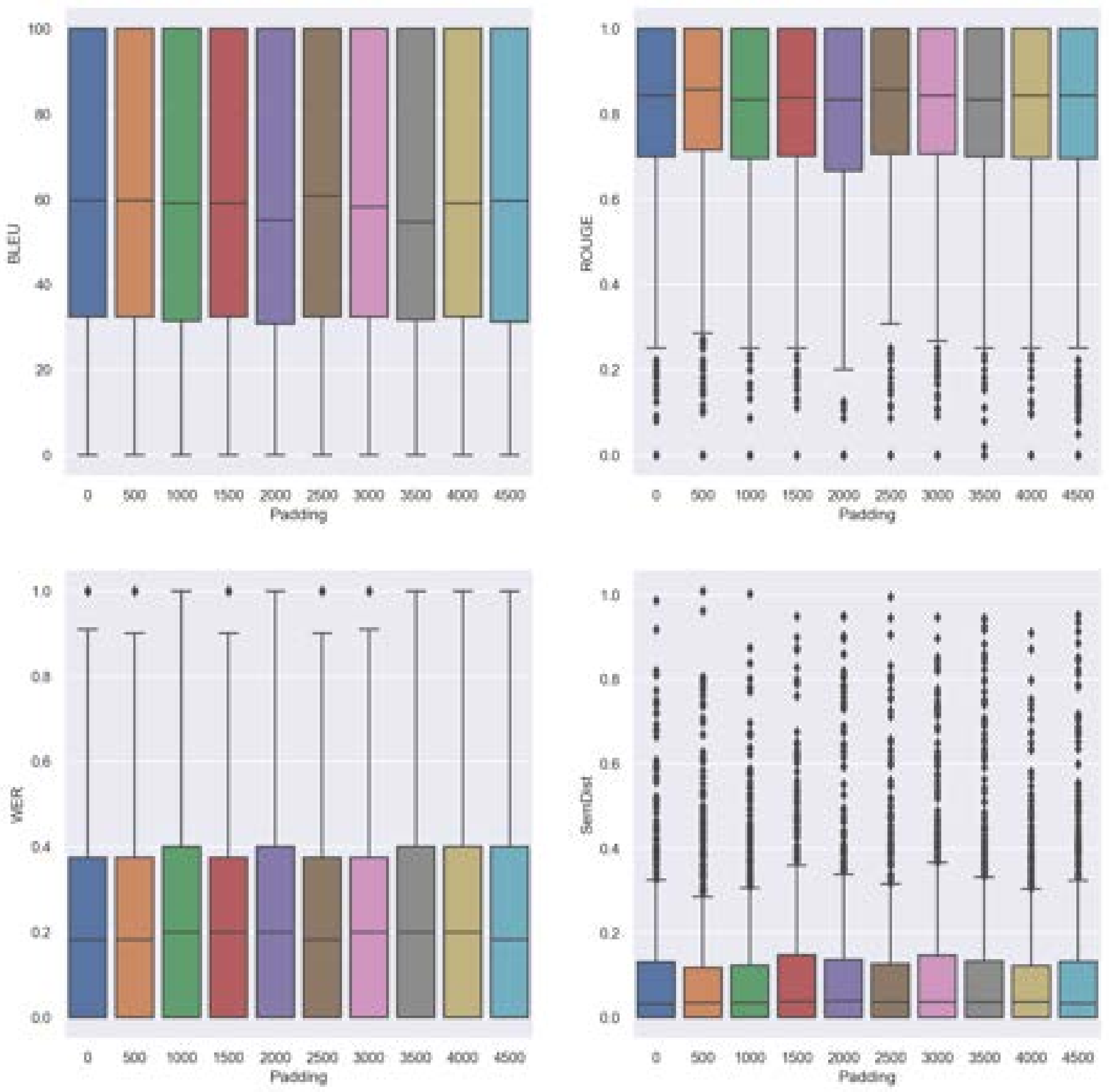

Figure 6.8: Comparison between the amount of padding added to audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the Whisper model.

In Figure 6.8 the Whisper model shows that the padding does not influence any of the scores.

Comparing the average scores across the different ST models (Table 6.2) shows that all models perform well with little difference between them. The Whisper model performs the best overall while the ZHAW (lowercase) model exhibits the lowest scores on all performance metrics.

Model	**Avg. BLEU**	**Avg. ROUGE**	**Avg. WER**	**Avg. SemDist**
ZHAW (lowercase)	56.29	0.78	0.26	0.12
FHNW	73.11	0.89	0.16	0.05
Microsoft	52.65	0.77	0.29	0.1
Whisper	59.74	0.8	0.26	0.1

Table 6.2: Comparison of the average BLEU, ROUGE, WER and SemDist performance scores on the ZHAW (lowercase), FHNW, Microsoft and Whisper model.

6.2.4 Conclusion

In this experiment, the relation between the amount of padding added to the beginning and end of an audio segment and the performance of a ST model was examined. The results showed no significant change in performance when padding was added to an audio segment. Which is to be desired from any ST model. It is therefore conclusive, that the duration itself is not the sole reason for a decrease in performance on longer audio segments.

Thus, the hypothesis, that the amount of added padding to an audio segment does not influence the performance of ST models, is accepted.

It remains uncertain whether an increased duration has any impact on the performance of the ST models. To further investigate this the playback speed is modified to increase and decrease the duration of audio segments (Chapter 6.3).

6.3 SPEEED - Variable Playback Speed

The previous experiments suggest that there is a worsening performance if the audio duration is increased. This behavior is further investigated in this experiment by changing the playback speed of the tested audio segments to increase and decrease the audio duration.

6.3.1 Hypothesis

Modifying the playback speed has a negative influence on the performance of ST models.

6.3.2 Method

For this experiment, the test set of the STT4SG corpus (Chapter 4.2) was chosen because it has not been used during the training of any of the ST models. From the test set, 10'000 samples were selected and grouped into seven different groups. To each segment in a group, a specific factor is applied to the playback speed. The factors range from 0.25 to 1.75 and include factor 1 as a reference group.

To change the playback speed of the selected audio samples two methods were employed. When slowing down audio, pydub (Chapter 3.2) already provides an appropriate function. To speed up the audio, the sample rate was increased and the pitch was adjusted down to compensate for the changed speed.

The created samples are then transcribed by the ZHAW (lowercase), FHNW (short), Microsoft and Whisper models without the need to segment the samples. To evaluate the performance of the ST models the BLEU and ROUGE scores were calculated as well as the WER and SemDist.

6.3.3 Results

For each ST model, the performance metrics are plotted using four groups of box plots arranged in a grid of two by two. The plots for the BLEU and ROUGE scores are plotted in the first row and in the second the WER and SemDist are plotted.

Each box plot represents one of seven factors that are applied to the playback speed. The fourth factor, one, is the reference case to which no factor has been applied.

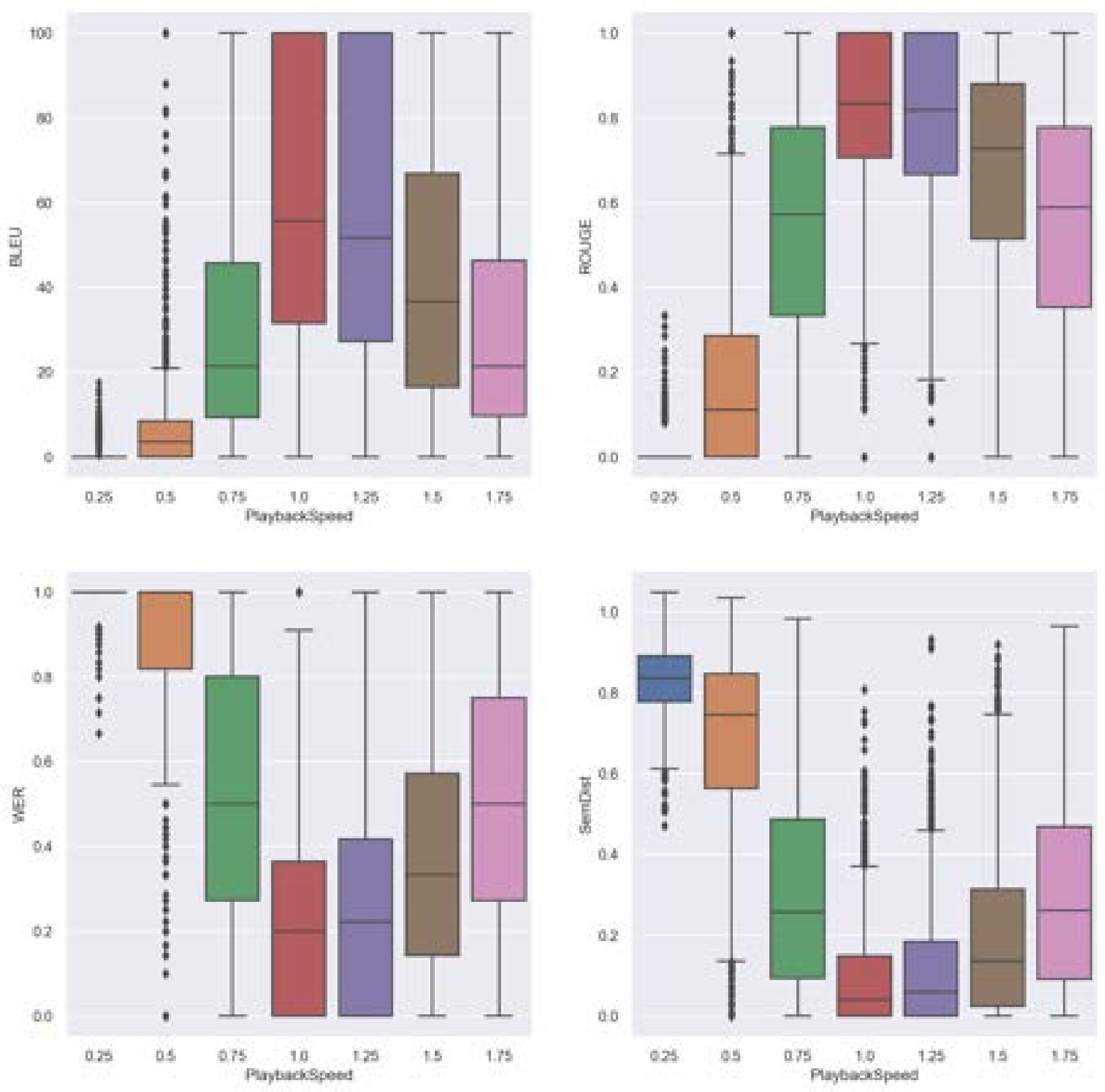

Figure 6.9: Comparison between the duration of audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the ZHAW (lowercase) model.

The ZHAW (lowercase) model shows (Figure 6.9) that the BLEU, ROUGE, WER and SemDist are best when the playback speed is not adjusted. If the audio segments are slowed down the decrease in all performance metrics is worse than when the segments are sped up. When the audio segments are sped up with factor 1.25 the results are still at a usable level.

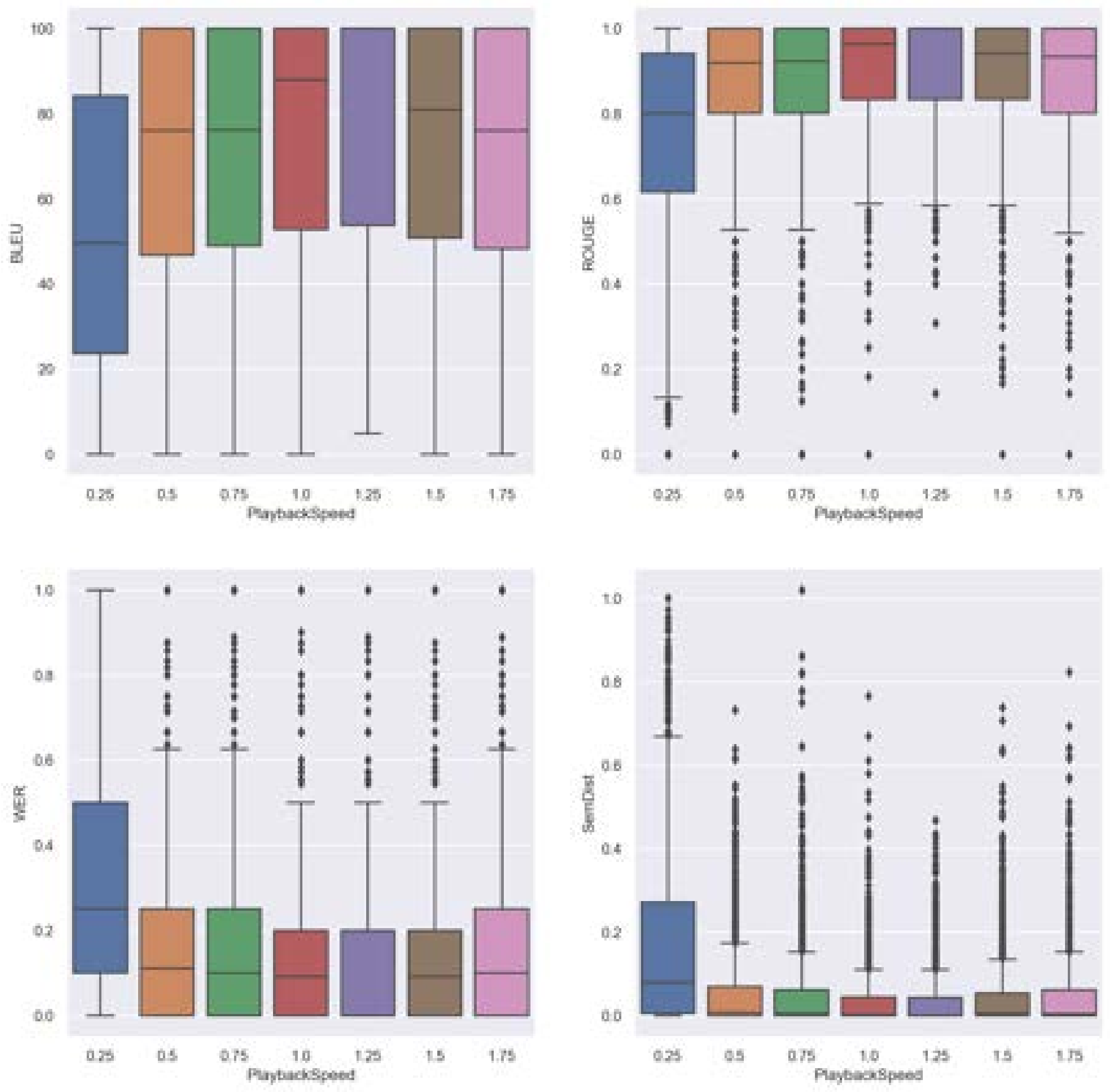

Figure 6.10: Comparison between the duration of audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the FHNW model.

The FHNW model shows (Figure 6.10) a really high performance on all playback speeds except when slowing it down by a factor of 0.25 and when speeding it up by a factor of 1.75. Otherwise, the BLEU and ROUGE score is very high while the WER and SemDist are low. It is also visible that when audio segments are slowed down the decrease in all performance metrics is worse than when the segments are sped up.

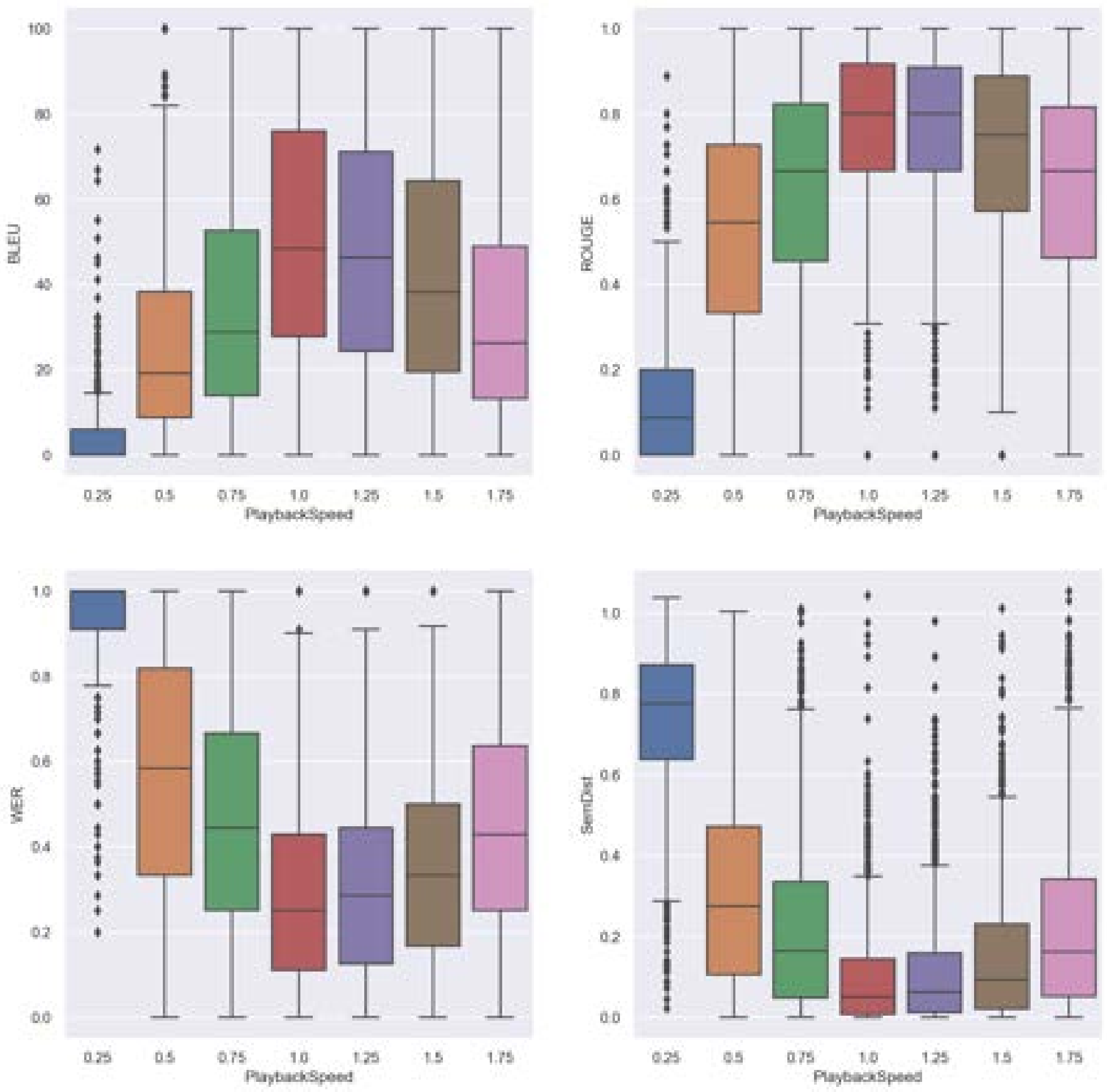

Figure 6.11: Comparison between the duration of audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the Microsoft model.

The Microsoft model shows (Figure 6.11) that the BLEU, ROUGE, WER and SemDist are best when the playback speed is not adjusted. If the audio segments are slowed down the decrease in all performance metrics is worse than when the segments are sped up. When the audio segments are sped up with factor 1.25 the results are still at a usable level. Compared to the ZHAW (lowercase) and FHNW model the decrease in performance is not as drastic and the difference between factor 1 and 1.25 is very small.

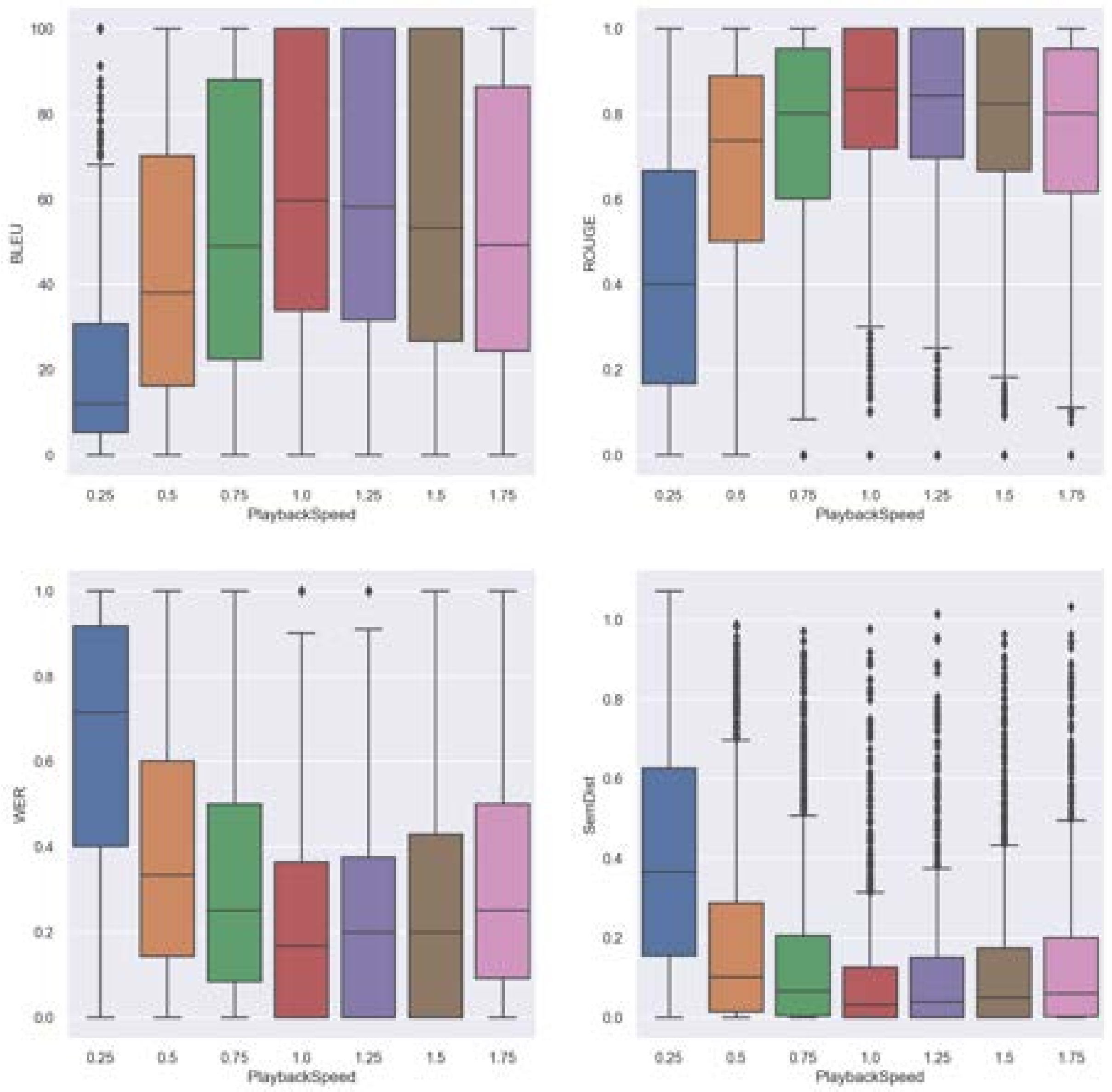

Figure 6.12: Comparison between the duration of audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the Whisper model.

Figure 6.12 shows that the BLEU, ROUGE, WER and SemDist on the Whisper model are best when the playback speed is not adjusted. If the audio segments are slowed down the decrease in all performance metrics is worse than when the segments are sped up. When the audio segments are sped up with factor 1.25 and 1.5 the results are still at a usable level.

Comparing the average scores over all models the FHNW model seems to be the most robust when it comes to playback speed changes (Table 6.3).

Model	**Avg. BLEU**	**Avg. ROUGE**	**Avg. WER**	**Avg. SemDist**
ZHAW (lowercase)	32.97	0.51	0.55	0.36
FHNW	70.80	0.87	0.17	0.07
Microsoft	35.87	0.59	0.50	0.27
Whisper	49.94	0.70	0.36	0.17

Table 6.3: Comparison of the average BLEU, ROUGE, WER and SemDist performance scores on the ZHAW (lowercase), FHNW, Microsoft and Whisper model.

6.3.4 Conclusion

In this experiment, the duration has been modified by changing the playback speeds.

It revealed that slowing down audio segments yields worse results compared to speeding them up. This might be attributed to the algorithm that is used to slow down audio, where additional data is inserted to lengthen the segment. Increasing the playback speed by a factor of 1.25 results in usable transcriptions. The FHNW model demonstrated the most robust handling of speed changes. Compared to the other models it performed extraordinarily on all speed factors except 0.25.

The hypothesis that adjusting the playback speed has a negative influence on the performance of the ST models, can be accepted.

In this experiment, the time it takes to change the playback speed and the time to transcribe the segments has not been recorded. It would be interesting to see if increasing the speed by a factor of 1.25 would lead to an overall shorter inference time and thus lead to a lower resource consumption while still producing usable results.

Instead of changing the duration of audio segments with padding or by slowing it down, it could be changed by increasing the amount of sentences. The impact of more content in an audio segment is examined in the next experiment (Chapter 6.4).

6.4 NRSENT - Number of Sentences

A previous experiment (Chapter 6.2) showed that by increasing the duration with additional silence the ST models performance does not change. This experiment examines if increasing the duration by adding more sentences to an audio segment has an impact on the performance of ST models.

6.4.1 Hypothesis

The performance of a model decreases if more sentences appear within an audio segment.

6.4.2 Method

For this experiment, the test set of the STT4SG corpus (Chapter 4.2) was chosen because it has not been used during the training of any of the ST models. This experiment is conducted on the ZHAW (lowercase), FHNW (long), Microsoft (short), Microsoft (long) and Whisper models. 10'000 audio segments between 2 and 30 seconds were created to execute the experiment, these belong into one of four groups (Figure 6.13).

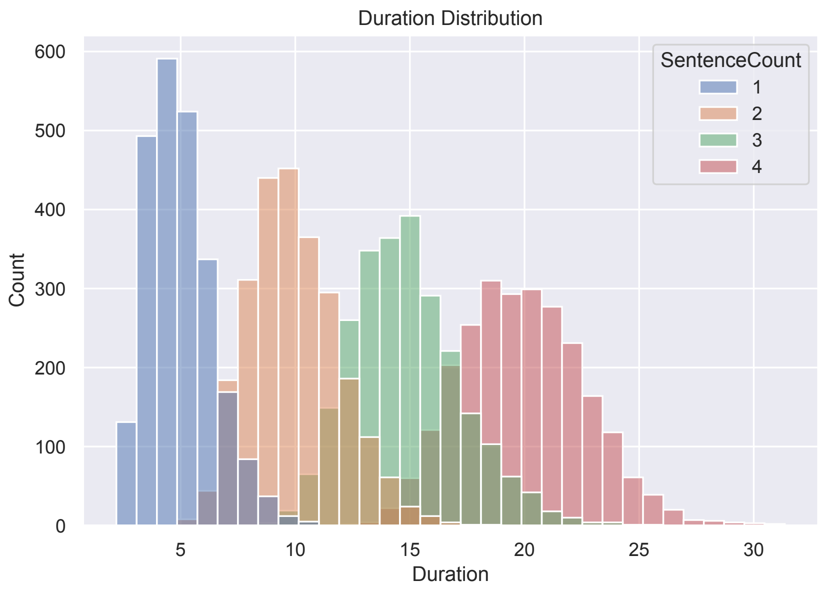

Figure 6.13: Duration distribution per sentence count group of the audio segments used to run the experiment NRSENT.

Audio segments in group one contain one sentence, in group two contain two sentences and so on. A sample from the test set might appear in multiple groups or audio segments but each sentence within one audio segment is distinct.

The FHNW and Microsoft models offer a version of their model that is capable of transcribing audio segments longer than 15 seconds. This is possible by splitting the audio segments, transcribing them individually and merging them before returning the results. For the Microsoft model, both versions are used during this experiment. Ideally, the threshold at which the duration is too long and the audio should be segmented can be found. For the FHNW model only the version that segments the audio is used to get the best possible result. For the ZHAW and Whisper models no segmentation is applied to the audio samples.

To evaluate the performance of the ST models the BLEU and ROUGE scores were calculated as well as the WER and SemDist.

6.4.3 Results

For each ST model, the performance metrics are plotted using four groups of box plots arranged in a grid of two by two. The plots for the BLEU and ROUGE scores are plotted in the first row and in the second the WER and SemDist are plotted.

Blue represents the audio segments that contain one sentence, yellow the ones with two sentences, green segments with three sentences and read the segments with four sentences.

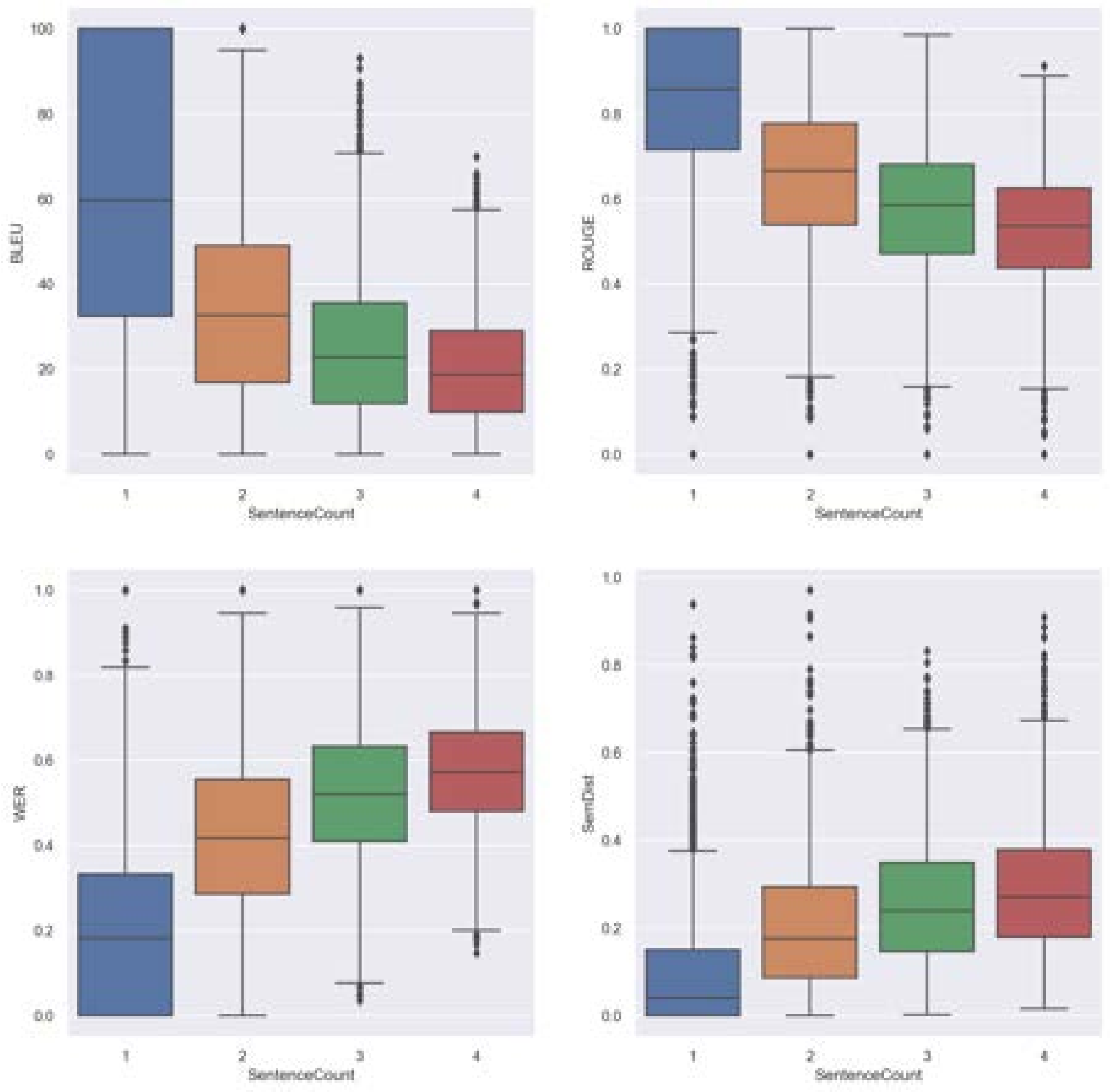

Figure 6.14: Comparison between the number of sentences added to audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the ZHAW (lowercase) model.

The ZHAW (lowercase) model shows a significant drop in performance in Figure 6.14 as soon as more than one sentence appears within a segment. The BLEU and ROUGE scores decrease the more sentences are present in an audio segment. The WER and SemDist increase with more sentences being present. For all scores, the largest variation in the result appears to be in audio segments that contain one sentence.

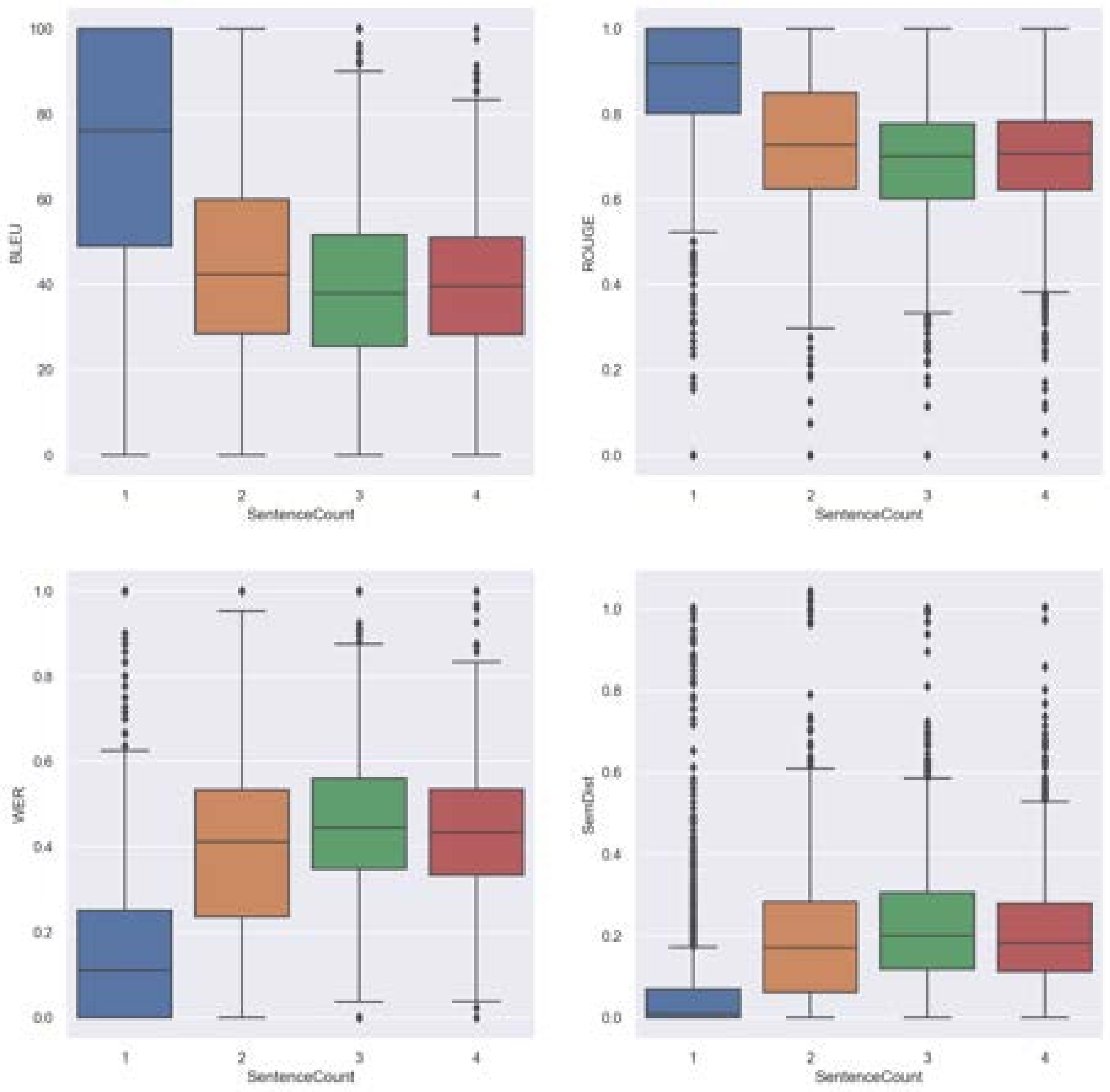

Figure 6.15: Comparison between the number of sentences added to audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the FHNW model.

The FHNW model also shows a significant drop in performance in Figure 6.15 as soon as more than one sentence appears within a segment. The BLEU and ROUGE scores decrease the more sentences are present in an audio segment. The WER and SemDist increase with more sentences being present. For all scores, the largest variation in the result appears to be in audio segments that contain one sentence.

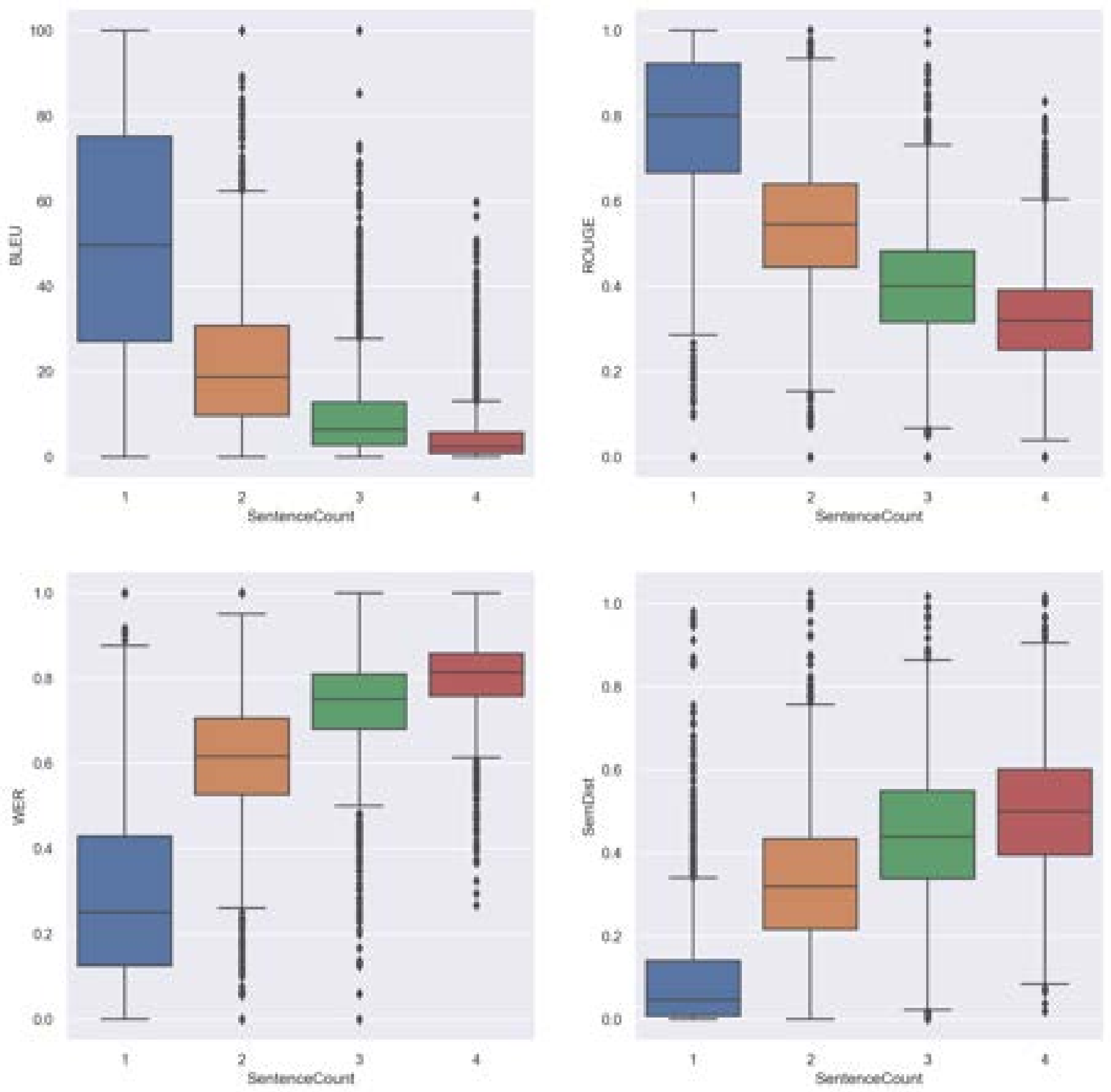

Figure 6.16: Comparison between the number of sentences added to audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the Microsoft (short) model.

The Microsoft model without segmentation shows the largest drop in performance in Figure 6.16 as soon as more than one sentence appears within a segment. The BLEU and ROUGE scores decrease the more sentences are present in an audio segment. The WER and SemDist increase with more sentences being present. For all scores, the largest variation in the result appears to be in audio segments that contain one sentence.

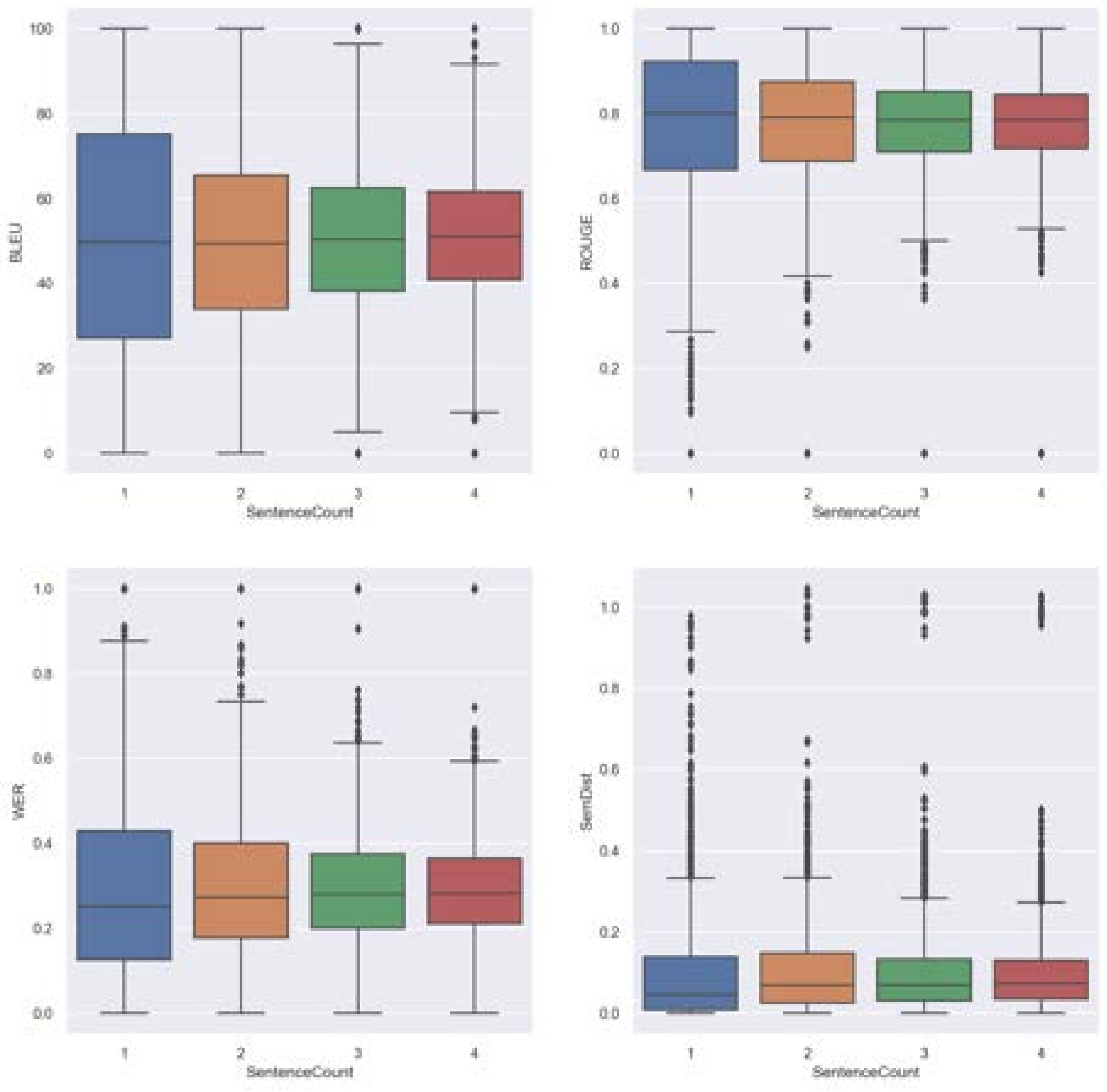

Figure 6.17: Comparison between the number of sentences added to audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the Microsoft (long) model.

The Microsoft model with segmentation shows a very consistent performance on the BLEU and ROUGE scores as well as the WER and SemDist no matter the number of sentences in an audio segment. Only the variation in the results for the BLEU and ROUGE scores changes (Figure 6.17).

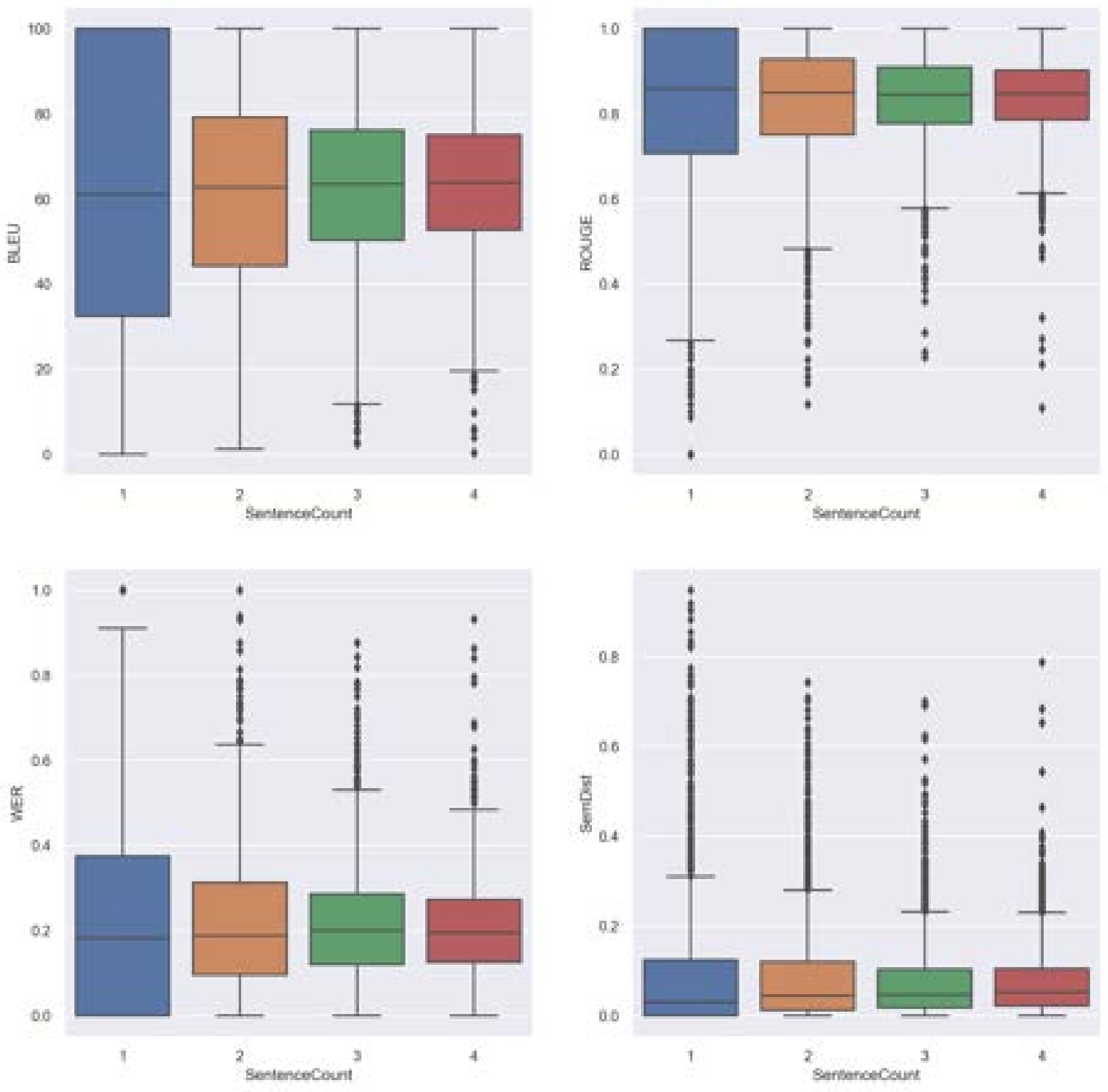

Figure 6.18: Comparison between the number of sentences added to audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the Whisper model.

The Whisper model also shows a consistent performance of all performance metrics on audio segments with multiple sentences. On the BLEU and ROUGE scores as well as the WER a decrease in variation is visible for segments with more sentences (Figure 6.18).

Comparing the average scores across the different ST models (Table 6.4) shows that the Whisper model performs the best in this scenario. The Microsoft model without segmentation scores the worst.

Model	Avg. BLEU	Avg. ROUGE	Avg. WER	Avg. SemDist
ZHAW (lowercase)	34.99	0.64	0.44	0.21
FHNW (long)	48.64	0.74	0.36	0.17
Microsoft (short)	21.95	0.51	0.61	0.35
Microsoft (long)	51.02	0.77	0.29	0.10
Whisper	62.15	0.83	0.22	0.08

Table 6.4: Comparison of the average BLEU, ROUGE, WER and SemDist performance scores for the NRSENT experiment on the ZHAW (lowercase), FHNW (long), Microsoft (short), Microsoft (long) and Whisper model.

6.4.4 Conclusion

In this experiment, the relationship between the audio duration and the model performance was examined.

According to the results from the ZHAW (lowercase) model, when a segment contains more than one sentence, its performance deteriorates to a point where it becomes practically unusable. It is worth noting that the ZHAW (lowercase) model was not exposed to segments with more than one sentence during training. The FHNW model showed the same pattern as the ZHAW (lowercase) model, however, the average performance scores were higher. The performance of the Microsoft (short) model was also notably inadequate when it came to segments comprising multiple sentences. Surprisingly, even in two-sentence segments, which are below 15 seconds, the model's performance remained subpar. This suggests that the model is likely not trained on segments containing multiple sentences. The Microsoft (long) model demonstrates remarkable stability in its performance. This suggests that there is a good method of segmenting longer audio samples into smaller chunks and merging them correctly. The Whisper model even showed an increasing performance, when a segment contains more than one sentence. This can be explained by the training process of the model where it was exposed to segments with 30 seconds length and almost none of the test samples from this experiment exceeded the 30 second length.

In conclusion, the findings indicate that the decline in performance observed in longer audio segments cannot be solely attributed to the increased duration. Rather, it predominantly arises from the increased number of sentences within the segments. This issue is inherent to the training approach of the ST models. Consequently, in order to achieve optimal outcomes, the application of segmentation and merging techniques becomes imperative when dealing with multi-sentence segments. Moreover, when a model is exclusively trained on single sentence segments, it becomes crucial to accurately identify the sentence boundaries during the segmentation process.

Thus the hypothesis can be accepted for the ZHAW (lowercase), and Microsoft (short) models. Whenever more than one sentence appears in an audio segment the performance drops significantly. For the Whisper model the hypothesis cannot be accepted, its performance stays consistent even with more than one sentence per segment.

Unfortunately, due to time constraints, the FHNW (short) version could not be tested, it would be interesting to see if it showed the same behavior as the ZHAW (lowercase) and Microsoft (short) models. Furthermore, it would also be insightful to test the ZHAW (lowercase) model with some kind of segmentation and merging to compare it to the FHNW (long) and Microsoft (long) versions.

The poor performance of the ZHAW (lowercase) and Microsoft short models can be attributed to their training methodology. However, it is worth considering whether there exists a potential solution to enhance their performance by incorporating padding between the sentences (Chapter 6.5). The results suggested to also explore the use of a model specifically trained on audio segments containing multiple sentences to improve the performance in such cases. By training the model on data that more closely matches real-world scenarios, better results can be achieved for speech translation tasks involving multiple sentences (Chapter 6.6).

6.5 SENPAD - Padding between Sentences

The previous experiment (Chapter 6.4) examined the relationship between the number of sentences and the performance. It demonstrated that without some kind of segmentation of the audio files the performance of the ST models drops significantly. This experiment tries to improve the performance of the ST models without segmenting the input audio segments but instead by adding padding between two sentences.

6.5.1 Hypothesis

The performance of a model increases if additional padding between two sentences is added.

6.5.2 Method

For this experiment, the test set of the STT4SG corpus (Chapter 4.2) was chosen because it has not been used during the training of any of the ST models. 10'000 audio segments containing each two sentences and a varying amount of padding between 0 and 1000 ms were created for this experiment. There are five groups of padding each increasing the padding size by 250 ms. The padded audio segments were then translated by the ZHAW, FHNW (short), Microsoft (short) and Whisper models. To evaluate the performance of the ST models the BLEU and ROUGE scores were calculated as well as the WER and SemDist.

6.5.3 Results

For each ST model, the performance metrics are plotted using four groups of box plots arranged in a grid of two by two. The plots for the BLEU and ROUGE scores are plotted in the first row and in the second the WER and SemDist are plotted.

Each box plot represents one of five categories of padding added. The first category shown in blue is the reference case to which no padding has been added. In each other category, a specific amount of padding between 0 and 1000 ms has been added. The categories are sorted by the amount from smallest to largest.

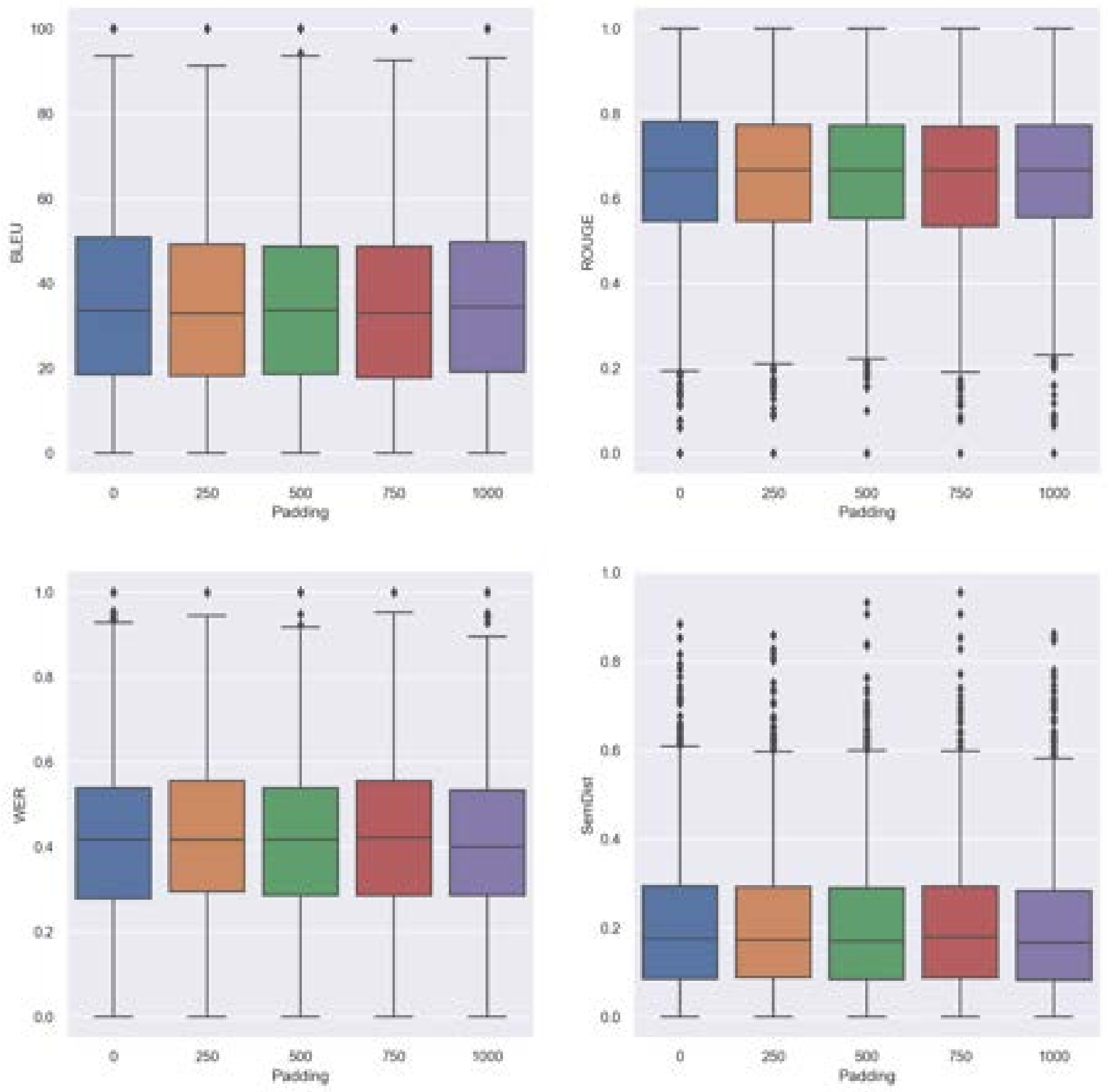

Figure 6.19: Comparison between the amount of padding added between two sentences and the performance (BLEU, ROUGE, WER and SemDist) of the ZHAW (lowercase) model.

In Figure 6.19 the ZHAW (lowercase) model shows that the padding has no significant influence on the BLEU and ROUGE scores as well as the WER and SemDist.

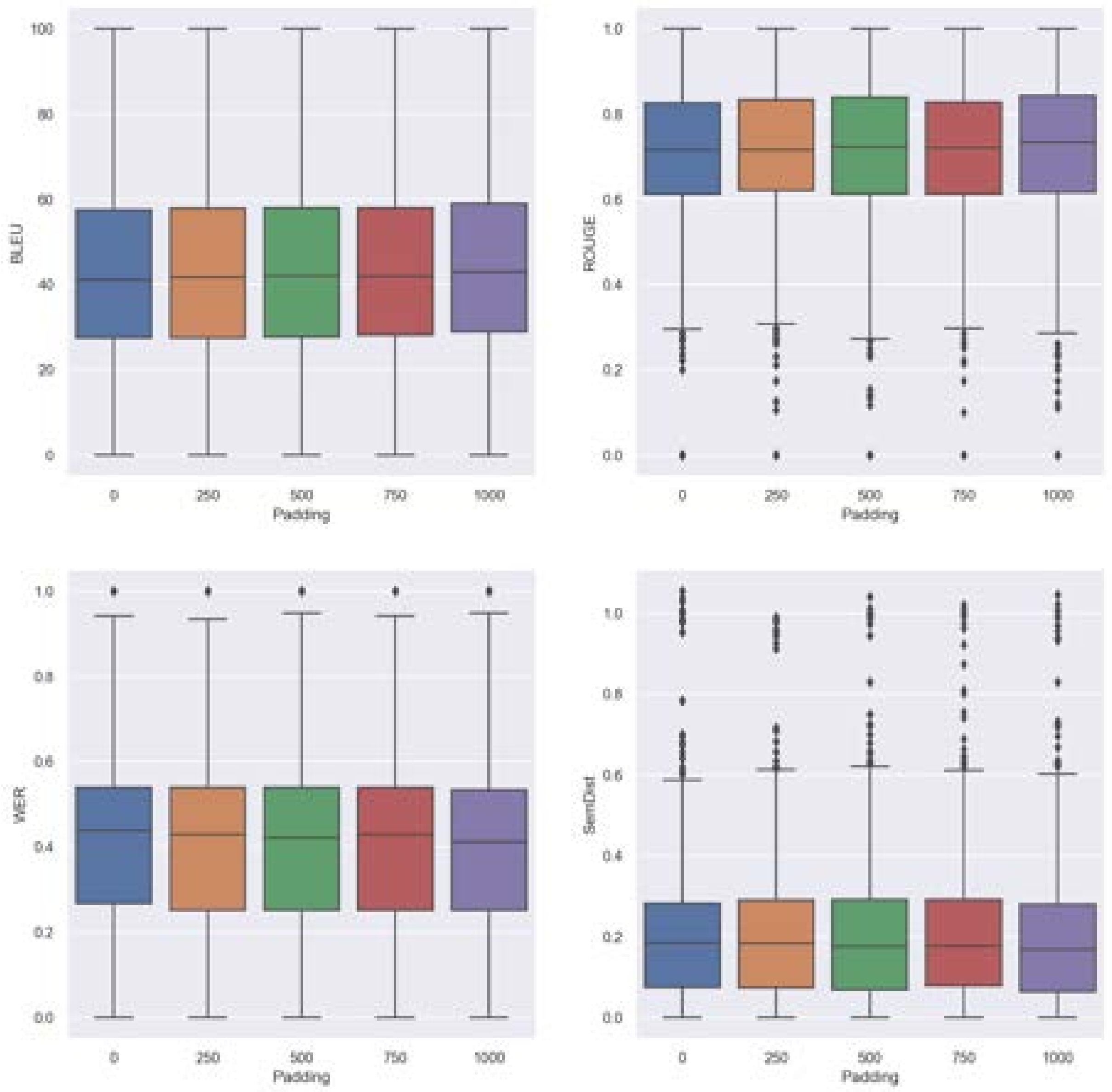

Figure 6.20: Comparison between the amount of padding added between two sentences and the performance (BLEU, ROUGE, WER and SemDist) of the FHNW model.

In Figure 6.20 the FHNW model shows that the padding has no significant influence on the BLEU and ROUGE scores as well as the WER and SemDist.

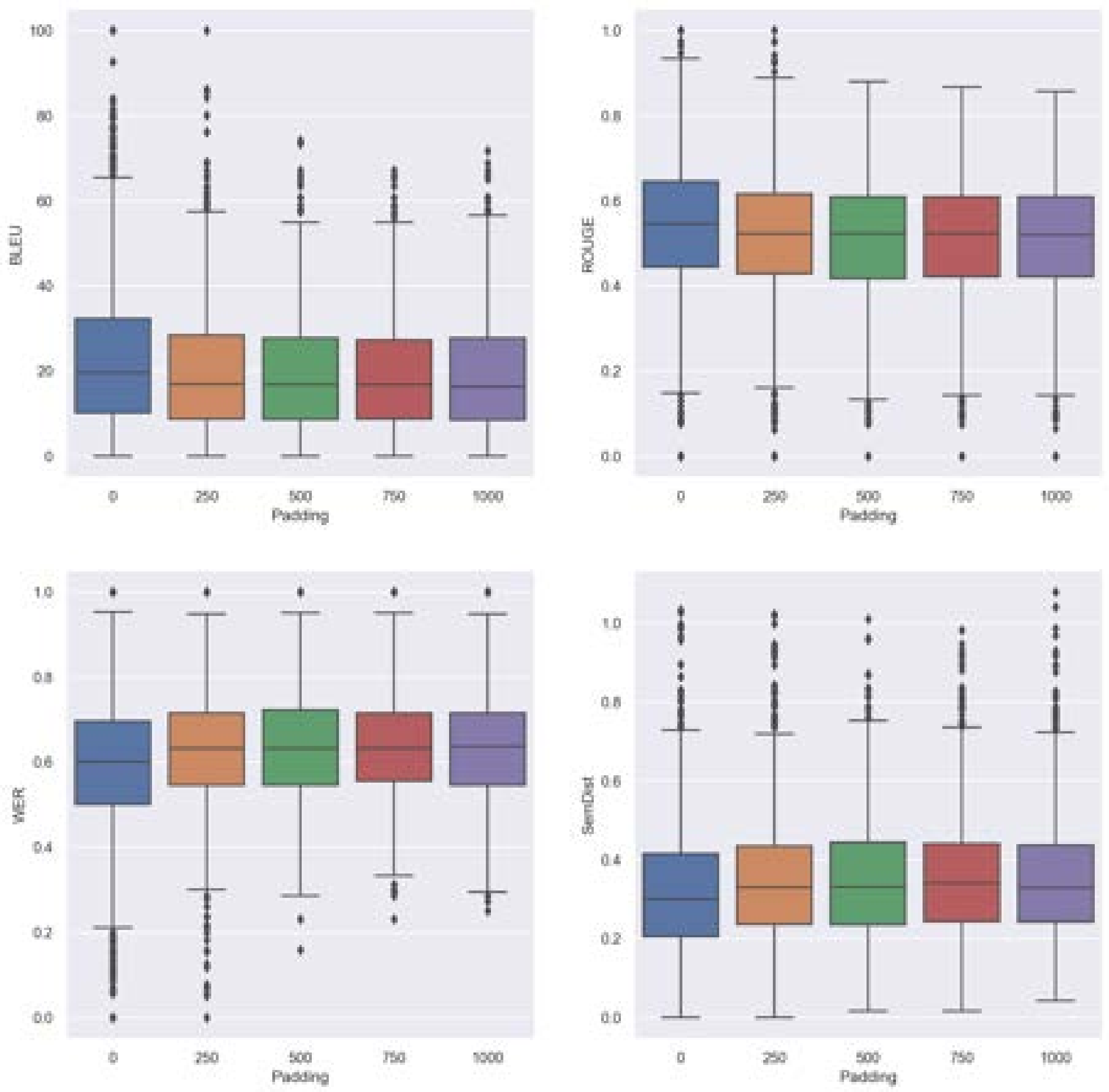

Figure 6.21: Comparison between the amount of padding added between two sentences and the performance (BLEU, ROUGE, WER and SemDist) of the Microsoft model.

In Figure 6.21 the Microsoft (short) model shows that the padding has no significant influence on the BLEU and ROUGE scores as well as the WER and SemDist.

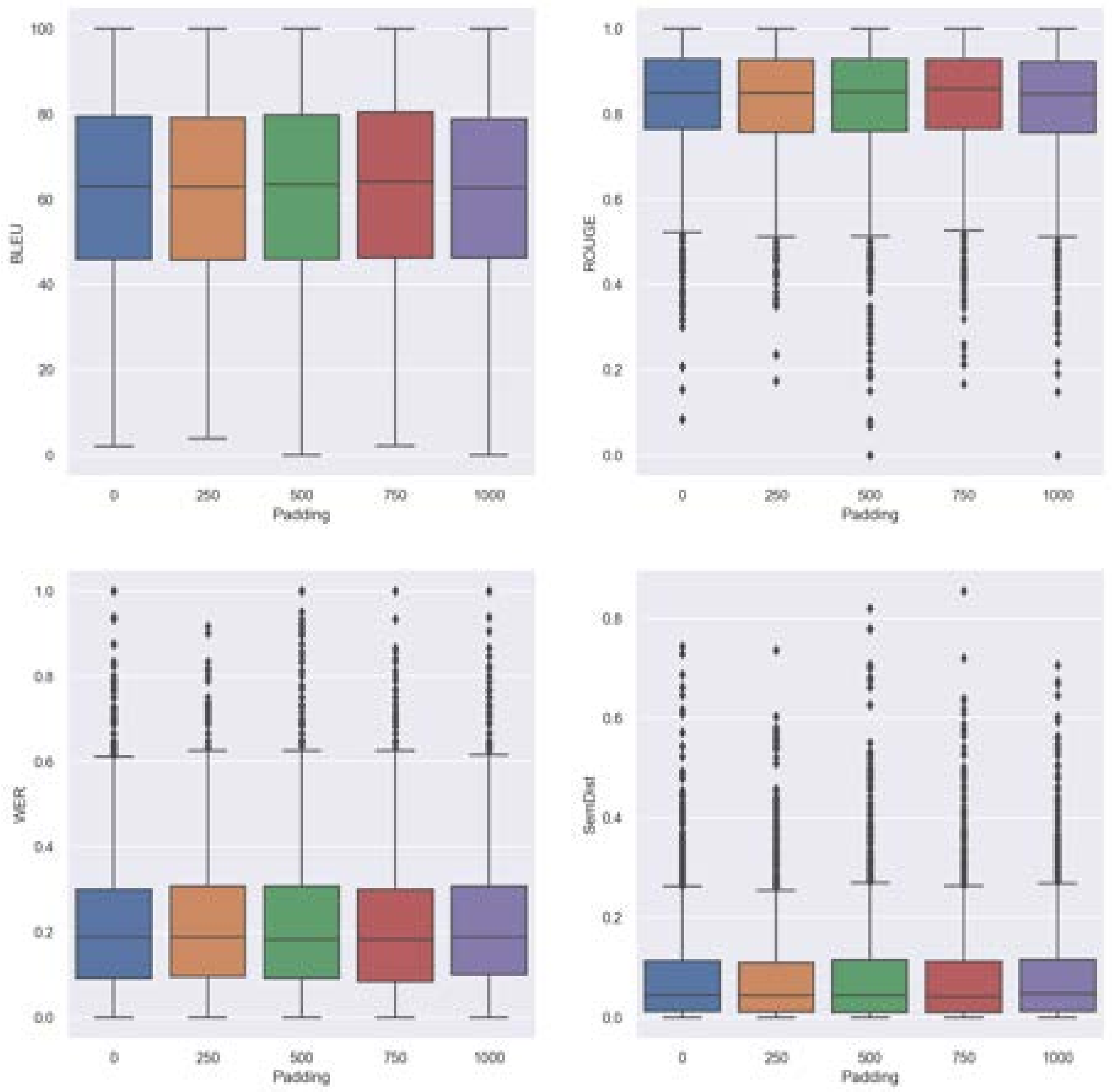

Figure 6.22: Comparison between the amount of padding added between two sentences and the performance (BLEU, ROUGE, WER and SemDist) of the Whisper model.

In Figure 6.22 the Whisper model shows that the padding has no significant influence on the BLEU and ROUGE score as well as the WER and SemDist.

6.5.4 Conclusion

In this experiment, it was investigated whether the performance of the ST models on two sentence segments could be improved by adding additional padding between two sentences.

None of the models under observation showed an improvement with the added padding. This shows again that just because the duration of the audio segments changed the performance does not have to change.

The hypothesis, that the performance of a model increases if additional padding between two sentences is added, cannot be accepted.

6.6 MULSEN - Improved Multi Sentence Model

The previous experiments have shown that the ZHAW (lowercase), FHNW and Microsoft models cannot handle segments with multiple sentences well without some kind of segmentation. In this experiments, a model was specifically trained on multi-sentence segments and compared to the ZHAW (lowercase) model which is exclusively trained on single sentence segments.

6.6.1 Hypothesis

The performance of a model on multi-sentence segments increases if it was trained on audio segments containing multiple sentences.

6.6.2 Method

For this experiment the VARLEN (Chapter 6.1) and NRSENT (Chapter 6.4) experiments are conducted on the newly trained ZHAW (multi-sentence) model. The dataset, preprocessing and performance evaluations are the same as on the previous experiments. The only difference is the model used to transcribe the data.

6.6.3 Results

The performance metrics from the VARLEN experiment are plotted using four scatter plots arranged in a grid of two by two. The plots for the BLEU and ROUGE scores are plotted in the first row and in the second the WER and SemDist are plotted. For this experiment, a scatter plot is created for each performance metric. Each audio segment is represented by a dot. The location of the dot depends on the performance metric using the y-axis and the duration using the x-axis.

To get a better understanding of the distribution within one axis a histogram is shown on top of each plot for the x-axis. It shows the distribution of the audio segments duration. And on the right side of the plot, a histogram is shown that highlights the distribution of the performance metric. In addition to the histograms, a linear regression line has been fitted to the scatter plot to reveal any tendencies and help to prove or disprove the hypothesis.

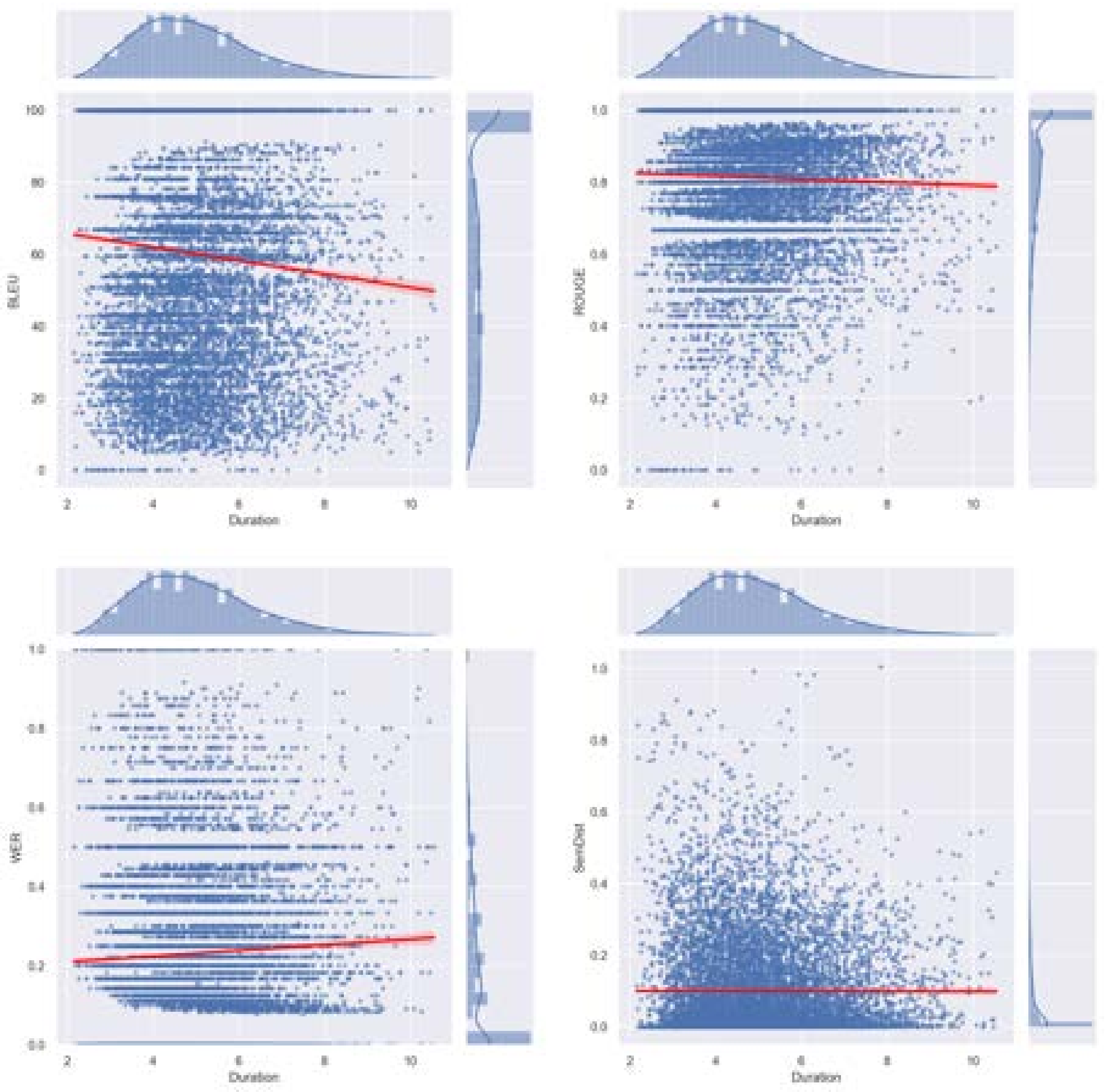

Figure 6.23: Comparison between the duration of audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the ZHAW (multi-sentence) model.

The BLEU, ROUGE, WER and SemDist for the ZHAW (multi-sentence) model (Figure 6.23) are almost identically to the ones from the ZHAW (lowercase) model on the VARLEN experiment (Figure 6.1). The BLEU and ROUGE scores tend to decrease on segments with a longer duration, while the WER increases. The SemDist is stable across different durations.

For the results of the NRSENT experiment, the performance metrics are plotted using four groups of box plots arranged in a grid of two by two. The plots for the BLEU and ROUGE scores are plotted in the first row and in the second the WER and SemDist are plotted.

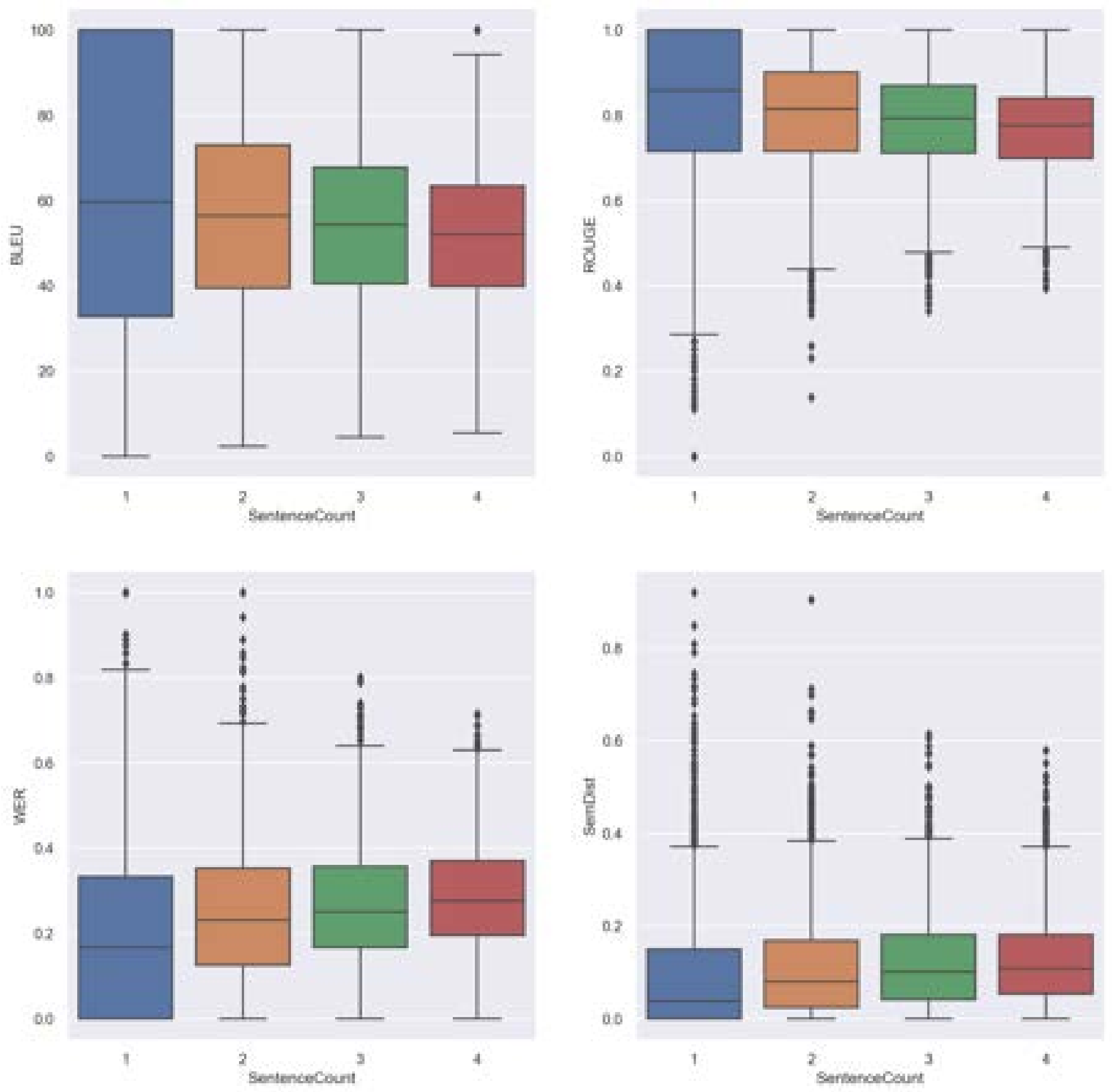

Figure 6.24: Comparison between the number of sentences added to audio segments and the performance (BLEU, ROUGE, WER and SemDist) of the ZHAW (multi-sentence) model.

The BLEU and ROUGE scores seem to decrease lightly (Figure 6.24) and the WER and SemDist increase the more sentences appear in a segment. However, compared to the ZHAW (lowercase) Model (Figure 6.14) the decrease is less drastic and the results are in a usable range.

Compared to the ZHAW (lowercase) model the new ZHAW (multi-sentence) model performs equally on

Model	**Avg. BLEU**	**Avg. ROUGE**	**Avg. WER**	**Avg. SemDist**
VARLEN ZHAW (lowercase)	60.09	0.81	0.23	0.10
VARLEN ZHAW (multi-sentence)	60.12	0.81	0.23	0.10
NRSENT ZHAW (lowercase)	34.99	0.64	0.44	0.21
NRSENT ZHAW (multi-sentence)	55.39	0.79	0.26	0.12

Table 6.5: List of the average BLEU, ROUGE, WER and SemDist performance scores on the VARLEN and NRSENT experiment for the ZHAW (multi-sentence) model.

the VARLEN experiment and much better on the NRSENT experiment as shown in Table 6.5.

6.6.4 Conclusion

In this experiment a model specifically trained on multi-sentence segments is tested on the VARLEN and NRSENT experiments.

The ZHAW (multi-sentence) model exhibits comparable performance to the ZHAW (lowercase) model in the VARLEN experiments. As expected, the ZHAW (multi-sentence) model outperforms the ZHAW (lowercase) model in the NRSENT experiment. This experiment shows something quite obvious, if a model is trained on a specific task, it performs well on said task. However, while it does perform well on multi-sentence segments it also performs very well on single sentence segments.

The hypothesis, that the performance of a model on multi-sentence segments increases if it was trained on audio segments containing multiple sentences, can be accepted.

In this experiment, no segmentation has been applied. To investigate if the ZHAW (multi-sentence) model is useful in real-life scenarios it also needs to perform well on segmented audio where sentences might be cut. This is further explored in the MERGER experiment (Chapter 6.7).

6.7 MERGER - Compare Merging

Previous experiments have shown that it is crucial to segment longer audio recordings before transcribing them. In this experiment, two segmentation methods are compared.

(1) A sentence-based segmentation that segments audio recordings by sentences. This is considered an ideal approach for models that have been trained on audio segments containing exactly one sentence. With a sentence-based segmentation, the transcribed sentences can be joined one by one without any special algorithm for combining them.

(2) A fixed-window segmentation where audio recordings are split into fixed lengths with an overlap between the segments. This splits sentences which might not be ideal for a model trained solely on full sentences. This approach also requires a merging algorithm that combines two adjacent segments based on their overlap.

It is expected that the fixed-window segmentation approach performs worse, however, if a model can handle incomplete sentences then it might still result in very similar results as the sentence-based sentence.

6.7.1 Hypothesis

The performance of an ST model decreases if a fixed-window segmentation is used compared to a sentence-based segmentation.

6.7.2 Method

For this experiment, the test set of the STT4SG corpus (Chapter 4.2) was chosen because it has not been used during the training of any of the ST models. From the test set 1'000 audio recordings have been created with each containing 30 different sentences.

The experiment is conducted on the ZHAW (lowercase), ZHAW (multi-sentence), Microsoft and Whisper models. Unfortunately due to time constraints the FHNW model could not be tested. For each model, both segmentation methods are tested.

For the sentence-based segmentation, each of the 30 sentences is transcribed individually before combining the results in a final transcript which is used to evaluate the BLEU, ROUGE, WER and SemDist scores.

For the fixed-window segmentation, a window size of 10 seconds with 5 seconds overlap is used on the ZHAW (lowercase), ZHAW (multi-sentence) and Microsoft model. For the Whisper model, a 30 second window with 10 second overlap is chosen, because it has been trained on 30 second segments. Each segment is then transcribed and merged with the next segment. The performance metrics are evaluated on the final merged transcript.

6.7.3 Results

To compare the two segmentation methods the average BLEU, ROUGE, WER and SemDist scores are collected per method.

Model	**Avg. BLEU**	**Avg. ROUGE**	**Avg. WER**	**Avg. SemDist**
ZHAW (lowercase)	59.73	0.80	0.24	0.10
ZHAW (multi-sentence)	60.27	0.81	0.23	0.10
Microsoft	52.24	0.78	0.29	0.09
Whisper	60.93	0.81	0.24	0.08

Table 6.6: Average BLEU, ROUGE, WER and SemDist performance scores for a sentence-based segmentation on the ZHAW (lowercase), ZHAW (multi-sentence), Microsoft and Whisper model.

Table 6.6 shows that the sentence-based segmentation averages the same as the models scored on the VARLEN experiment (Chapter 6.1).

Model	**Avg. BLEU**	**Avg. ROUGE**	**Avg. WER**	**Avg. SemDist**
ZHAW	24.10	0.56	0.55	0.27
ZHAW (multi-sentence)	54.22	0.77	0.29	0.13
Microsoft	51.61	0.77	0.30	0.09
Whisper	63.28	0.83	0.22	0.01

Table 6.7: Average BLEU, ROUGE, WER and SemDist performance scores for a fixed-window segmentation with 10 s windows and 5 s overlap on the ZHAW (lowercase), ZHAW (multi-sentence), Microsoft and Whisper model.

The fixed-window based segmentation shows (Table 6.7) that the ZHAW (lowercase) model performs the worst on all performance metrics while Whisper reaches the best scores across all performance metrics, even exceeding the sentence-based segmentation.

6.7.4 Conclusion

The objective of this experiment was to figure out how well a fixed-window segmentation works compared to a sentence-based segmentation, which is considered ideal because the models are trained on sentences containing a complete sentence.

The results obtained from the sentence-based segmentation align with the results from the VARLEN experiment (6.1) in which one sentence segment have been tested. The results from the fixed-window segmentation show that a nearly similar score can be reached. Except for the ZHAW (lowercase) model performed poorly on the fixed-window segmentation possibly due to the lack of incomplete or multi-sentence segments used during training.

The Whisper model showed better performance on the fixed-window segmentation than on the sentence-based segmentation. This is again due to the way the model was trained. Whisper was trained on 30 second segments [1] and not on sentence-based segments. Interestingly the Microsoft sentence-based segmentation and the fixed-window segmentation performed almost identically. This suggests that the fixed-window size in addition to the Levenshtein Merger (Chapter 5.3) performs comparable to the sentence-based segmentation.

The hypothesis, that the performance of an ST model decreases if a fixed- window segmentation is used compared to a sentence-based segmentation, can be accepted, except for the Whisper model. However, to which degree the performance decreases depends on other factors such as that the model can handle incomplete sentences.

This experiment does not reflect real-life scenarios because each sentence is from a different speaker and there is always a clear separation between two sentences, such scenarios are evaluated in the RLLIFE experiment (Chapter 6.8).

6.8 RLLIFE - Real-Life Scenario

The previous experiments were conducted under tightly controlled scenarios, in which for example segments contained only complete sentences. However, this does not reflect real-life scenarios. In order to bridge this gap and obtain insights into the performance of the ST models, it is necessary to evaluate their effectiveness on real-life recordings. From the findings of previous experiments, it is expected that some of the models might not perform particularly well on real-life recordings. However, ideally, the findings of the NRSENT (Chapter 6.4) and MERGER (Chapter 6.7) should indicate how well a model performs on real-life recordings.

6.8.1 Hypothesis

Speech translation models that perform well on the NRSENT and MERGER experiments will perform comparably on real-life recordings.

6.8.2 Method

Real-life recordings are recordings that contain multiple sentences from one or more speakers. The sentences are coherent and mostly complete and clearly spoken. For this experiment, a recording from a weather report (Chapter 4.1) is used.

The original audio was segmented and transcribed by the ZHAW (lowercase), ZHAW (multi-sentence), FHNW (long), Microsoft (long) and Whisper models. The ZHAW (lowercase), ZHAW (multi-sentence) and Whisper models all used the same fixed-window segmentation and merging with a window size of 20 seconds and 5 seconds of overlap. The Microsoft and FHNW model used their proprietary segmentation and merging methods.

To evaluate the performance of the ST models the BLEU and ROUGE scores were calculated as well as the WER and SemDist.

6.8.3 Results

The BLEU, ROUGE, WER and SemDist performance measures were collected during the experiment.

Model	**BLEU**	**ROUGE**	**WER**	**SemDist**
ZHAW (lowercase)	35.75	0.49	0.80	0.28
ZHAW (multi-sentence)	39.75	0.70	0.39	0.16
FHNW	67.73	0.85	0.20	0.05
Microsoft	78.94	0.92	0.10	0.003
Whisper	74.24	0.88	0.15	0.06

Table 6.8: BLEU, ROUGE, WER and SemDist calculated for the transcript produced by the ZHAW (lowercase), ZHAW (multi-sentence), FHNW, Microsoft and Whisper models.

The results show (Table 6.8) that the ZHAW (lowercase) model performs the worst on all metrics. The ZHAW (multi-sentence) has a slightly higher BLEU and ROUGE score and a much better WER and SemDist than the ZHAW (lowercase) model. The FHNW model shows a good performance on all metrics. The Microsoft and Whisper model achieved a very high score on the BLEU and ROUGE metrics while also having a minimal WER and SemDist.

6.8.4 Conclusion

In this experiment, the models are tested on a real-life recording to see how well the results from the previous experiments translate into the real world. And more importantly to see if they are useful in scenarios that they most likely will be used in, creating transcripts from video or audio recordings.

The results show that while the ZHAW (multi-sentence) model performed very well in the MERGER experiment (Chapter 6.7) it does not perform well in this real-life scenario. Why this is happening is unclear but it indicates, that while it is arguably an improved version of the ZHAW (lowercase) model it is still missing a crucial piece to perform as well as the FHNW, Microsoft and Whisper model.

The result of the Whisper model also shows that the poor performance of the ZHAW (lowercase) and ZHAW (multi-sentence) models can not be attributed to the segmentation or merging.

The hypothesis that speech translation models that perform well in the NRSENT and MERGER experiments will perform comparably on real-life recordings, can not be accepted because while the ZHAW (multi-sentence) model performed well in the NRSENT and MERGER experiment it performed badly in this experiment.

In this experiment, only one real-life sample has been tested. To make a relevant observation a lot more samples would be needed which unfortunately are not as widely available as the datasets used in other experiments. Future experiments would either need to create more realistic scenarios with the existing datasets or collect more transcripts from real-life scenarios.

Chapter 7

Discussion and Outlook

Whether the duration of an audio segment has an influence or not was investigated in the experiments. The results provide valuable insights into the performance of various ST models under different conditions, as well as some suggestions on how to improve the results of these models.

The VARLEN experiment revealed that the ZHAW (lowercase), FHNW, and Microsoft models tended to perform worse on longer audio segments, while the Whisper model showed an improvement in performance with increasing audio segment duration. The PADDIN experiment demonstrated that adding padding before and after an audio segment had no impact on performance. In the SPEEED experiment, it was observed that changing the playback speed of the audio showed a decreasing performance. Slowing down audio had a worse effect compared to speeding it up. The FHNW model performed well across a range of speed adjustments. The NRSENT experiment indicated that the ZHAW (lowercase), FHNW, and Microsoft models performed poorly when multiple sentences appeared in an audio segment, but the performance remained consistent when using the version of the FHNW and Microsoft models that segmented the audio. The SENPAD experiment revealed that adding padding between two sentences had no effect on performance. Training a model on multi-sentence segments, as shown in the MULSEN experiment, improved performance on audio segments with more than one sentence while maintaining performance on single sentence segments. The MERGER experiment demonstrated that fixed-window segmentation yielded comparable results to a sentence-based segmentation for the Multi-Sent, Microsoft, and Whisper models, while the ZHAW (lowercase) model performed subpar. Lastly, the RLLIFE experiment highlighted that the findings from the ZHAW (multi-sentence) model did not directly translate to real-life recordings, with the FHNW model performing well and the Microsoft and Whisper models excelling in performance.

From these results, the following can be concluded:

(1) While audio length has an influence on the score it is not the decisive factor on the performance. Other factors include sentence count and playback speed.

(2) On the ZHAW (lowercase), FHNW and Microsoft model it is crucial to apply segmentation even on audio segments below 15 seconds if they contain more than one sentence.

(3) Increasing the playback speed by a factor of 1.25 can lead to results that are comparable to the ones without a playback speed adjustments.

(4) By training a model on multi-sentence segments the performance of an ST model can be improved in a scenario where segmentation of the input audio is required.

(5) A fixed-window segmentation approach with the Levenshtein Merger (Chapter 5.3) achieved comparable results to a sentence-based segmentation, which is considered ideal.

In the SPEEED experiments, the inference times of the ST models were not recorded. In future research, it would be interesting to see how much time it takes to speed up an audio file and transcribe it. Then compare it to transcribing with the original playback speed. If transcribing a sped up recording is faster it could be used to reduce resource consumption.

The RLLIFE experiment (Chapter 6.8) demonstrates that while the newly trained ZHAW (multi-sentence) model performed very well in the experiments, it does not mean that it works well on real-life recordings. Further testing on more real-life samples has to be conducted to manifest this point. There is potentially another missing piece to make ST models perform well in real-life scenarios that have yet to be discovered.

Acronyms

AI	Artificial Intelligence
ASR	Automatic Speech Recognition
BLEU	Bilingual Evaluation Understudy
CMA	Constrained Model Adaptation
CNN	Convolutional Neural Network
CTC	Connectionist Temporal Classification
FHNW	University of Applied Sciences Northwestern Switzerland
MT	Machine Translation
NLP	Natural Language Processing
NNLM	Natural Network Language Model
RNN	Recurrent Neural Network
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
SemDist	Semantic Distance
SST4SG	Speach-to-Text for Swiss-German
ST	Speech-Translation
STT	Speech-to-Text
VAD	Voice Activation Detection
VTLN	Vocal tract length normalization
WER	Word Error Rate
ZHAW	Zurich University of Applied Sciences

List of Figures

List of Tables

Bibliography

[1] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," 2022.

[2] "Language support - language and voice support for the speech service." ,URL: `https://www.microsoft.com/en-us/research/blog/microsoft-translator-now-translating-100-languages-and-counting/`, Oct 2021. [Accessed 06.06.2023].

[3] N. Hollenstein and N. Aepli, "Compilation of a Swiss German dialect corpus and its application to PoS tagging," in *Proceedings of the First Workshop on Applying NLP Tools to Similar Languages, Varieties and Dialects*, (Dublin, Ireland), pp. 85–94, Association for Computational Linguistics and Dublin City University, Aug. 2014.

[4] A. Magueresse, V. Carles, and E. Heetderks, "Low-resource languages: A review of past work and future challenges," 2020.

[5] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," 2020.

[6] N. Team, M. R. Costa-jussà, J. Cross, O. Çelebi, M. Elbayad, K. Heafield, K. Heffernan, E. Kalbassi, J. Lam, D. Licht, J. Maillard, A. Sun, S. Wang, G. Wenzek, A. Youngblood, B. Akula, L. Barrault, G. M. Gonzalez, P. Hansanti, J. Hoffman, S. Jarrett, K. R. Sadagopan, D. Rowe, S. Spruit, C. Tran, P. Andrews, N. F. Ayan, S. Bhosale, S. Edunov, A. Fan, C. Gao, V. Goswami, F. Guzmán, P. Koehn, A. Mourachko, C. Ropers, S. Saleem, H. Schwenk, and J. Wang, "No language left behind: Scaling human-centered machine translation," 2022.

[7] L. Miao, J. Wu, P. Behre, S. Chang, and S. Parthasarathy, "Multilingual transformer language model for speech recognition in low-resource languages," 2022.

[8] V. K. Rangarajan Sridhar, J. Chen, S. Bangalore, A. Ljolje, and R. Chengalvarayan, "Segmentation strategies for streaming speech translation," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Atlanta, Georgia), pp. 230–238, Association for Computational Linguistics, June 2013.

[9] C. Amrhein and B. Haddow, "Don't discard fixed-window audio segmentation in speech-to-text translation," 2022.

[10] I. Tsiamas, G. I. Gállego, J. A. R. Fonollosa, and M. R. Costa-jussà, "SHAS: Approaching optimal Segmentation for End-to-End Speech Translation," in *Proc. Interspeech 2022*, pp. 106–110, 2022.

[11] M. Gaido, M. Negri, M. Cettolo, and M. Turchi, "Beyond voice activity detection: Hybrid audio segmentation for direct speech translation," 2021.

[12] T. Potapczyk and P. Przybysz, "SRPOL's system for the IWSLT 2020 end-to-end speech translation task," in *Proceedings of the 17th International Conference on Spoken Language Translation*, (Online), pp. 89–94, Association for Computational Linguistics, July 2020.

[13] W. B. Cavnar, J. M. Trenkle, *et al.*, "N-gram-based text categorization," in *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, vol. 161175, Las Vegas, NV, 1994.

[14] M. Ogihara and T. Li, "N-gram chord profiles for composer style representation.," in *ISMIR*, pp. 671–676, Citeseer, 2008.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[17] S. Baccianella, A. Esuli, F. Sebastiani, *et al.*, "Sentiwordnet 3.0: an enhanced lexical resource for sentiment analysis and opinion mining.," in *Lrec*, vol. 10, pp. 2200–2204, 2010.

[18] A. Dey, M. Jenamani, and J. J. Thakkar, "Senti-n-gram: An n-gram lexicon for sentiment analysis," *Expert Systems with Applications*, vol. 103, pp. 92–105, 2018.

[19] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, "A survey of large language models," 2023.

[20] F. Jelinek, *Statistical methods for speech recognition.* MIT press, 1998.

[21] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *Advances in Neural Information Processing Systems* (T. Leen, T. Dietterich, and V. Tresp, eds.), vol. 13, MIT Press, 2000.

[22] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018.

[23] OpenAI, "Gpt-4 technical report," 2023.

[24] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[25] D. Jurafsky and J. H. Martin, *Speech and Language Processing (3rd ed. draft).* Jan 2023.

[26] R. Shadiev, W.-Y. Hwang, N.-S. Chen, and Y.-M. Huang, "Review of speech-to-text recognition technology for enhancing learning," *Journal of Educational Technology Society*, vol. 17, no. 4, pp. 65–84, 2014.

[27] R. Duan, H. Wu, and R. Zhou, "Faster matrix multiplication via asymmetric hashing," 2023.

[28] C. DAVENPORT, "Google translate processes 143 billion words every days," *Android Police*, Oct. 2018.

[29] *Sprachenlandschaft in der Schweiz.* No. 23164427, Neuchâtel: Bundesamt für Statistik (BFS), Sep 2022.

[30] "recapp.ch: Automatische transkription." URL: `https://recapp.ch/`. [Accessed 06.06.2023].

[31] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," ch. 10.8, 2023.

[32] C.-M. U. S. Dept., "Speech understanding systems: summary of results of the five-year research effort at Carnegie-Mellon University.," 4 2015.

[33] A. Graves, "Sequence transduction with recurrent neural networks," 2012.

[34] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba, "Addressing the rare word problem in neural machine translation," 2015.

[35] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, (New York, NY, USA), p. 369–376, Association for Computing Machinery, 2006.

[36] A. Hannun, "Sequence modeling with ctc," *Distill*, 2017. https://distill.pub/2017/ctc.

[37] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," 2019.

[38] A. Morris, V. Maier, and P. Green, "From wer and ril to mer and wil: improved evaluation measures for connected speech recognition.," 01 2004.

[39] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, (USA), p. 311–318, Association for Computational Linguistics, 2002.

[40] H. Ji, "Test-of-Time Award Papers." URL: `https://naacl2018.wordpress.com/2018/03/22/test-of-time-award-papers/`, Mar. 2018. [Accessed 02.06.2023].

[41] "Modelle bewerten | AutoML Translation-Dokumentation." URL: `https://cloud.google.com/translate/automl/docs/evaluate`. [Accessed 01.06.2023].

[42] M. Post, "A call for clarity in reporting BLEU scores," in *Proceedings of the Third Conference on Machine Translation: Research Papers*, (Belgium, Brussels), pp. 186–191, Association for Computational Linguistics, Oct. 2018.

[43] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, (Barcelona, Spain), pp. 74–81, Association for Computational Linguistics, July 2004.

[44] Y. Li, D. McLean, Z. Bandar, J. O'Shea, and K. Crockett, "Sentence similarity based on semantic nets and corpus statistics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 8, pp. 1138–1150, 2006.

[45] Y. Schraner, C. Scheller, M. Plüss, and M. Vogel, "Swiss german speech to text system evaluation," 2022.

[46] M. Plüss, J. Deriu, Y. Schraner, C. Paonessa, J. Hartmann, L. Schmidt, C. Scheller, M. Hürlimann, T. Samardžić, M. Vogel, and M. Cieliebak, "Stt4sg-350: A speech corpus for all swiss german dialect regions," 2023.

[47] M. Plüss, L. Neukom, C. Scheller, and M. Vogel, "Swiss parliaments corpus, an automatically aligned swiss german speech to standard german text corpus," 2021.

[48] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," in *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*, pp. 4211–4215, 2020.

[49] K. D. Mohan and J. Skotdal, "Microsoft translator: Now translating 100 languages and counting!," Oct 2021.

[50] E. Urban, "Language support - speech service - azure cognitive services." https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/language-support?tabs=sttsupported-languages, Feb 2023. [Accessed 08.06.2023].

[51] E. Urban, "Speech translation quickstart - speech service - azure cognitive services." https://learn.microsoft.com/en-us/azure/cognitive-services/speech-service/get-started-speech-translation?source=recommendationsamp;tabs=windows%5C%2Cterminalamp;pivots=programming-language-python, Sep 2023. [Accessed 08.06.2023].

[52] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 11 2019.

[53] D. Prasetya, A. Wibawa, and T. Hirashima, "The performance of text similarity algorithms," *International Journal of Advances in Intelligent Informatics*, vol. 4, 05 2018.

[54] V. I. Levenshtein *et al.*, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, vol. 10, pp. 707–710, Soviet Union, 1966.

Appendices

Appendix A

Transcripts

A.1 TeleZüri Reference Transcript

Guten Abend miteinander. Ich bin leider kein Komiker, kein Wetterschmöcker und schon gar kein Zoodirektor. Dafür kann ich ihnen als Miss Schweiz etwas darüber erzählen, wie sie auch bei schlechtem Wetter gestylt durch den Tag kommen. Heute Abend müssen Sie sich diesbezüglich keine Sorgen machen, denn in der Nacht kommen Sternchen hervor, die Wolken verziehen sich. Morgen Vormittag ist es dann abgesehen von ein bisschen Nebel und Wolken recht sonnig. Temperaturen morgen morgen nur bei 5 - 7 Grad, am besten legen Sie schon mal den Strick Pullover bereit, weil die sind nämlich im Moment sowieso der letzte Schrei. Am Nachmittag geht es sonnig weiter, es gibt nur ein paar Wolken. Die Temperaturen klettern bis auf 18 Grad, dann liegt für die coolen vielleicht sogar der t Shirt look drinnen. 0 Grad Grenze liegt morgen auf 3500 Meter tshirt Look funktioniert dort halt nur für dich hart Gesottene. Am Samstagmorgen kann es im Flachland Nebel haben, sonst ist es strahlend schön, unbedingt die Sonnenbrille einpacken. Am Sonntag ist erst Nebel möglich, es geht aber mit viel Sonnenschein weiter. Ab und zu ziehen ein paar Wolkenfelder vorbei. Der Wochenstart ist recht sonnig und warm, ab dem Dienstag tut es leider wieder zu. Am Mittwoch ist es bewölkt und nass, zum Glück sind aber Regenstiefel immer noch total im Trend. So das war es mit dem Wetter, morgen an dieser Stelle die beiden Nationalräte Daniel Jositsch und Christof Mörgeli. Vermutlich gibt es dann Hagel von links und Gewitter von rechts. Einen schönen Abend miteinander.

A.2 RLLIFE ZHAW (lowercase) Transcript

guten abend miteinander ich bin leider kein komiker keine wetter schmecker und schon gar kenzenundirektedafür kann ich sind nämlich moment sowieso el letzten schrei am nachmittag an sonnig weiter gibt nur ein paar wochen temperaturen klettern bis auf achtzehn grad tag für die coolen vielleicht sogar den tschartloktrei null grad grenze liegt morgen auf 350 meter der t sat lok funktioniert im flachland in im flachland in nebel sonst ist strahlend schön und bringt sonnen sie einpacken am sonntag ich ihnen als miss schweiz etwas darüber erzählen wir sie auch bei zuerst schnebel mögliches geht aber mit viel sonnenschein weiter in und a sind ein paar wolkenfelder vorbei den wochen start ist wochenstart ist recht sonnig und warm ab dienstag trotzden leider wieder zu auch der mittwoch ist bewölkt und nas zum glück sind aber regenstiefel immer noch total soderswert mit mit dem wetter morgen an dieser stelle die beiden nationalräte daniel josic und christoph mergeln vermutlich gibt es den hagel von links und gewitter von auch bei schlechtem wetter gesellt mit sich diesbezüglich keine sorgen machen weil in der nacht kommen sternde für ein verziehe sch morgen vormittag ist es dann abgese von nebel sieben grad am besten legen sie schon mal den streckpoliperat weil die sind nämlich aeiomeneodser

A.3 RLLIFE ZHAW (multi-sentence) Transcript

Guten Abend mittichen leider kein Komiker keine wetter schmöcker und schon gar keinen Dafür kann ich Ihnen als Miss Schweiter etwas darüber erzählen, wie sie auch bei schlechtem Wetter eilt durch den Tag . Heute Abend muss diesbezüglich keine Sorgen machen, weil in der Nacht kommen Sternchen für eine Wolke verziehen. Morgen Vormittag ist es dann abgesehen, von Nebel und Wolkenrecht, Temperaturen, messen Morgen nur bei fünf G sieben Grad, am besten legen sie schon Strekttolleparat, weil diese nämlich im Moment sowieso delle Schrei. Am Nachmittag geht Sonic weiter gibt nur ein paar Wochen Temperaturen klettern bis auf achtzehn Grad dann liegt für die coolen vielleicht sogar den Null Grad Grenze morgen auf Fr. 3'500 Teshetlokfunktionität halt nur für zur . Am Samstagmorgen kann es im Flachland Nebela sonst ist eine Strahlen und bringt Sonnenbrille einpacken. Am Sonntag ist ehrschnbenmöglich, geht aber mit er schnell möglich, geht aber mit viel Sonnenschein weiter, ab und zu sind ein paar Wochen felder vorbei. Der Wochen startische Recht, sonnig und warm ab Dienstag trotzdem leider wieder zu. Auch der Mittwoch ist bewölkt und nass zum Glück sind aber Regenstiefel immer noch total en. Sondern wärs mit dem Wetter mon an dieser Stelle die beiden Nationalräte Daniel Josig und Christoph Mörgeli, vermutlich gibt es dann handel Links und Gewitter von rechts einen schönen am

A.4 RLLIFE FHNW Transcript

Guten Abend, Miteinander, ich bin leider kein Komiker, kein Wetterschmecker und schon gar kein Zoo-Direktor, dafür kann ich Ihnen als Miss-Schweiz etwas darüber erzählen, wie sie auch bei schlechtem Wetter gesteilt durch den Tag kommen. Heute Abend müssen Sie sich diesbezüglich keine Sorgen machen, denn in der Nacht kommen die Sterne nach vorne, die Wolken verziehen sich. Morgen Vormittag ist es abgesehen von etwas Nebel und Wolken recht sonnig, die Temperaturen morgen Vormittag nur bei 5 bis 7 Grad, am besten legen sie den Strickpuli , denn diese sind im Moment sowieso, der letzte Schrei. Am Nachmittag geht es sonnig weiter, es gibt nur ein paar Wolken, die Temperaturen klettern bis auf 18 Grad, dann liegt für die coolen vielleicht sogar der Te-Shirt-Look drin. Die Nullgradgrenze liegt morgen auf 3500 Meter, der T-Shirt-Lok funktioniert nur für die hart gestoten. Am Samstagmorgen kann es im Flachland Nebel haben, sonst ist es strahlend schön, unbedingt die Sonnenbrille einpacken. Am Sonntag ist zuerst Nebel möglich, es geht aber mit viel Sonnenschein weiter, ab und zu ziehen ein paar Wolkenfelder vorbei, der Wochenstart ist recht sonnig und warm, ab Dienstag wird es leider wieder geschlossen. Auch der Mittwoch ist bewölkt und nass, zum Glück sind aber Regenstiefels immer noch total inne. So, das wäre es mit dem Wetter, morgen an dieser Stelle die beiden Nationalräte, Daniel Josetsch und Christoph Mörgeli, vermutlich gibt es dann Hagel von links und von rechts einen schönen Abend.

A.5 RLLIFE Microsoft Transcript

Guten Abend miteinander, ich bin leider kein Komiker, kein Wetterschmöcker und schon gar kein Zoodirektor, dafür kann ich ihnen als Miss Schweiz etwas darüber erzählen, wie sie auch bei schlechtem Wetter geteilt durch den Tag kommen. Heute Abend müssen Sie sich diesbezüglich keine Sorgen machen, denn in der Nacht kommen Sternchen hervor. Die Wolken verziehen sich. Morgen Vormittag ist das denn abgesehen von ein bisschen Nebel und Wolken recht sonnig. Temperaturen morgen morgen nur bei 5 bis 7 Grad. Am besten legen Sie schon mal den Strick Pullover bereit. Weil die sind nämlich im Moment sowieso der letzte Schrei. Am Nachmittag geht es sonnig weiter, es gibt nur paar Wolken, Temperaturen, klettern bis auf 18 Grad, dann liegt für die coolen, vielleicht sogar den t Shirt, lockt Rehm die 0 Grad Grenze liegt morgen auf 3500 Meter tshirt Look funktioniert dort halt nur für dich hart gesottene am Samstagmorgen kann im Flachland Nebel haben, sonst ist es strahlend schön, unbedingt Sonnenbrille einpacken am Sonntag ist zuerst Nebel möglich. Es geht aber mit viel Sonnenschein weiter. Ab und zu ein paar Wolkenfelder vorbei. Der Wochenstart ist recht sonnig und warm, ab Dienstag wird es dann leider wieder zu. Auch Mittwoch ist bewölkt und nass, zum Glück sind aber regenstiefel immer noch total an so, das wärs gewesen mit dem Wetter morgen an dieser Stelle die beiden Nationalräte Daniel Jositsch und Christoph Mörgeli. Vermutlich gibt dann Hagel von links und Wetter von rechts einen schönen Abend miteinander.

A.6 RLLIFE Whisper Transcript

Guten Abend miteinander. Ich bin leider kein Komiker, kein Wetterschmöcker und schon gar kein Zoodirekter. Dafür kann ich Ihnen als Miss Schweiz etwas darüber erzählen, wie Sie auch bei schlechtem Wetter durch den Tag kommen. Heute Abend müssen Sie sich keine Sorgen machen, denn in der Nacht kommen die Sternchen, die Wolken verziehen sich. Morgen Vormittag ist es dann abgesehen von Nebel und Wolken recht sonnig. Temperaturen morgen Morgen nur bei 5 bis 7 Grad. Am besten legen Sie schon mal den Streckpulli bereit, denn die sind im Moment sowieso der letzte Schrei. Am Nachmittag geht es sonnig weiter, es gibt nur ein paar Wolken. Die Temperaturen klettern bis auf 18 Grad, dann liegt für die coolen vielleicht sogar der T-Shirt-Look drin. Die Null-Grad-Grenze liegt morgen auf 3500m. Der T-Shirt-Look funktioniert dort nur für die Härtgesottenen. Am Samstagmorgen kann es im Flachland Nebel haben, sonst ist es strahlend schön. Unbedingt die Sonnenbrille einpacken! Am Sonntag ist zuerst Nebel möglich, es geht aber mit viel Sonnenschein weiter. Ab und zu ziehen ein paar Wolkenfelder vorbei. Der Wochenstart ist recht sonnig und warm, aber am Dienstag tut es dann leider wieder zu. Auch der Mittwoch ist bewölkt und nass. Zum Glück sind aber Regenstiefel immer noch total in. So, das wäre es gewesen mit dem Wetter. Morgen an dieser Stelle die beiden Nationalräte Daniel Josic und Christoph Mörgeli. Vermutlich gibt es dann Hagel von links und Gewitter von rechts. Einen schönen Mittenand.

Appendix B

Code & Manual

All code required to run the experiments is available on github.zhaw.ch: `https://github.zhaw.ch/ba-vandenik-saarofel/exp_pipeline`

B.1 Experiment: Pipeline

B.1.1 Setup

1. Install python and pip

   ```
   apt update
   apt install python3-pip cmake libsndfile1 ffmpeg
   ```

2. Install requirements

   ```
   pip install -r ./requirements.txt
   ```

3. Create a `.env` file containing the following variables:

- `SPEECH_KEY` and `SPEECH_REGION` are used for the Microsoft transcriber. They can be obtained from the Azure Portal after creating a **Speech service** resource.
- `HF_TOKEN` is used when the `speaker_diarization` segmentation is active. During the segmentation a model is downloaded from HuggingFace. The token can be obtained on the profile page of your HF account
- `SUPABASE_URL` and `SUPABASE_KEY` are used for the `SupabaseReporter` to store the result in a supabase[1] instance. Check the Supabase section on how to set it up correct.

4. Set the python path in the `start_run.sh` to use the correct python environment
5. Optionally: To use supabase as a datastore for the experiments the database should be initalized using the `./supabase/init.sql` script and the `./supabase/stored-procedures.sql` script.
6. Optionally: To view the results in a web application the Vox Visor (https://github.com/MrF3lix/vox-visor) can be used.

[1]`https://supabase.com/`

B.1.2 How to run the pipeline

1. Check the configuration and set it according to your needs.
2. Add the audio input files and the reference transcripts to the `path_to_wavs` folder configured in the `preprocessor_config.json` file (`./data` by default).
3. Run `./start_run.sh` from the console to run the pipeline in the background.
4. Check the `./output.log` to see the pipelines console output.

To monitor the pipeline process more closely, the `./pid.log` file contains its process id.

If something goes wrong during a pipeline step the run can be stopped with the `./stop_run.sh` script. To resume the pipeline, uncomment the completed pipeline steps in the `pipeline.py` file and start the pipeline again with the `./start_run.sh` script.

B.1.3 Configuration

The configuration files in the `/config` folder provide a way to customize the pipeline and set the parameters used by each pipeline step.

The config json files are loaded into python dataclasses. Each configuration dataclass object contains all configurations needed for a pipeline step. Described in this chapter are the most relevant attributes, and their default value.

The detailed configuration allows for the possibility of each pipeline step to be run on its own and with great controll given to the user.

`pipeline_config.json`

```json
{
    "segmenter_class_id": "<Segmenter Class>",
    "transcriber_class_id": "<Transcriber Class>",
    "merger_class_id": "<Merger Class>"
}
```

The values `<Segmenter Class>`, `<Transcriber Class>` and `<Merger Class>` must match the string defined in the `class_id` variable in each implementation of a pipeline step.

Options for the `segmenter_class_id` include:

- `single_segment`
- `set_size`
- `pre_segmented`
- `shas`
- `speaker_diarization`

Options for the `transcriber_class_id` include:

- `zhaw`
- `fhnw`
- `microsoft`
- `whisper`

Options for the `merger_class_id` include:

- `naive`
- `levenshtein`

`preprocessor_config.json`

```
{
    "path_to_wavs": "./data",
    "path_to_output": "./results/data/",
    "sample_rate": 16000
}
```

- `path_to_wavs` sets the path to the input files.
- `path_to_output` sets the path in which the files gets stored after the prepocessing. It should be consistent with the `path_to_wavs` configuration from the `segmenter_config.json`.
- `sample_rate` sets the sample rate which the segments are set to. The models all require a SR of 16'000 Hz to work properly.

`segmenter_config.json`

```
{
    "path_to_checkpoint": "./models/sfc/mult_sfc_model_epoch-4.pt",
    "path_to_segmentations": "./results/segmenter/",
    "path_to_wavs": "./results/data/",
    "pre_segmented_path_to_segmentations": "./data/",
    "set_size_window_size_s": 30,
    "set_size_overlap_size_s": 10
}
```

- `path_to_checkpoint` is used for the `shas` segmenter. It references a model to automatically segment the audio files.
- `path_to_segmentations` is the output path of the segmentation step.
- `path_to_wavs` is the path to the .wav files to be segmented.
- `pre_segmented_path_to_segmentations` is the path in which the `pre-segmenter` looks for the manually created segmentations.
- `set_size_window_size_s` is configuration for the fixed-window segmentation. It defines how large the windows are in seconds.
- `set_size_overlap_size_s` is configuration for the fixed-window segmentation. It defines how large the overlap between the windows is in seconds.

`cutter_config.json`

```
{
    "path_to_segmentations": "./results/segmenter/",
    "path_to_cut_segments": "./results/cutter/",
    "path_to_wavs": "./results/data/",
}
```

- `path_to_segmentations` is the path the segmentations generated by the segmenter.
- `path_to_cut_segments` is the path to where the cut audio files are created.
- `path_to_wavs` is the path to the .wav files to be cut.

`transcriber_config.json`

```
{
    "path_to_checkpoint": "./models/ch_mix/multi-sent/checkpoint-21000",
    "path_to_vocab": "./models/ch_mix/multi-sent/checkpoint-21000/vocab.json",
    "path_to_kenlm_model": "./models/language-model/full.binary",
    "path_to_wavs": "./results/cutter/",
    "path_to_output": "./results/transcriber/",
    "device": "cuda",
    "ms_long_audio": true,
    "fhnw_long_audio": false,
    "fhnw_delay_s": 10
}
```

- `path_to_checkpoint` sets the path to the ZHAW model folder.
- `path_to_vocab` sets the path to vocabulary.
- `path_to_kenlm_model` path to the kenlm language model.
- `path_to_wavs` is the path to the .wav files to be transcribed.
- `path_to_output` is the path to where the generated transcription files are created.
- `device` is the device used to run language models (CPU / GPU).
- `ms_long_audio` defines if the version of microsoft is used that automatically segments the input. Should be set to true if the input files are longer than 10 seconds.
- `fhnw_long_audio` defines if the version of fhnw is used that automatically segments the input. Should be set to true if the input files are longer than 15 seconds.
- `fhnw_delay_s` is the delay in seconds in between each call to the FHNW API.

`merger_config.json`

```
{
    "path_to_input": "./results/transcriber/",
    "path_to_output": "./results/merger/",
    "levenshtein_edge_words_to_remove": 2
}
```

- `path_to_input` is the path to the transcript files generated by the transcriber.
- `path_to_output` is the output path to where the merged transcripts are created.
- `levenshtein_edge_words_to_remove` is the number of words that are removed at the edges of a transcription when using the levenshtein merger.

`scorer_config.json`

```
{
    "path_to_merged_output": "./results/merger/",
    "path_to_ground_truth": "./results/data/",
    "path_to_score_output": "./results/scorer/"
}
```

- `path_to_merged_output` is the path to the merged transcript files generated by the merger.
- `path_to_ground_truth` is the folder in which the ground truth references are stored.
- `path_to_score_output` is the path to where the generated score files are created.

`reporter_config.json`

```
{
    "experiment_id": "67d912f0-6ab0-44bf-ac6b-d9cf4684e90f",
    "path_to_references" : "./results/data/",
    "path_to_scores": "./results/scorer/"
}
```

- `experiment_id` should be set if the `supabase` reporter is used. It defines to which experiment the results should be added.
- `path_to_references` is the folder in which the ground truth references are stored.
- `path_to_scores` is the folder in which the score files generated by the scorer are stored.