Utrecht University

Improving the Automatic Speech Recognition Model Whisper with Voice Activity Detection

by

Isa Dielen

to obtain the degree of Master of Science
in Applied Data Science
at Utrecht University.
In cooperation with the Dutch National Police

Daily Supervisors
D.S. Islakoglu (Utrecht University)
N. Hulzebosch (External)

Examiners
Dr. I.R. Karnstedt-Hulpus (first)
V.J. Shahrivari (second)

Student Number: 9481141

PREFACE

Given my academic background in Criminology and Applied Data Sciences, optimising an Automatic Speech Recognition (ASR) model for the Dutch National Police proved to be an excellent match. This project covers a topic that greatly intrigues me and aligns with my intended professional path within this specific domain. The learning curve that I went through in a relatively short time frame of just 10 weeks is immense, and I am very thankful the department TROI for giving me this opportunity. This project marks the end of my academic career, and the past year at Utrecht University have inspired me and created a lot of memorable memories that I will prosper for the rest of my life.

Throughout this research, I have further developed my programming skills in Python and gained a lot of knowledge regarding Automatic Speech Recognition models, and the advantageous the pre-processing technique voice activity detection can bring to such models. From the start of my thesis, I was given a lot of freedom to explore different opportunities, which has helped me in defining the scope of this study, setting boundaries, and has sparked my intrinsic motivation to make this project a success.

However, of course, I could not have completed this research on my own. Therefore, I would like to express my appreciation to my daily supervisors from the Utrecht University Duygu Islakoglu and Vahid Shahrivari for providing me with expert knowledge and for their guidance throughout this research. The weekly meetings in which they asked critical questions really helped me in improving the quality of my work. They were always very approachable, and very quick in offering help. Additionally, I would like to extend my biggest gratitude to the external supervisor Nils Hulzebosch, since the comprehensiveness of this research could not have been possible without his assistance. Nils debugged one of the tested algorithms in this study for almost 1.5 week, after which I was able to extend the number of experiments in my research and provide a more complete story on the performance of different ASR models. Additionally, I would like to thank Hilde Spits for giving me the opportunity to do my graduate project at TROI Amsterdam.

Last but not least, I would also like to thank Dr. Ioana Karnstedt-Hulpus for evaluating our poster presentation and my final thesis as the first examiner, and Vahid Shahrivari as the second examiner. I am very much looking forward to your evaluation!

Isa Dielen
July 7, 2023
Amsterdam

ABSTRACT

The Dutch National Police possesses a substantial amount of audio data, and transcribing these audio files manually for analysis purposes is very labour intensive. That is why the department TROI developed an application that uses Whisper, a state-of-the-art Automatic Speech Recognition model, to automatically transcribe these audio files. Although the accuracy of Whisper is quite high, its execution time is relatively slow, posing a challenge when needing to transcribe large amounts of audio files. Considering Whisper's minimalist strategy for data pre-processing, it is conceivable that incorporating advanced pre-processing techniques could further optimize its performance in terms of running time, without a considerable comprise in accuracy. Therefore, this research aims to investigate the influence of the pre-processing method voice activity detection on Whisper's performance.

The experiments in this research compare the performance of Whisper, Faster-Whisper and WhisperX on Dutch long-form audio data of two datasets: CGN and NFI-FRITS. Faster-Whisper and WhisperX both incorporate a voice activity detection model, Silero VAD and PyAnnote VAD, respectively. Additional experiments with hyperparameter tuning and testing the voice activity detection models are conducted. The evaluation metrics used in this research are Word Error Rate, precision, recall, F1 score and Real-Time Factor.

The results demonstrate that both Faster-Whisper and WhisperX outperform the baseline Whisper model. They exhibit improved WER, precision, recall, F1-score, and RTF, indicating the advantages of incorporating voice activity detection models within the Whisper framework. Generally, WhisperX is outperforming Faster-Whisper across the different datasets and settings, on almost all performance metrics. Furthermore, the effects of tuning speech probability thresholds in Faster-Whisper and WhisperX are not clear, as they do not show a specific trend. The comparison between Silero and PyAnnote VAD shows variations in precision and recall, where PyAnnote VAD, as incorporated in WhisperX, is outperforming on precision, and Silero VAD, as incorporated in Faster-Whisper, is outperforming on recall.

In conclusion, incorporating a voice activity detection model as a pre-processing technique enhances the performance of Whisper by improving transcription accuracy measured by the Word-Error Rate, precision, recall, and F1-score, and reducing the execution time measured by the Real-Time Factor. However, the effects of tuning the speech probability threshold combined with Faster-Whisper and WhisperX, are limited. Future research is recommended to examine alternative voice activity detection models, test the memory usage of the models, investigate differences between Whisper implementations, and look into alternative pre-processing techniques.

The code is available on this Github page: https://github.com/isadielen/MSc_WhisperVAD.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ASR	Automatic Speech Recognition
CGN	Corpus Gesproken Nederlands
FN	False negative
FP	False positive
NFI-FRITS	Netherlands Forensic Institute's Forensically Realistic Intercepted Telephone Speech
RTF	Real-time factor
TROI	Team Rendement Operationale Informatie
TP	True positive
VAD	Voice Activity Detection
WER	Word Error Rate

1 INTRODUCTION

In recent years, there has been a growing interest in automatic speech recognition (ASR) systems, with its influence extending to various domains such as health, education, and robotics [1]. Within speech science and engineering, speech recognition is one of the fastest developing fields, driven by the human desire to automate simple tasks that require human-machine interactions [2, 3], such as creating transcripts of audio files. ASR refers to computer and software-based techniques that automatically convert spoken words into text [4]. With the rapid progress of technologies, ASR systems are incorporated within multiple applications due to their functionality and user-friendly nature [1].

Whisper [5] is a state-of-the-art and innovative Automatic Speech Recognition (ASR) model developed by OpenAI[1] and released in September 2022. It consists of an encoder-decoder transformer architecture [6]. The model's efficacy is rooted in its extensive training on an impressive dataset of 680,000 hours of multilingual audio of speech [7, 8]. This dataset includes a vast 540,000 hours of English speech and a substantial 117,000 hours spanned across 96 diverse languages [9]. The rich, multifaceted training data helps Whisper's ability in rendering precise textual transcriptions and illustrates the significant, yet often underestimated, potential of scaling supervised pre-training in the realm of speech recognition [5]. In addition to Whisper, multiple re-implementations are developed including Faster-Whisper[2], WhisperX[3], and Whisper-Jax.[4]

The Criminal Investigation Department of the Dutch National Police among others collects and analyses audio data for the purpose of detecting criminal offenses. This department possesses a substantial amount of audio data, including seized data. Transcribing these audio files manually for analysis purposes is very labour-intensive, and that is why the department TROI[5] of the Dutch National Police is working on ways to convert the speech to text automatically using an ASR system, enabling detectives to quickly search for relevant information in audio files.

The Dutch National Police are unable to employ ASR models that run on cloud platforms since police data is too sensitive to send to the cloud. Therefore, an ASR model must be able to run locally in a closed environment. Prior to the launch of Whisper, there existed alternative open-source models such as Wav2Vec [10] which operated on local systems; however, the level of quality provided by these models was significantly lower compared to that of Whisper [5].

Although the accuracy of Whisper is quite high, the problem of this model lies in its extensive execution time. The execution time is roughly 1.5 times real-time, meaning that the execution

1 https://openai.com/research/whisper
2 https://github.com/guillaumekln/faster-whisper
3 https://github.com/m-bain/whisperX
4 https://github.com/sanchit-gandhi/whisper-jax
5 TROI (Team Rendement Operationele Informatie) is a specialized department employing experts in data science, data analysis, and software development. TROI aims to facilitate applications for the analysis of large volumes of data, enabling detectives to effectively work through this information. For instance, one of their applications makes it possible to find specific concepts such as weapons in images.

time of Whisper for an audio file of 9 hours is 6 hours. This protracted duration poses an inconvenience to the application, especially given the large volume of audio files. Therefore, TROI is looking for ways to optimize this algorithm in terms of speed without compromising on its accuracy.

Considering the minimalist strategy for data pre-processing employed by Whisper [5], it is conceivable that incorporating advanced pre-processing techniques could further optimize its performance. This potential enhancement could expedite the execution time of Whisper without a considerable compromise in accuracy. Despite the extensive research conducted on ASR models in general, little to no research has been focused specifically on Whisper, mainly due to its novelty. Additionally, no prior research has been focused on comparing the performance of three distinct implementations, Whisper, Faster-Whisper, and WhisperX on Dutch long-form audio data. Accordingly, a more thorough understanding of Whisper's performance, or a re-implementation that incorporates a pre-processing technique on Dutch long-form audio data, will fill this research gap and could ultimately lead to an advice for the Dutch National Police on which implementation and pre-processing techniques to use.

1.1 RESEARCH OBJECTIVES

The main objective of the underlying research is to compare the performance of the baseline model Whisper to (1) Faster-Whisper and (2) WhisperX on Dutch long-form audio data. The performance will be evaluated based on the following metrics: Word Error Rate, precision, recall, F1-score, and speed in terms of Real-Time Factor, which will be defined in Chapter 2.3. Therefore, the main research question is the following:

> *How do state-of-the-art enhanced Whisper models including the pre-processing technique voice activity detection perform compared to the baseline model Whisper on Dutch long-form audio in terms of Word Error Rate, precision, recall, F1-score and Real-Time Factor?*

1.2 THESIS OUTLINE

This research exists of six chapters and the content is organized as follows. First, Chapter 2 provides background information on ASR systems and how these systems can be evaluated. Additionally, an overview of the current research on Whisper and its re-implementations is provided. Chapter 3 discusses the data used in this research, including the steps taken for data preparation, as well as the ethical and legal considerations associated with the data. Subsequently, in Chapter 4, the methodology used in this work is elaborated on and the different steps required to reach the research objectives are laid out. Chapter 5 presents the main results of Whisper and its re-implementations on the different performance metrics. Furthermore, Chapter 6 discusses the findings as presented in the aforementioned results chapter, while also explaining certain limitations inherent in this research. Finally, Chapter 7 sets out the main conclusions and provides an answer to the research question. Moreover, suggestions are made for possible future research.

2 BACKGROUND & RELATED WORK

The purpose of this chapter is to provide background information on ASR systems and discuss metrics that can be used to evaluate the performance of such a system. Additionally, this chapter reviews the literature available on Whisper and two re-implementations called Faster-Whisper and WhisperX. First, Section 2.1 briefly explains the four different modules in a basic ASR system, followed by Section 2.2 which explains voice activity detection as a pre-processing techniques. Section 2.3 elaborates on five different metrics that can be used to evaluate the performance of an ASR system. Finally, Section 2.4 reviews the available literature on Whisper and two re-implementations, Faster-Whisper and WhisperX.

2.1 ASR SYSTEMS

In this section, a brief discussion will be provided for each module of an ASR system. An ASR system allows a computer to take an audio file or direct speech from a microphone as input and convert it into text. An ideal ASR is capable of perceiving the given input and identifying the spoken words in the input accurately [11]. An ASR system consists of different modules: a pre-processing module, a feature extraction module, a classification model, and a language model. The specific architecture of an ASR system is determined by the choice of models employed within each module [1]. Figure 2.1.1 represents a basic structure of an ASR system. The following paragraphs will explain the modules in more detail.

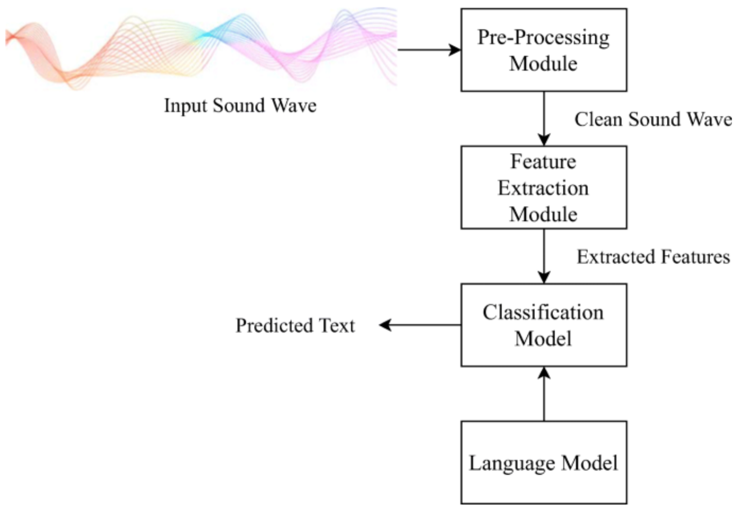


Figure 2.1.1: Basic structure of an ASR [11]

Pre-processing is the first and most important phase in developing an efficient ASR system [2, 12, 13]. Pre-processing techniques modify the input speech signal so that it becomes more suitable for feature extraction analysis, i.e., the next module of the ASR system. The goal of pre-processing a speech signal is to make an ASR system computationally more efficient and enhance the accuracy [2]. There are several pre-processing techniques according to literature, and some of the most mentioned techniques are (1) Voice Activity Detection, (2) Noise

Removal (i.e., denoising), (3) Pre-emphasis, (4) Framing, (5) Windowing and (6) Normalization. The work of [12] states that this is also the order in which a speech signal should be pre-processed.

After pre-processing, the clean speech signal is passed through the feature extraction module [11]. The feature extraction module is responsible for determining the valuable components of an audio signal that contribute to identifying linguistic content, while eliminating irrelevant information. This process involves transforming speech signals into a stream of feature vectors containing coefficients that carry the necessary information for identifying a given utterance [14, 15, 11]. Utterances are the speaking of a word, and can be a single word, a few words, a sentence, or even multiple sentences [14].

Furthermore, as third component the classification model is an important component of an ASR system, since it utilizes the extracted features from the preceding module to predict the corresponding text. Its primary task is to classify and determine the phoneme or word spoken in the input signal. The classifier learns the relationship between the given input audio features and their associated text or phonemes [11].

The language model is the last module of the ASR [11]. A language model helps to organize words to form meaningful sentences. The model indicates how likely a sequence of words can appear together [4]. Language models use structural constraints of a language to predict the probabilities of the occurrence of a word for a specific word sequence. Although a classifier and a language model are similar, they serve distinct purposes. A classifier is responsible for mapping speech signals to its closes possible word sequence, whereas a language model focuses on the occurrence probability of word sequences. It checks how likely a particular sequence of words is to occur naturally in a given language [11].

2.2 VOICE ACTIVITY DETECTION

Since the quality of speech recognition algorithms highly depends on the efficiency of the speech signal pre-processing [2, 12], it is important to discuss pre-processing techniques in the light of enhancing the performance of the Whisper algorithm. According to the work of [12], voice activity detection (VAD) is the initial pre-processing technique among several techniques as discussed in Section 2.1. The decision is made to only focus on the voice activity detection technique as a pre-processing step. According to the literature, voice activity detection is a key pre-processing technique used to detect speech segments within an utterance [16]. It involves segmenting an audio signal into parts that contain speech and unvoiced parts [17, 12].

As illustrated in Figure 2.2.1, the voice activity detection model generates outputs for each segment in an audio file, representing the model's estimation of speech presence probabilities ("Voice Activity Estimate" in image).[1] Adjusting the threshold value within the voice activity detection model, if incorporated into an ASR model, influences the transcription process of the ASR. Higher threshold values result in fewer segments being considered as speech, thereby reducing the number of segments that the ASR model transcribes.

Voice activity detection techniques are used to reduce the computational costs and response time of speech recognition models by only passing detected speech frames into the recognition algorithm [18], so that the non-speech parts are not transcribed. It is expected that by

1 https://wiki.aalto.fi/pages/viewpage.action?pageId=151500905

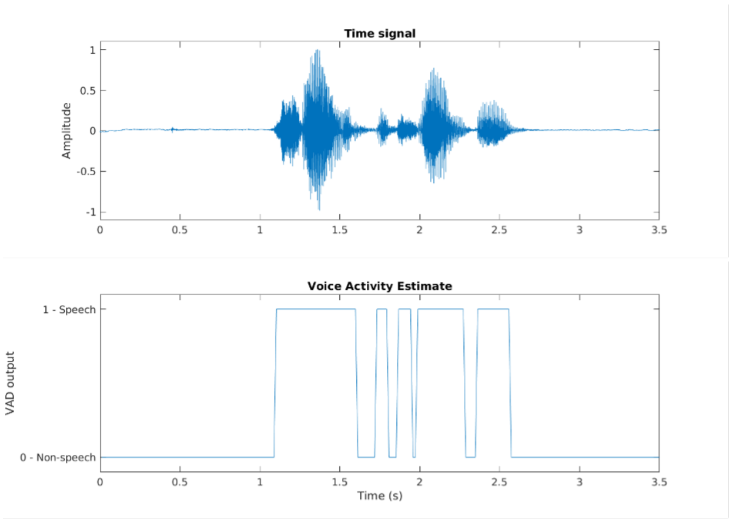


Figure 2.2.1: Voice activity detection model

incorporating a voice activity detection model into an ASR model, the transcription speed will increase. Prioritizing only this technique allows a deeper exploration of the performance of ASR models in combination with this pre-processing technique.

2.3 EVALUATION METRICS

After offering a thorough understanding of the structure of ASR systems, the following section delves into the evaluation of these systems. Evaluating the accuracy of an ASR model can be challenging, since the output of an ASR may not have the same length as the manually created transcription used as the ground truth [11]. Various metrics are employed to assess the performance of an ASR model, including the Word Error Rate, Precision, Recall, F1-score, and Real-Time Factor. This subsection will discuss how these evaluation metrics are built up.

Word Error Rate

The Word Error Rate (WER) is a widely used metric for estimating the performance of an ASR system [19], as it calculates errors on the word level rather than on the sentence level [11]. The WER is the ratio of incorrect words in the ASR output to the total number of words processed [20]. The lower the WER the better the model. The WER can be calculated as follows [21]:

$$WER = \frac{I + D + S}{N} \tag{2.1}$$

Incorrect words may result from substitutions (S), insertions (I), and deletions (D). N is the total number of words in the ground truth [11]. To illustrate the calculation of the WER, consider the following two sentences in Table 2.3.1:

Table 2.3.1: Word Error Rate example

Transcript	*Text*
Reference	*"The conference is scheduled for next Monday"*
Prediction	*"The conference is next Sunday"*

In this example, the output of the ASR model has two deletions (*"scheduled"* and *"for"*), and one substitution (*"Monday"* becomes *"Sunday"*). The WER can then be calculated as follows:

$$WER = \frac{0 + 2 + 1}{7} = 0.429 \tag{2.2}$$

Precision, Recall, and F1-score

The following three performance metrics are based on true positives (TPs), false positives (FPs), and false negatives (FNs). TPs represent words correctly detected by the ASR model, overlapping with the ground truth [22]. FPs are words detected by the ASR model but not present in the ground truth. This is also called hallucination.[2] FNs are words that exist in the ground truth but go undetected by the model .[3] To illustrate TP's, FP's, and FN's consider again the two sentences stated above. In this example, there are four true positives (*"The"*, *"conference"*, *"is"*, and *"next"*), two false negatives (*"for"* and *"next"*), and zero false positives (i.e., no words are added).

Precision refers to the ratio of correctly predicted words by the ASR model among all predicted words [23]. Precision is also known as the positive predictive value and can be calculated as follows [24, 23, 25]:

$$Precision = \frac{True\ Positives}{TruePositives + FalsePositives} \tag{2.3}$$

A high precision value indicates that, of all the positive predictions the ASR model made, a high proportion of them are correct, suggesting a low occurrence of false positives. Conversely, a low precision value indicates that, of all the positive predictions the model made, a significant portion of them are incorrect, indicating that the model is generating many false positives relative to true positives.

Recall, also known as sensitivity or true positive rate, refers to the ratio of correctly predicted words by the ASR model to the actual positive instances. The actual positive instances are all the words that should have been predicted by the ASR model, reflecting the words spoken in the audio [23, 24]. Recall can be calculated as follows [23, 25]:

$$Recall = \frac{True\ Positives}{TruePositives + FalseNegatives} \tag{2.4}$$

A high recall value indicates that the ASR model successfully recognizes all the actual words of the audio file. In contrast, a low recall value indicates that the model is omitting a considerable number of words from the input audio.

The F1-score is a combination of the two metrics as described above: precision and recall [23]. The F1-score can be calculated as follows [23, 25]:

$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.5}$$

The F1-measure is a weighted combination of precision and recall, and shows the overall accuracy of the speech segment detection [23, 25]. The higher the F1-score, the higher the overall accuracy of the output of the ASR model.

2 https://github.com/openai/whisper/blob/main/model-card.md
3 https://www.italk2learn.com/evaluating-the-performance-of-automatic-speech-recognition-systems/

Real-Time Factor

The last performance metric that this research will look into is the Real-Time Factor (RTF). RTF is the most commonly used metric for calculating the speed of a proposed model and can be computed as follows:

$$RTF = \frac{P}{I} \tag{2.6}$$

where P is the time taken by the model to process the input and I is the duration of the input audio. If RTF equals 1, then the input audio was processed in "real time" [11].

2.4 LITERATURE REVIEW

The purpose of this section is to provide a review of the literature available on the performance of Whisper and the re-implementations Faster-Whisper and WhisperX that both incorporate the pre-processing technique of voice activity detection.

2.4.1 Whisper

The Whisper algorithm, as shortly discussed in Chapter 1, has a transformer architecture which involves a two-part process: the encoder, which transforms speech inputs into context-rich representations, and the decoder, which interprets these representations into corresponding language tokens [8].

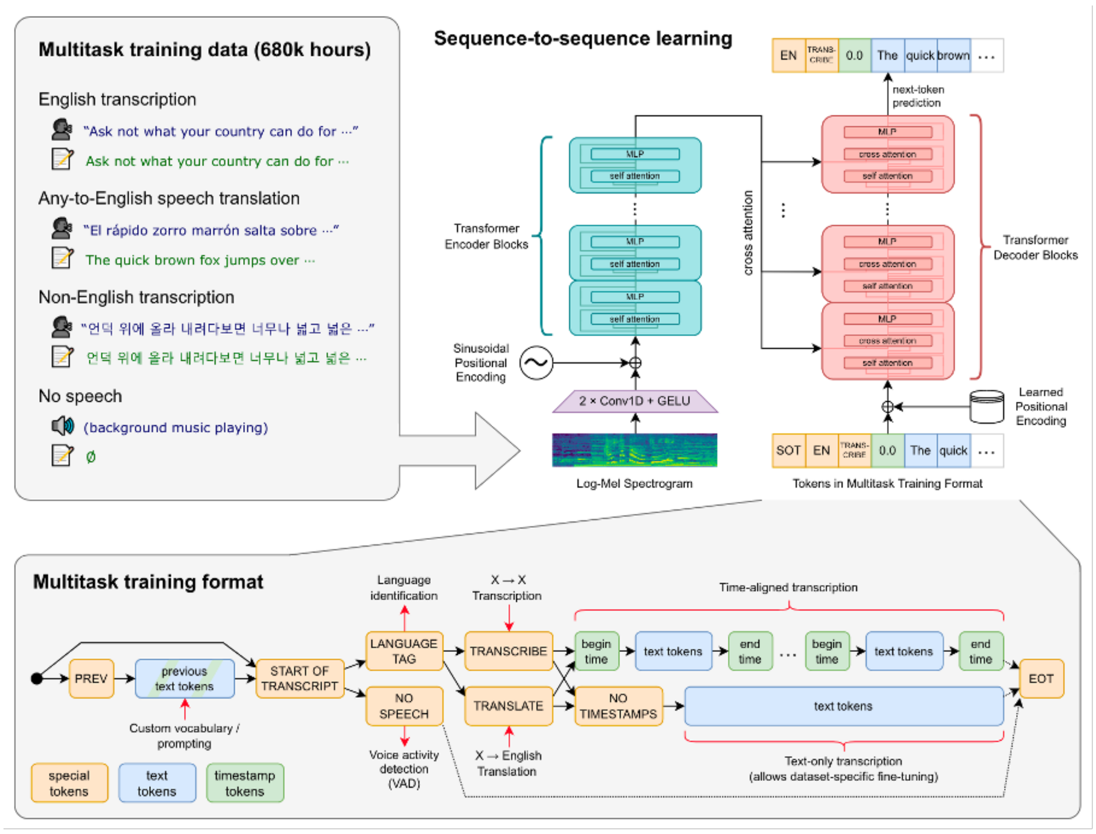

Multitask training data (680k hours)
English transcription
"Ask not what your country can do for ···"
Ask not what your country can do for ···
Any-to-English speech translation
"El rápido zorro marrón salta sobre ···"
The quick brown fox jumps over ···
Non-English transcription
"언덕 위에 올라 내려다보면 너무나 넓고 넓은 ···"
언덕 위에 올라 내려다보면 너무나 넓고 넓은 ···
No speech
(background music playing)
∅
Sequence-to-sequence learning
EN TRANS-CRIBE 0.0 The quick brown ···
next-token prediction
MLP
cross attention
self attention
Transformer Encoder Blocks
Transformer Decoder Blocks
cross attention
Sinusoidal Positional Encoding
2 × Conv1D + GELU
Log-Mel Spectrogram
Learned Positional Encoding
SOT EN TRANS-CRIBE 0.0 The quick ···
Tokens in Multitask Training Format
Multitask training format
PREV
previous text tokens
Custom vocabulary / prompting
START OF TRANSCRIPT
LANGUAGE TAG
Language identification
NO SPEECH
Voice activity detection (VAD)
TRANSCRIBE
X → X Transcription
TRANSLATE
X → English Translation
begin time
text tokens
end time
Time-aligned transcription
NO TIMESTAMPS
text tokens
Text-only transcription (allows dataset-specific fine-tuning)
EOT
special tokens
text tokens
timestamp tokens

Figure 2.4.1: Whisper pipeline
[5]

Figure 2.4.1 visualizes this pipeline. Whisper is designed for multitasking, starting with the dual functions of discerning the language spoken and detecting the presence of speech. Subsequent stages define the task to be performed - whether transcription or translation - and incorporate timestamps [5]. In the Whisper pipeline in Figure 2.4.1, the top part represents the encoder-decoder architecture, whereas the bottom represents the multitask training format. This systematic approach allows Whisper to effectively fulfill a variety of tasks, including transcription, translation, language identification, and voice activity detection [9, 5].

Different model sizes are used to train in order to study the scaling properties of Whisper (see Table 2.4.1) [5]. Training on a diverse dataset covering a broad distribution of audio from different environments, recording setups, speakers, and languages significantly improves the robustness of a speech recognition model. This makes it possible to achieve high-quality results on unseen datasets, without dataset-specific finetuning. Table 2.4.1 illustrates the aforementioned model sizes [5].

Table 2.4.1: Architecture details of the Whisper model family [5]

	Whisper			
Model	*Layers*	*Width*	*Heads*	*Parameters*
Tiny	4	384	6	39M
Base	6	512	8	74M
Small	12	768	12	244M
Medium	24	1024	16	769M
Large	32	1280	20	1550M

Whisper exhibits strong performance across various tasks and languages; however, it has a few shortcomings that are addressed within this research. Firstly, although Whisper demonstrates quite good performance in terms of Word Error Rate for Dutch (i.e., 9.3, 5.8, 12.9, 6.7, 22.4, 41.2) when compared to other relatively low-resources languages, this validation is not representative of the underlying research because of diverse characteristics of the five tested datasets by Whisper. The first dataset, Multilingual LibriSpeech (MLS), comprises read aloud audiobooks [26]. Common Voice 5.1, the second dataset, involves Dutch audio data of read aloud sentences that are displayed on a screen [27]. VoxPopuli, the third dataset, contains recordings of speeches from European Parliament events [28]. The fourth dataset, FLEURS, consists of parallel speech recordings of sentences from Wikipedia, read by three native speakers [29]. Lastly, the CoVoST2 dataset, a multilingual speech translation corpus, offers translated speech recordings, including Dutch, with multiple speakers reading donated sentences [30]. It is important to note that these datasets differ significantly from data that the Dutch National Police encounters. Therefore, different datasets will be used, as discussed in Chapter 3.

Moreover, an additional limitation identified in Chapter 1 is the considerable execution time associated with Whisper. Another problem relates to the occurrence of hallucinations in Whisper's output, with the ASR model generating texts that are not actually present in the corresponding audio input.[2] In an attempt to address these problems, there are re-implementations of Whisper to improve execution time and reduce hallucinations. Two re-implementations are Faster-Whisper and WhisperX, both including a voice activity detection model as part of their framework, are discussed in the upcoming subsections.

2.4.2 Faster-Whisper and Silero VAD

Faster-whisper[4] is a re-implementation of Whisper which uses CTranslate2[5], a C++ and Python library that enables an efficient inference with Transformer models [6]. At present, little to no official paper has been published regarding the usage and performance of Faster-Whisper. Consequently, the available information about this re-implementation is derived solely from their GitHub page.[4]

Faster-Whisper, as opposed to the original Whisper implementation, incorporates a pre-processing technique, namely a voice activity detection model called Silero VAD [31]. Silero VAD employs a neural network and is pre-trained on large corpora that include over 100 languages. Silero VAD is a multi-head attention based neural network[6] and outputs a probability of a segment being speech, which is considered speech above a pre-defined threshold. The threshold is selected by the user of the model and determines if there is speech in the given audio. If the speech probability of an audio chunk is higher than the set threshold, it is assumed to be speech. Depending on the desired result, thresholds should be fine-tuned for a specific data set or domain. Silero VAD is compared to other voice activity detection models such as SpeechBrain, WebRTC, and a commercial model, and it shows improvements on both precision and recall.[7] Additionally, the model performs well on audio files from different domains with various background noise and quality levels.[8]

Faster-Whisper is compared to the original Whisper implementation on CPU and GPU using different evaluation metrics, including time and memory usage. On both the large-v2 model on GPU and the small model on CPU, Faster-Whisper is up to 4 times faster than the original Whisper model for the same accuracy while using less memory.[9]

2.4.3 WhisperX and PyAnnote VAD

Another re-implementation of Whisper is WhisperX [22], proposed by the research group Visual Geometry Group of the Oxford University. The developers created this re-implementation to overcome several challenges of the original Whisper, such as drifting, hallucination, and repetition [22]. WhisperX has, among other things, incorporated the PyAnnote VAD as a pre-processing technique [32] consisting of a recurrent neural network [33]. PyAnnote VAD outputs $yt = 0$ if there is no speech at time t and $yt = 1$ if there is speech at time t. Timestamps with prediction scores greater than a tunable threshold Θ_{VAD} are marked as speech [32]. PyAnnote 2.0 for voice activity detection is compared to models that are specifically trained for voice activity detection and it shows - in general - better results on false alarm rate, missed detection rate, precision, recall, and F1-score [34]. After the segmentation of the voice activity detection, the input audio is then cut and merged into approximately 30-second input chunks with boundaries that lie on minimally active speech regions. The resulting speech segments are transcribed with Whisper [22]. Figure 2.4.2 illustrates the pipeline of WhisperX.

The WhisperX large-v2 model is compared to previous state-of-the-art work in speech transcription, namely the original Whisper large-v2 model and Wav2Vec2.0 base model. The developers of WhisperX used several evaluation metrics, including WER, speed, precision and re-

4 https://github.com/guillaumekln/faster-whisper
5 https://github.com/OpenNMT/CTranslate2/
6 https://thegradient.pub/one-voice-detector-to-rule-them-all/
7 https://github.com/snakers4/silero-vad/wiki/Quality-Metrics#info
8 https://github.com/snakers4/silero-vad
9 https://github.com/guillaumekln/faster-whisper

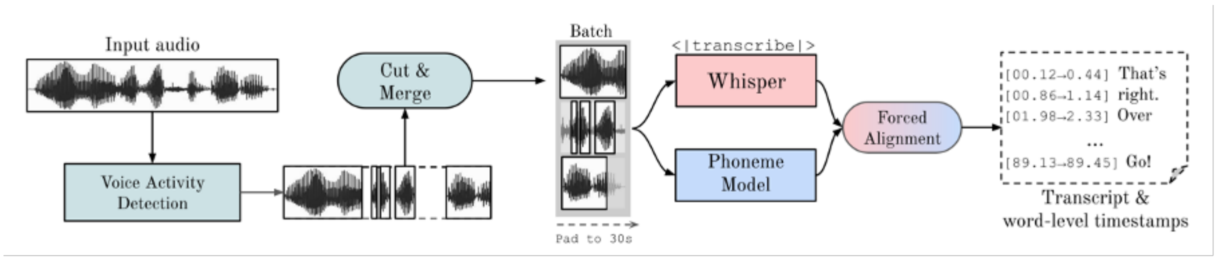


Figure 2.4.2: Pipeline of WhisperX
[22]

call, and they compared the three models on three different datasets. The first dataset is called AMI Meeting Corpus, which consist of recordings of meetings [35]. Switchboard-1 Telephone Speech Corpus (SWB), the second dataset, consists of two-sided telephone conversations [36]. Finally, TED-LIUM 3 consists of 11 TED talks [37]. The overall results of WhisperX show significant improvements on the WER compared to both Wav2Vec2.0 and the original Whisper. Besides, WhisperX outperforms both Wav2Vec2.0 and Whisper in transcription speed. The developers of WhisperX also looked into the specific effects of the implementation of a voice activity detection model. These results show that a voice activity detection model as a pre-processing technique is beneficial for the general transcription quality, and this benefit is more pronounced on word segmentation precision and recall. Finally, batched inference with voice activity detection as in WhisperX provides an almost twelve-fold speed increase with no performance loss, since each segment in the batch can be independently transcribed, which overcomes the limitations of buffered transcription as in the original Whisper [22].

According to the literature reviewed, Whisper is showing promising results on the Word Error Rate for Dutch audio files. However, the datasets on which it was tested have different characteristics from the data used in this research. In addition, the execution time of Whisper is quite high, and the output suffers from hallucination. Two re-implementations have been proposed, called Faster-Whisper and WhisperX, both of which include a voice activity detection model, to overcome these shortcomings. This research aims to conduct a comprehensive comparison between Whisper, Faster-Whisper, and WhisperX on Dutch long-form audio data, which has not been previously explored in the existing literature.

3 DATA

This chapter describes the data used in this research to compare the performance of different Whisper implementations. The research focuses on audio data including speech, and its corresponding transcriptions derived from two different datasets: CGN [38] and NFI-FRITS [39]. These datasets are similar to the data that the Dutch National Police encounters. Both datasets are explained within this chapter, however, it is important to note that direct access to the NFI-FRITS dataset is restricted, and there is limited publicly accessible information due to the sensitive nature of this data. As a result, the focus of this section will mainly be on the CGN dataset.

3.1 DATA DESCRIPTION

The NFI-FRITS dataset, provided by the Dutch National Police, consists of recordings of telephone speech intercepted by Dutch law officials during police investigations. It contains forensically realistic material, i.e., audio data from intercepted telephone speech originating from real police investigations [39]. The audio files are accompanied by their transcriptions. The dataset includes approximately 165 hours of annotated speech in 4188 different audio files. It consists of diverse speech material, including Dutch, Moroccan, Arabic, Berber, and Turkish. The recordings involve a total of 604 speakers, with 117 female and 427 male [39].

The Spoken Dutch Corpus (CGN) is an open-source dataset that includes recordings of everyday conversations on various subjects and in a variety of situations, along with their annotations and transcriptions [40]. The dataset aims to meet the needs of diverse research disciplines and applications [41]. Accordingly, the dataset includes (fragments of) radio and television programs, interviews, lectures, and telephone conversations. Table 3.1.1 provides an overview of all the components within this open-source dataset [39].

Table 3.1.1: Components CGN dataset

	CGN
Component	*Description*
Comp A	Spontaneous conversations (face-to-face)
Comp B	Interviews with teachers of Dutch
Comp C	Spontaneous telephone conversations (recorded via a switchboard)
Comp D	Spontaneous telephone conversations (recorded on minidisk)
Comp E	Simulated business negotiations
Comp F	Interviews, discussions, debates (broadcast)
Comp G	(Political) Discussions, debates, meetings (non-broadcast)
Comp H	Lessons recorded in a classroom
Comp I	Live (e.g. sport) commentaries (broadcast)
Comp J	News reports / reportages (broadcast)
Comp K	News (broadcast)
Comp L	Commentaries / columns / reviews (broadcast)
Comp M	Ceremonious speeches / sermons
Comp N	Lectures / seminars
Comp O	Read speech

The CGN dataset consists of approximately 900 hours of recorded audio along with corresponding transcriptions. It includes various types of speech components, including monologues and dialogues, with the number of speakers per component varying per audio file.[1] The CGN dataset comprises both Dutch and Flemish audio files, with approximately 76.23% of the files in Dutch and 23.77% in Flemish, on average [41].

To accommodate computational and time constraints of this research, the NFI-FRITS and CGN datasets are segmented. Regarding the CGN dataset, Component C and G are used. Component C was expected to be most similar to the data in the NFI-FRITS dataset, and Component G was included for the purpose of generalizability of the results. From these components, only the Dutch audio files are used. Regarding the NFI-FRITS dataset, the subset that is used in this research consists only of Dutch conversations between 2 speakers. In this research, NFI-FRITS and CGN are used to refer to these subsets. Table 3.1.2 provides basic statistics of the subsets employed in this research.

Table 3.1.2: Datasets used in research

Data	*Min*	*Mean*	*Max*	*Total*
NFI-FRITS				
Seconds	57.41	172.83	624.61	39404.34
Minutes	0.96	2.88	10.41	656.74
Num Files				228
CGN Comp C				
Seconds	209.11	556.24	599.81	199135.19
Minutes	3.49	9.27	10.00	3318.92
Num Files				358
CGN Comp G				
Seconds	160.06	831.70	1788.98	74853.41
Minutes	2.67	13.86	29.82	1247.56
Num Files				90

3.2 DATA COLLECTION METHODS

The data collection methods are different for the NFI-FRITS and CGN datasets. For the NFI-FRTIS dataset, audio files are obtained through the interception of telephone speech by Dutch law officials during police investigations. The individuals speaking in these recordings were likely unaware of being recorded. Background sounds such as periodic background sounds, background speakers, and cross-talks are excluded from the audio files [39]. Unfortunately, further details regarding the data collection method for these audio files are not publicly disclosed. In addition to the audio files, annotations, and transcriptions are collected. Native annotators in the relevant languages listened to the audio files, identified speaker names and genders (male/female), and processed sensitive information, including telephone numbers and case names. Subsequently, the audio was processed again, and speech containing potentially identifying information about an individual's identity is removed by setting the digital

1 https://lands.let.ru.nl/cgn/doc_Dutch/topics/design/design.htm#intro

sample values to zero. Additional metadata, such as the level of background noise, age group, language proficiency, and conversation type, are also added [39]. Alongside these annotations, separate transcriptions are created for each speaker in the telephone conversations, although specifics about this process remain undisclosed [39].

The CGN dataset comprises recordings obtained through diverse methods. Certain components, such as interviews and discussions (f), are obtained from external parties, while face-to-face conversations (a) are recorded between recruited volunteers in their home environments. Another group of individuals was instructed to record data in various settings, such as shops, workplaces, or restaurants. Additionally, components like classes and lectures are directly recorded at educational institutions with permission given by the schools [42]. The CGN dataset includes orthographic transcriptions for all recordings. These transcriptions provide a verbatim record of the spoken content without correcting grammatical errors or filling in incomplete words.[2] Moreover, metadata such as gender, age, and geographical region are noted [42]. In the process of creating the orthographic transcriptions, repetitions, slips, and hesitations are explicitly represented; background noises, on the other hand, are only under certain conditions reflected in the transcript. In order to align the orthographic transcription with the corresponding speech signal, time markers are added at regular intervals of approximately two to three seconds[3]. Moreover, additional notes are incorporated into the transcriptions, as outlined in Table 3.3.1.

3.3 DATA PRE-PROCESSING AND HANDLING

This section presents the procedures for the pre-processing and handling of the data in this research. First, Subsection 3.3.1 discusses the conversion of audio files and the extraction of features. Additionally, Subsection 3.3.2 elaborates on the data cleaning process followed by Subsection 3.3.3 which discusses the transformation of the audio files and corresponding transcriptions. Finally, Subsection 3.3.4 addresses the ethical and privacy considerations related to the datasets used in this research.

3.3.1 Audio File Conversion and Feature Extraction

No prior conversion of audio files or feature extraction was necessary for either dataset. The Whisper framework incorporates a built-in function that accepts various audio files formats, including m4a, mp3, webm, mp4, mpga, wav, and mpeg.[4] In the case of the NFI-FRITS and CGN datasets, which are both in .wav format, no additional processing was required. Furthermore, Whisper includes a feature execution function that directly operates on the raw audio files. This function converts the audio files into Log-Mel Spectrograms; representations that contain information about time, frequency, and amplitude. Thus, no additional feature extraction method was employed.

2 https://taalmaterialen.ivdnt.org/wp-content/uploads/documentatie/cgn_website/doc_Dutch/topics/project/orthography/index.htm

3 https://ivdnt.org/images/stories/producten/documentatie/cgn_website/doc_Dutch/topics/project/orthography/index.htm

4 https://help.openai.com/en/articles/7031512-whisper-api-faq

3.3.2 Data Cleaning

No specific cleaning steps are performed on the audio files of either dataset. However, the transcriptions are cleaned. For the NFI-FRITS dataset, some pre-processing of the transcriptions has been carried out by the Dutch National Police. This involved the removal of words that are in an exclusion list, such as "UNK", "PERSON", "xxx", "ggg", "vvv", and "uh". Additionally, certain parts of words, such as "-uh", "*s", "*x", "*u", "*a" are eliminated. Punctuation marks are also removed and the text is transformed to lowercase.

The preparation of the original transcriptions involved several steps. The manual-generated transcriptions of the CGN dataset differed in format from the output of Whisper and its re-implementations. While Whisper's output was sorted chronologically, the original transcripts were sorted by speaker. This mismatch in formats posed challenges in accurately comparing the outputs, as it could lead to incorrect alignments and with that incorrect calculations of errors. To address this, the original transcriptions are sorted based on the provided time stamps. Furthermore, in the subsequent step, the sentences are further cleaned using text processing methods. The orthographic transcriptions of the CGN dataset contained additional words or symbols which are represented in Table 3.3.1). These additional words and symbols are removed.

Table 3.3.1: Words/symbols in orthographic transcriptions CGN dataset

Word/Symbol	*Meaning*
$*d$	Dialect word
$*u$	Mispronounced word
$*a$	Incomplete word
$*v$	Foreign word
$*x$	Unintelligible word
$*z$	Word pronounced with a regional accent
ggg	A sound produced by the speaker (e.g., laughing)
xxx	One or more unintelligible word
Xxx	An unintelligible proper noun

Besides, the original transcriptions included punctuation marks, which are also removed, and the transcriptions are converted to lowercase. Table 3.3.2 illustrates an example of the outcome of the cleaning process of a transcription of the CGN dataset. The transcriptions produced by Whisper or a re-implementation undergo the same processing, which will be elaborated upon in Chapter 4 in further detail. This ensures greater comparability between the original transcriptions and the predictions generated by whisper or a re-implementation.

3.3.3 Data Transformation

Aforementioned, the audio files in the CGN dataset are in a format that is supported by Whisper (.wav). Hence, it was unnecessary to transform the audio files. The transcriptions in the CGN dataset are in the .ort format.[5] To enable a comparison between the original transcriptions and the transcriptions provided by Whisper or a re-implementation, it was necessary to parse the .ort files and extract necessary information, i.e., the speaker, start time, end time, and transcription. This information was needed for further steps in this research, as will be

5 https://taalmaterialen.ivdnt.org/wp-content/uploads/documentatie/cgn_website/doc_Dutch/topics/index.htm

Table 3.3.2: Original vs cleaned transcription

Transcription	*Meaning*
Original	[ja. ze zegt*x uh*x je raakt helemaal onder de indruk zei ze. mmm. dus. mijn stem klinkt denk ik heel raar want ik ben een beetje verkouden. mmm. ja. hoor je dat erg? beetje wel. ja wel een beetje maar niet heel erg ja nou ja. ja. maar dat weten ze dan toch wel. nou ja maar dat maakt toch niet uit. ggg. ggg. nou. mensen ik ben verkouden. ggg. ggg. ggg. ja. hoor je mij? ggg. ggg. oh. ja. ja]
Cleaned	[ja ze zegt uh je raakt helemaal onder de indruk zei ze mmm dus mijn stem klinkt denk ik heel raar want ik ben een beetje verkouden mmm ja hoor je dat erg beetje wel ja wel een beetje maar niet heel erg ja nou ja ja maar dat weten ze dan toch wel nou ja maar dat maakt toch niet uit nou mensen ik ben verkouden ja hoor je mij oh ja ja]

explained in Chapter 4. The specific format of the transcriptions in the NFI-FRITS dataset has not been disclosed. However, the Dutch National Police provided the transcriptions in a format suitable for subsequent analyses.

3.3.4 Ethics and Privacy

The speech content of the NFI-FRITS dataset has been anonymised by zeroing out fragments that might disclose the real identity of speakers. On average, 3.85s of speech (2.7%) was nulled as a result of the anonymisation procedure [39]. Besides, metadata that contain information that can disclose the real identify of a particular speaker was deleted or replaced with salted hashes or ID-numbers [39]. In the subset of the NFI-FRITS dataset used for this research, all meta-data such as gender and accent, is removed.

In the CGN dataset, each speaker in the corpus are assigned a unique identification code. Information about the speakers is included in the meta-data. The developers of the corpus used descriptions that could not led to the identify of a speaker, ensuring that their anonymity remains protected. They used a classification according to age, class, socio-economic class, etc. [42]

Due to the sensitive nature of the NFI-FRITS dataset, specific findings cannot be publicly disclosed in this research; however, a description of these results will be provided. This research uses the CGN dataset to illustrate specific findings and provide examples.

4 METHODOLOGY

The aim of this research is to evaluate the performance of Whisper and re-implementations that incorporate a voice activity detection model into Whisper on Dutch long-form audio data. This chapter describes the methodology used to evaluate the performance of Whisper, Faster-Whisper, and WhisperX. . First, Section 4.1 explains the general research setup, after which Section 4.2 explains the conducted experiments in this research. Finally, Section 4.3 discusses the implementation of the different evaluation metrics used in this research.

4.1 GENERAL RESEARCH SETUP

In order to conduct all experiments on the server of the Dutch National Police, specific requirements must be satisfied for the chosen implementations. These requirements include: being an open-source implementation, written in Python, regularly updated, accompanied by a pre-trained model and properly licensed. Whisper, Faster-Whisper and WhisperX, including Silero VAD and PyAnnote VAD, meet all of these requirements, making them suitable for comparison in this research. Whisper, Faster-Whisper, and WhisperX are first compared with their default settings. Additionally, the comparison explores the effects of adjusting hyperparameters in the models i.e., *condition_on_previous_text*, *VAD_filter*, and VAD_threshold (threshold_onset and threshold_offset). All these experiments are conducted on the NFI-FRITS and CGN dataset. Figure 4.1.1 illustrates the general research setup of the underlying research. Considering the slow inference time on central processing units (CPUs) when using larger models, and the availability of computational resources, the models are executed on a Nvidia Titan RTX graphics processing unit (GPU).

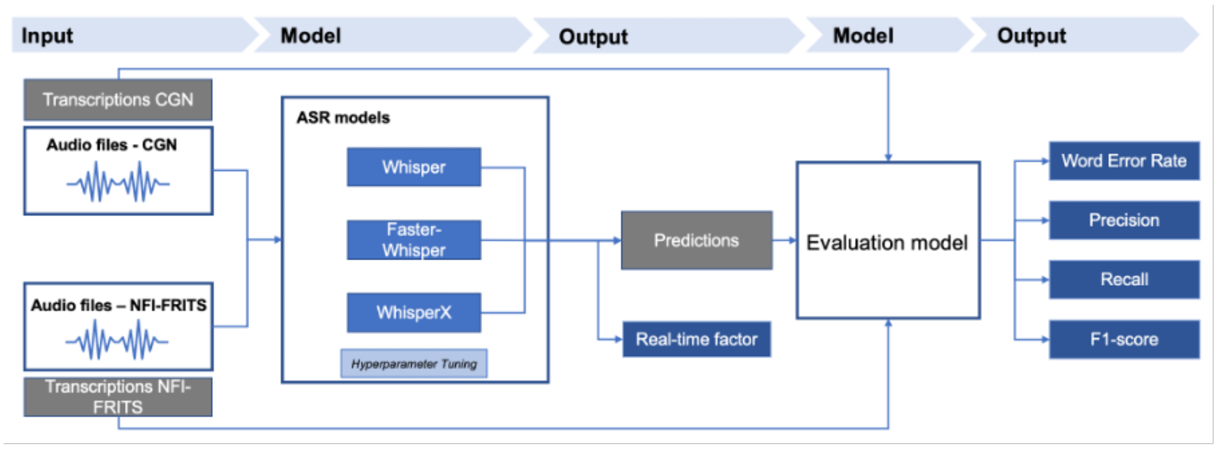


Figure 4.1.1: Research Pipeline

As can be seen in Figure 4.1.1, the input for the models consists of the audio files from two datasets along with their corresponding transcriptions. The audio files are transcribed by Whisper, Faster-Whisper and WhisperX, while simultaneously monitoring the Real-Time Factor. Subsequently, the predictions and original transcriptions are compared in an evaluation model to assess the WER, precision, recall, and F1-score.

4.2 EXPERIMENTS

This chapter discusses the conducted experiments within this study. First, Subsection 4.2.1 explains the experiments that are performed with Whisper, Faster-Whisper, and WhisperX, as well as the hyperparameter tuning within these models. Subsection 4.2.2 outlines the conducted experiments to assess the variation in performance between the two voice activity detection models. These experiments aim to evaluate the impact of incorporating different voice activity detection models within an ASR model. The focus lies on comparing the results solely based on the output of these models, disregarding any analysis of the generated texts.

4.2.1 Whisper, Faster-Whisper, and WhisperX

This subsection discusses the details of the experiments that are conducted in this research. Figure 4.2.1 illustrates these different experiments. Within Whisper and its re-implementations, there are multiple hyperparameters that can be tuned. This research focuses on the following hyperparameters *condition_on_previous_text*, *VAD_filter*, and threshold, as requested by the Dutch National Police. These hyperparameters will be explained in this section. Other hyperparameters that can be tuned include model_size, compute_type, beam_size, and batch_size. These are not tuned in this research, to limit the scope of work.

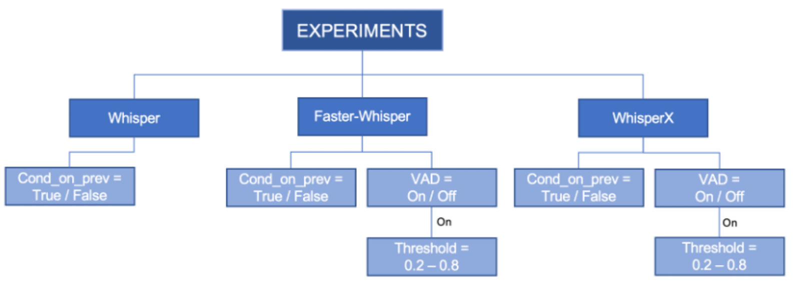


Figure 4.2.1: Different experiments of this study

Whisper, Faster-Whisper and WhisperX all offer different model sizes. To be able to compare the performance of the three different implementations with each other, the decision was made to use models of the same size. Larger models have been shown to achieve a better accuracy [5], therefore this research uses the large-v2 model, which is the largest model at time of this research.

First, the *condition_on_previous_text* hyperparameter can be adjusted within the three different models to True and False. If True, the previously generated output of the model is provided as a prompt for the next predictions. Although this leads to more consistent texts, the model becomes more prone to getting stuck in a failure loop such as repetition.[1] Another risk of setting this hyperparameter to True is hallucination, i.e., generating text while this text is not being said in the original audio. Therefore, according to the work of [22], setting this hyperparameter to False is beneficial for robust transcriptions. The default setting of this hyperparameter for Whisper and Faster-Whisper is True, and for WhisperX it is False. In this research, experiments are conducted with *condition_on_previous_text* being set to True and False for all three models to guarantee a fair comparison between the models.

Another hyperparameter that is tuned in this research is related to the voice activity detection

1 https://github.com/openai/whisper/blob/main/whisper/transcribe.py

models used in Faster-Whisper and WhisperX. As explained in Subsection 2.4, both Faster-Whisper and WhisperX incorporate an optional voice activity detection model (*VAD_filter*). By enabling the *VAD_filter* in the models, parts of the audio without speech are filtered out. By default, Faster-Whisper sets the *VAD_filter* to False, while WhisperX sets the *VAD_filter* to True. In order to examine the effects of the voice activity detection model, experiments are conducted with the model being enabled and disabled. In Faster-Whisper, the hyperparameter *VAD_filter* can easily be disabled, however, in WhisperX this hyperparameter is not explicitly included. Therefore, the same effect is achieved by setting the speech probability threshold in the voice activity detection model of WhisperX to 0. In this way, the voice activity detection model is disabled, as it considers all segments to be speech.

Finally, experiments are performed to explore the effect of changing the threshold in the voice activity detection model of Faster-Whisper and WhisperX. As described before, Faster-Whisper incorporates the Silero VAD model to filter out parts of the audio without speech.[2] Silero VAD outputs speech probabilities for each audio chunk, and probabilities above the threshold value are considered as speech.[3] The right column in Figure 4.2.2 illustrates this threshold. The threshold parameter, a number between 0 and 1, in the transcribe function of Faster-Whisper is adjusted. By setting the threshold to a high value, such as 0.9, it is expected Whisper to be faster, as it will only process a limited number of audio chunk that are classified as speech by the voice activity detection model. Consequently, Whisper will have fewer chunks to transcribe. This adjustment in threshold value is expected to enhance the precision of transcriptions, as the audio inputs are more likely to be speech instead of other noises, which is expected to reduce hallucinations of Whisper. Conversely, it is expected that the recall value will decrease due to an increase in false negatives, as Whisper may omit numerous audio segments that do contain speech in case of a high threshold value in the voice activity detection model. The default value in Silero VAD in Faster-Whisper is 0.5, and multiple threshold values above and below 0.5 are used to test the influence on the performance. By doing so, well-balanced research can be conducted.

WhisperX incorporates the PyAnnote VAD model.[4] The threshold value functions differently in PyAnnote VAD compared to Silero VAD in Faster-Whisper. The PyAnnote VAD model uses two values, Onset and Offset, to determine the presence of speech in an audio chunk. As illustrated in the left figure of Figure 4.2.2, the PyAnnote VAD model considers an audio chunk as speech once it exceeds the Onset threshold, and it continues to be classified as speech until it falls below the Offset threshold.[5] The default values are Onset: 0.500 and Offset: 0.363.

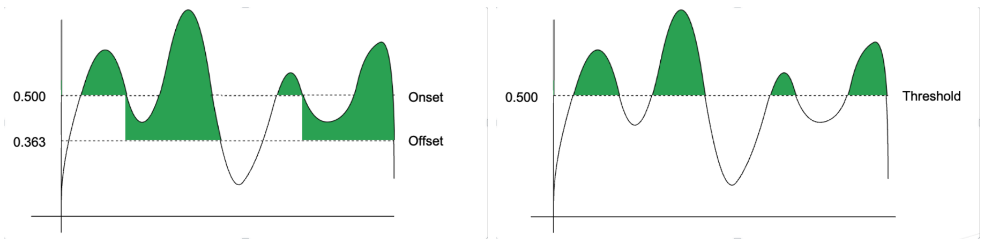


Figure 4.2.2: A visual overview of the hyperparameters in PyAnnote (left) and Silero VAD (right). Green segments are considered speech.

2 https://github.com/guillaumekln/faster-whisper/tree/ad58ba26ab8b3d871b8d4f9962cf8a669c3d41c1
3 https://github.com/guillaumekln/faster-whisper/blob/master/faster_whisper/vad.py
4 https://github.com/m-bain/whisperX/tree/befe2b242eb59dcd7a8a122d127614d5c63d36e9
5 https://github.com/PyAnnote/PyAnnote-audio/blob/develop/tutorials/voice_activity_detection.ipynb

To determine the optimal approach for tuning the Onset and Offset values, preliminary tests are performed. These tests examined the output differences when scaling the Onset and Offset values linearly, as well as when setting both values to an equal value (i.e., omitting the margin between Onset and Offset), resulting in a threshold parameter similar to the threshold parameter in the Silero voice activity detection model of Faster-Whisper. The results of these preliminary tests indicated minimal variation in the output. Consequently, the decision was made to set the Onset and Offset values to be identical. As an example, the Onset and Offset values of 0.500 and 0.363 in Figure 4.2.2 become one threshold value of 0.5. This choice ensures a fairer comparison between Faster-Whisper and WhisperX. As a result, the voice activity detection threshold values in both Faster-Whisper and WhisperX are tuned in the same way, as illustrated in Table 4.2.1.

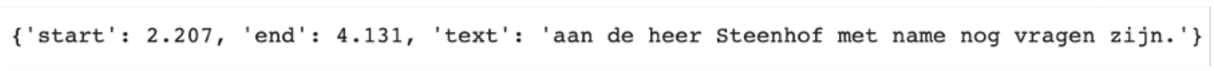
Table 4.2.1: Thresholds WhisperX (L) and Faster-Whisper (R)

Onset/Offset PyAnnote VAD	*Threshold* Silero VAD
0.20	0.20
0.25	0.25
0.30	0.30
0.35	0.35
0.40	0.40
0.45	0.45
0.50	0.50
0.55	0.55
0.60	0.60
0.65	0.65
0.70	0.70
0.75	0.75
0.80	0.80

4.2.2 Silero and PyAnnote

In addition to comparing Whisper, Faster-Whisper, and WhisperX on the different performance metrics, this research also examines the variation between the two voice activity detection models employed in Faster-Whisper and WhisperX. Specifically, Silero VAD in Faster-Whisper, and PyAnnote VAD in WhisperX. This analysis aims to determine whether any performance differences observed between Faster-Whisper and WhisperX can be attributed to variances in the voice activity detection models' performance. This subsection discusses the methods used for this comparison.

The original transcriptions of the CGN Component C and G datasets, as well as the NFI-FRITS dataset, contain timestamps indicating the occurrence of spoken words. Figure 4.2.3 visualizes these timestamps of one of the original transcriptions.



```
{'start': 2.207, 'end': 4.131, 'text': 'aan de heer Steenhof met name nog vragen zijn.'}
```

Figure 4.2.3: Timestamps in an original transcription of CGN Component G

Both Sileroand PyAnnote VAD models output the likelihood of speech occurrence within specific segments of audio in probabilities. These models provide timestamps alongside the detected speech segments. Two functions are implemented, which require an audio file and a threshold value as inputs, and yield the start and end times of the identified speech segments in seconds. Experiments are conducted in which the threshold value is modified in the same way as described in Table 4.2.1. Subsequently, the predicted speech timestamps are compared with the original speech timestamps, as specified in the original transcriptions. A more detailed description of this comparison is provided in Section 4.3.

4.3 EVALUATION METRICS IMPLEMENTATION

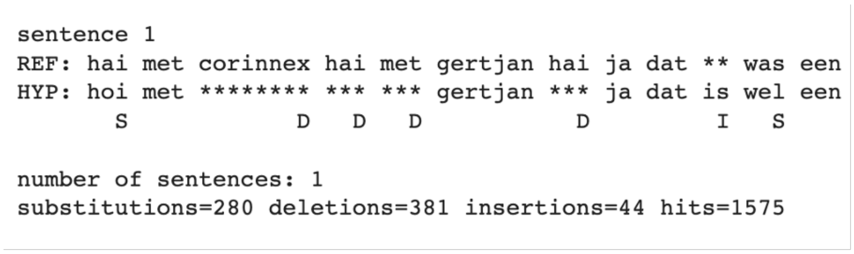
In this research, the experiments between Whisper and its re-implementations, involve providing audio files and their corresponding transcriptions as input to the developed functions, resulting in a dataframe comprising five performance metrics, namely: Word Error Rate (WER), precision, recall, F1-score, and Real-Time Factor (RTF), as described in Section 2.3. This section describes the implementation of the different performance metrics in more detail for these experiments. During the execution of an experiment, the RTF is monitored. The remaining performance metrics are computed after the execution, as they require the model's predictions for calculation. Finally, this section discusses how the performance of the two voice activity detection models is evaluated.

To determine the RTF, it is necessary to possess both the executing time of Whisper and the duration of the audio file. A function is implemented to calculate the duration of the given audio file in seconds by dividing the length of the audio file by its sample-rate. Eventually, the Real-Time Factor per audio file is determined by dividing the running time per audio file by the duration of the audio file.

After executing Whisper, the generated prediction undergoes processing. This process includes converting all the text to lowercase and removing punctuation marks. Additionally, any numerical values present in the prediction are converted into their written form in Dutch. This conversion is performed by using the telwoord package.[6] This conversion is consistent with the representation of numbers in the original transcriptions of the NFI-FRITS and the CGN datasets. For instance, the number "20" in a prediction is converted to the Dutch word "twintig" ("twenty"). However, this method is not applicable for decimal numbers.

Subsequently, the original transcriptions and the predictions are aligned with the JiWER package[7] which aligns a reference (original transcription) with a hypothesis (prediction of the ASR model) on a word-level. Within the JiWER package, the Levenshtein distance is used which is a measure of similarity between two strings. The algorithm finds the cost of the least expensive set of insertions, deletions, or substitutions that would be needed to transform one string into the other. The greater the Levenshtein distance, the more different the strings are [43, 44]. The time indications of the original transcription and a Whisper implementation often differ too much which makes it difficult to map them to each other correctly. That is why all speech in an audio file is considered as one sentence. By considering the text in both transcriptions as one sentence, JiWER itself can create a proper alignment. Figure 4.3.1 represents a part of the output of the JiWER function process_words(), where "REF" is the original transcription and "HYP" is the predicted transcription of Whisper.

6 https://pypi.org/project/telwoord/
7 https://jitsi.github.io/jiwer/ https://github.com/jitsi/jiwer



```
sentence 1
REF: hai met corinnex hai met gertjan hai ja dat ** was een
HYP: hoi met ******** *** *** gertjan *** ja dat is wel een
       S            D   D   D           D         I   S

number of sentences: 1
substitutions=280 deletions=381 insertions=44 hits=1575
```

Figure 4.3.1: Example JiWER alignment using the *process_words()* function

As can be seen in Figure 4.3.1, the JiWER alignment outputs different mutations, including insertions, deletions, substitutions and hits, which are measured on a word-level. Table 4.3.1 illustrates the meaning of these mutations. Based on these numbers, the other four performance metrics can be computed, namely the WER, precision, recall, and F1-score.

Table 4.3.1: Description of the mutations of the JiWER alignment output

Name	*Type*	*Description*
Hits (H)	int	The number of correct characters between reference and hypothesis sentences
Substitutions (S)	int	The number of substitutions required to transform hypothesis sentences to reference sentences
Insertions (I)	int	The number of insertions required to transform hypothesis sentences to reference sentences
Deletions (D)	int	The number of deletions required to transform hypothesis sentences to reference sentences

As described in Section 2.3, precision can be calculated by dividing the true positives (hits) by the sum of true positives and false positives (hits, substitutions and insertions). The recall is calculated by dividing the true positives (hits) by the sum of true positives and false negatives (hits, deletions and substitutions). The F1-score is the harmonic mean of precision and recall [25]. When the ASR model fails to generate a transcription, the precision, recall, and F1 values are assigned a value of 0, as it requires a penalty for such cases. Finally, the JiWER package has a function called wer() to calculate the WER.[8] This function calculates the WER as the sum of substitutions, insertions and deletions divided by the hits, substitutions, and deletions.[9]

To compare the two voice activity detection models, the predicted timestamps are compared to the timestamps in the original transcription using the PyAnnote.metrics library[10]. The evaluation of the voice activity detection classification task is conducted through the metrics DetectionErrorRate and DetectionAccuracy[11]. Within these metrics, the inputs consist of segments corresponding to the positive class, which represents speech segments, while gaps within the inputs are considered as the negative class, representing non-speech segments. The original timestamps (reference) are compared to the predicted timestamps (hypothesis) of the voice activity detection model. During this comparison, true negatives, true positives, false negatives,

8 https://jitsi.github.io/jiwer/reference/measures/
9 https://github.com/jitsi/jiwer/blob/master/jiwer/process.py
10 https://github.com/PyAnnote/PyAnnote-metrics/tree/develop
11 https://github.com/PyAnnote/PyAnnote-metrics/blob/develop/PyAnnote/metrics/detection.py

and false positives are calculated. In this case, true positives correspond to instances where the original timestamps align with the predicted timestamps for speech, and true negatives indicate instances where there is no speech detected by the voice activity detection model in case there is also no speech detected in the original audio and its corresponding transcription. False positives are the duration of non-speech incorrectly classified as speech, and false negatives are the duration of speech incorrectly classified as non-speech [45]. Based on these numbers, the performance metrics precision and recall are calculated in the way as described in Section 2.3.

5 PERFORMANCE RESULTS

The goal of this research is to compare the performance of Whisper with two re-implementations that incorporate the pre-processing technique of voice activity detection. The most important point of interest is therefore the extent to which the models, i.e., Whisper, Faster-Whisper, and WhisperX, perform on the different performance metrics: Word Error Rate (WER), precision, recall, F1-score, and Real-Time Factor (RTF). Therefore, the purpose of this chapter is to discuss the performance results of Whisper, Faster-Whisper, and WhisperX on CGN Component C and G, and NFI-FRITS. Additionally, the performance results of the threshold tuning within the voice activity detection models, and the performance results of the comparison between Silero and PyAnnote are presented. This chapter is organized as follows: Section 5.1 discusses the results of the experiments with Whisper, Faster-Whisper and WhisperX on CGN Component C, 5.2 discusses the results for CGN Component G, and 5.3 for the NFI-FRITS dataset. Section 5.4 presents the results of tuning the threshold values in Faster-Whisper and WhisperX. Finally, 5.5 discusses the results of the comparison between the two voice activity detection models: Silero and PyAnnote. In the tables presented in this chapter, the most optimal results are marked in bold. Model settings with "def" indicate the default settings of the model.

5.1 RESULTS ON CGN COMPONENT C

As mentioned above, this section discusses the results of the experiments with Whisper, Faster-Whisper, and WhisperX on Component C of the CGN dataset. Component C consists of telephone conversations that are recorded via a switchboard. To start with, Figure 5.1.1 illustrates the results of the three different models on the Word Error Rate (Y-axis) and the Real-Time Factor (X-axis).
The results in Figure 5.1.1 are for the models Whisper and WhisperX in their default settings, meaning that the hyperparameter *condition_on_previous_text* is set to True for Whisper and False for WhisperX, while the *VAD_filter* is set to On for WhisperX. For Faster-Whisper, the *condition_on_previous_text* is set to True, the *VAD_filter* is enabled, and the default threshold of 0.5 is used. WhisperX has an Onset value of 0.5 and an Offset value of 0.363.

In terms of the RTF, WhisperX demonstrates better performance with a mean RTF value of 0.058, indicating that WhisperX, on average, achieves faster transcriptions of the audio files in this dataset. Regarding the WER, Faster-Whisper achieves the best performance with a WER of 0.0324 (see Table 5.1.1 for details). Additionally, WhisperX exceeds Whisper in WER performance, with WhisperX achieving a WER of 0.326, while Whisper achieves a higher WER of 0.352.

As mentioned in the methodology Section (4), the hyperparameters *condition_on_previous_text* and *VAD_filter* are also altered for the three models to assess the effects on the different performance metrics. The results of this hyperparameter tuning for CGN Component C are illustrated in Table 5.1.1.
Table 5.1.1 shows the performance comparison of different models. Faster-Whisper, with *condition_on_previous_text* set to True and *VAD_filter* set to On, achieves the lowest WER of 0.324.

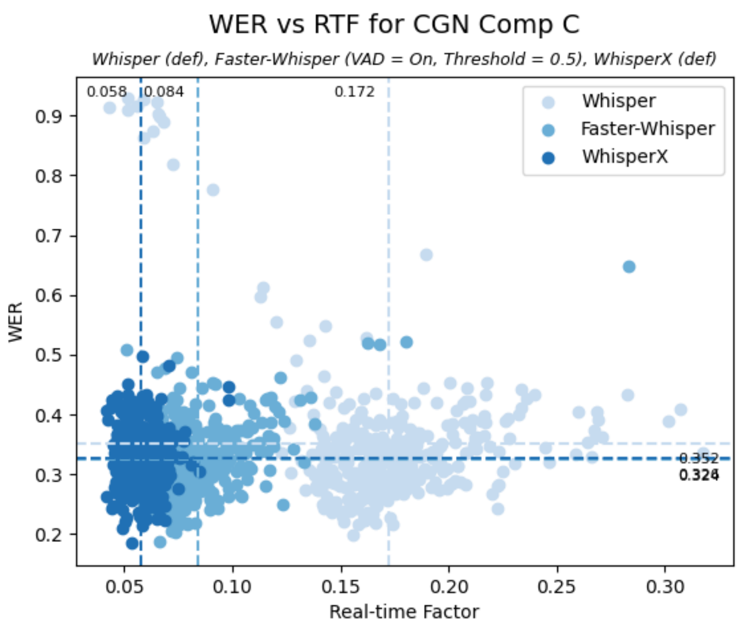


Figure 5.1.1: WER vs RTF on CGN Comp C for Whisper, Faster-Whisper, and WhisperX

Table 5.1.1: Performance metrics CGN Component C, thresholds in VAD are in default settings: for Faster-Whisper: 0.500, for WhisperX: Onset = 0.500, Offset = 0.363

			CGN Comp C				
Model	**cond_on_prev_text**	**VAD**	*WER*	*Precision*	*Recall*	*F1*	*RTF*
Whisper (=def)	True	N/A	0.352	0.813	0.667	0.729	0.172
WhisperX	True	On	0.326	**0.849**	0.685	**0.758**	0.056
WhisperX	True	Off	0.335	0.848	0.676	0.751	0.053
Faster-Whisper	True	On	**0.324**	0.838	**0.693**	**0.758**	0.084
Faster-Whisper (=def)	True	Off	0.327	0.836	0.692	0.756	0.087
Whisper	False	N/A	0.342	0.825	0.679	0.744	0.190
WhisperX (=def)	False	On	0.326	**0.849**	0.685	0.757	0.058
WhisperX	False	Off	0.335	0.848	0.676	0.751	**0.052**
Faster-Whisper	False	On	0.328	0.842	0.687	0.756	0.118
Faster-Whisper	False	Off	0.329	0.839	0.688	0.755	0.077

WhisperX, with the same settings and WhisperX in default settings, achieve the highest precision (0.849), while Faster-Whisper with *condition_on_previous_text* set to True and *VAD_filter* set to On, achieves the highest recall value of 0.693. Both WhisperX and Faster-Whisper, with these settings, attain the best F1-score of 0.758. In terms of RTF, WhisperX in all different settings outperforms Whisper and Faster-Whisper, with RTF values of 0.052, 0.053, 0.056 and 0.058. This implies that these models can transcribe the audio files of CGN Component C approximately three times faster than the baseline models Whisper, which have RTF values of 0.172 and 0.190.

5.2 RESULTS ON CGN COMPONENT G

This section presents the results of the experiments on CGN Component G, which consists of recordings of discussions, debates, and meetings. Figure 5.2.1 visualizes the results of the three different models concerning the WER on the Y-axis and the RTF on the X-axis. The aforementioned settings in Section 5.1 also apply to this figure.

As can be seen in Figure 5.1.1, WhisperX demonstrates superior performance in terms of RTF, achieving a value of 0.047, Faster-Whisper outperforms Whisper, with RTF values of 0.087 and 0.161, respectively. These findings indicate that WhisperX achieves almost three times faster transcriptions of audio files from CGN Component G compared to the baseline Whisper model. WhisperX also outperforms the other models in terms of WER, achieving a value of 0.171. Faster-Whisper achieves a slightly higher WER of 0.173, while Whisper exhibits a WER of 0.179. Detailed results can be found in Table 5.2.1.

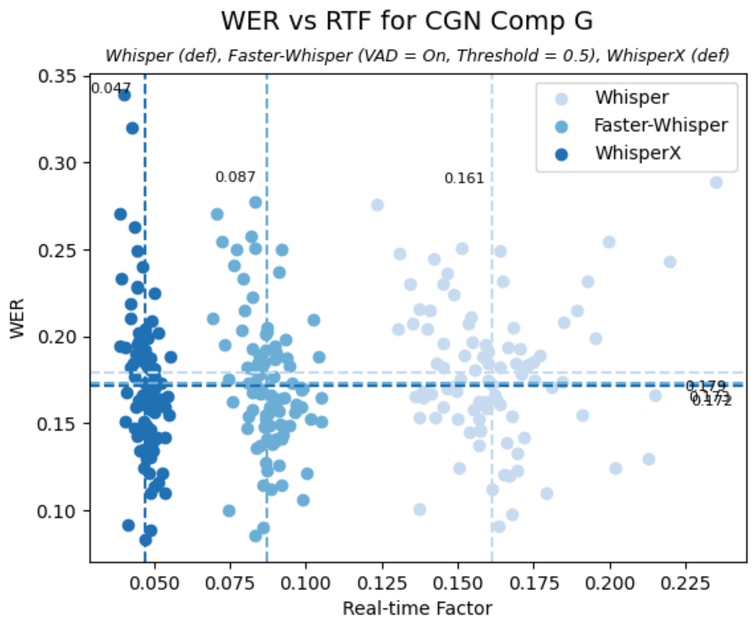


Figure 5.2.1: WER vs RTF on CGN Comp G for Whisper, Faster-Whisper, and WhisperX

Similar to the analysis of CGN Component C, the hyperparameters *condition_on_previous_text* and *VAD_filter* are altered for the three models for CGN Component G. The results of this hyperparameter tuning are illustrated in Table 5.2.1.

Table 5.2.1 shows relatively small differences in WER between Faster-Whisper and WhisperX, ranging from 0.171 to 0.173, while the baseline Whisper has higher WER values of 0.179 and 0.176. WhisperX outperforms the other models in precision with a value of 0.924. Faster-Whisper, with *condition_on_previous_text* set to False and *VAD_filter* set to Off, achieves the highest recall value of 0.838. WhisperX consistently achieves the highest F1-score of 0.878 in three of the four settings. Additionally, WhisperX demonstrates the best RTF performance with a value of 0.047.

Table 5.2.1: Performance metrics CGN Component G, thresholds in VAD are in default settings: for Faster-Whisper: 0.500, for WhisperX: Onset = 0.500, Offset = 0.363

			CGN Comp G				
Model	**cond_on_prev_text**	**VAD**	*WER*	*Precision*	*Recall*	*F1*	*RTF*
Whisper (=def)	True	N/A	0.179	0.914	0.832	0.870	0.161
WhisperX	True	On	**0.171**	**0.924**	0.837	**0.878**	0.051
WhisperX	True	Off	**0.171**	**0.924**	0.837	**0.878**	0.054
Faster-Whisper	True	On	0.173	0.922	0.836	0.876	0.087
Faster-Whisper (=def)	True	Off	0.173	0.920	0.837	0.876	0.065
Whisper	False	N/A	0.176	0.914	0.834	0.872	0.153
WhisperX (=def)	False	On	0.172	0.923	0.836	0.877	**0.047**
WhisperX	False	Off	**0.171**	**0.924**	0.837	**0.878**	0.053
Faster-Whisper	False	On	**0.171**	0.920	**0.838**	0.877	0.134
Faster-Whisper	False	Off	0.171	0.920	**0.838**	0.877	0.058

5.3 RESULTS ON NFI-FRITS

This section presents the results of the experiments of Whisper, Faster-Whisper and WhisperX on the NFI-FRITS dataset, which consists of recordings of telephone speech intercepted by Dutch law officials during police investigations.

Figure 5.3.1 illustrates the performance of the three models in terms of the WER on the Y-axis and the RTF on the X-axis. Remarkably, three predictions of the three models yielded a WER higher than the value of 4, originating from a single audio file. Further investigation revealed that the speakers in that audio file had a distinct accent, making it challenging to comprehend the spoken content, even for a human. Considering the difficulties encountered while listening to this file, it is reasonable to expect similar challenges for ASR models, particularly in the case of Dutch as a low-resource language. Consequently, this outlier audio file has been excluded from 5.3.1.

As can be seen in Figure 5.3.1, WhisperX outperforms Faster-Whisper and Whisper on the RTF. WhisperX achieves a RTF of 0.04, while Faster-Whisper and Whisper exhibit RTF values of 0.083 and 0.235, respectively. This indicates that WhisperX transcribes the audio files of NFI-FRITS about five times faster than the baseline model Whisper. Moreover, WhisperX exceeds Faster-Whisper and Whisper in terms of WER, with WER values of 0.448, 0.483, and 0.568 respectively (see Table 5.3.1). These findings indicate that WhisperX is the most effective model for this particular dataset, according to its superior WER and RTF performance when compared to Faster-Whisper and Whisper.

Similar to the analysis performed on CGN Components C and G, the hyperparameters *condition_on_previous_text* and *VAD_filter* are adjusted for the three models in the experiments on the NFI-FRITS dataset. The impact of this hyperparameter tuning is presented in Table 5.3.1. As can be seen in Table 5.3.1, WhisperX with *condition_on_previous_text* set to True and *VAD_filter* set to On demonstrates better performance in terms of WER, precision, recall, and F1-score for this dataset. The WER of this model is 0.443, the precision is 0.697, the recall value is 0.645, and the F1-score is 0.668. The best model in terms of RTF is WhisperX in its default setting, outperforming the other models with a RTF of 0.040. This is followed by WhisperX

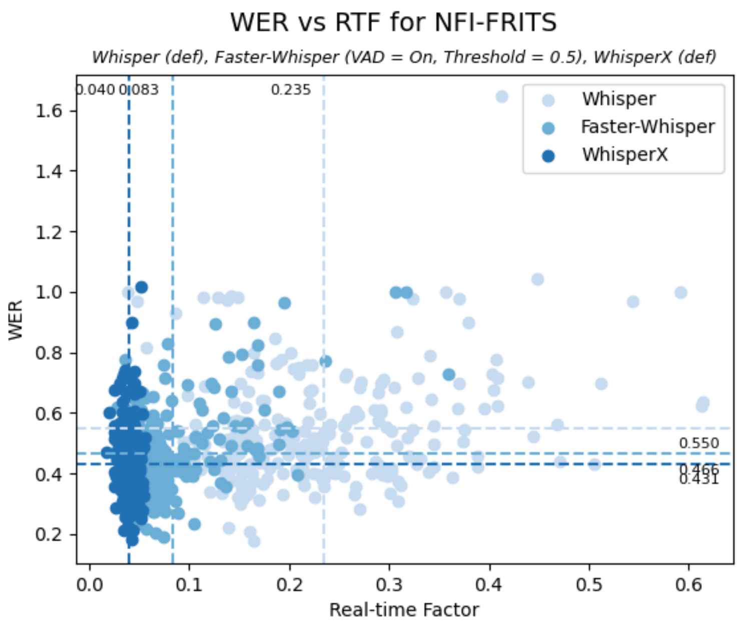


Figure 5.3.1: WER vs RTF on NFI-FRITS for Whisper, Faster-Whisper, and WhisperX. Outliers removed.

Table 5.3.1: Performance metrics NFI-FRITS, thresholds in VAD are in default settings: for Faster-Whisper: 0.500, for WhisperX: Onset = 0.500, Offset = 0.363

			NFI-FRITS				
Model	**cond_on_prev_text**	**VAD**	*WER*	*Precision*	*Recall*	*F1*	*RTF*
Whisper (=def)	True	N/A	0.568	0.602	0.550	0.569	0.235
WhisperX	True	On	**0.443**	**0.697**	**0.645**	**0.668**	0.053
WhisperX	True	Off	0.460	0.688	0.626	0.653	0.051
Faster-Whisper	True	On	0.483	0.666	0.620	0.639	0.083
Faster-Whisper (=def)	True	Off	0.561	0.591	0.534	0.555	0.163
Whisper	False	N/A	0.543	0.616	0.573	0.590	0.276
WhisperX (=def)	False	On	0.448	0.695	**0.645**	0.667	**0.040**
WhisperX	False	Off	0.468	0.685	0.625	0.651	0.050
Faster-Whisper	False	On	0.477	0.669	0.629	0.646	0.065
Faster-Whisper	False	Off	0.545	0.603	0.562	0.577	0.121

with *condition_on_previous_text* set to False and *VAD_filter* set to Off, having a RTF of 0.050, and WhisperX with *condition_on_previous_text* set to True and *VAD_filter* set to Off, having a RTF of 0.051.

5.4 RESULTS OF THRESHOLD TUNING

This section discusses the results of tuning the threshold value in the voice activity detection model of Faster-Whisper and WhisperX. An overview of all results of the threshold tuning can be found in the appendixes. Appendix A presents the results for CGN Component C, Appendix B for CGN Component G, and Appendix C for NFI-FRITS.

The performance of Faster-Whisper and WhisperX on CGN Component C does not show a clear trend when tuning the threshold values, as can be seen in Appendix A. While Faster-Whisper achieves better WERs and recall values with lower thresholds, WhisperX achieves worse WERs and recall values with lower thresholds. However, regarding the precision, higher thresholds in both models lead to higher values. Finally, regarding RTF only Faster-Whisper shows the trend that lower thresholds lead to a lower RTF.

The relationship between tuning the threshold values and performance metrics for CGN Component G is uncertain. While Faster-Whisper generally performs better with lower threshold values across the performance metrics WER, precision, and F1-score, the impact of tuning threshold values on WhisperX is unclear, as shown in Appendix B. Lower threshold values tend to result in the lowest WER for WhisperX, but higher threshold values may improve precision scores. The effects on recall, F1 scores, and RTF exhibit inconsistent patterns and are inconclusive.

Finally, for the NFI-FRITS dataset, higher threshold values generally lead to better results for both Faster-Whisper and WhisperX (see Appendix C. In Faster-Whisper, the lowest WER of 0.469 is achieved with a threshold of 0.75. For WhisperX, higher thresholds result in higher precision, recall, and F1-scores. Additionally, both models exhibit improved RTF performance with higher threshold values.

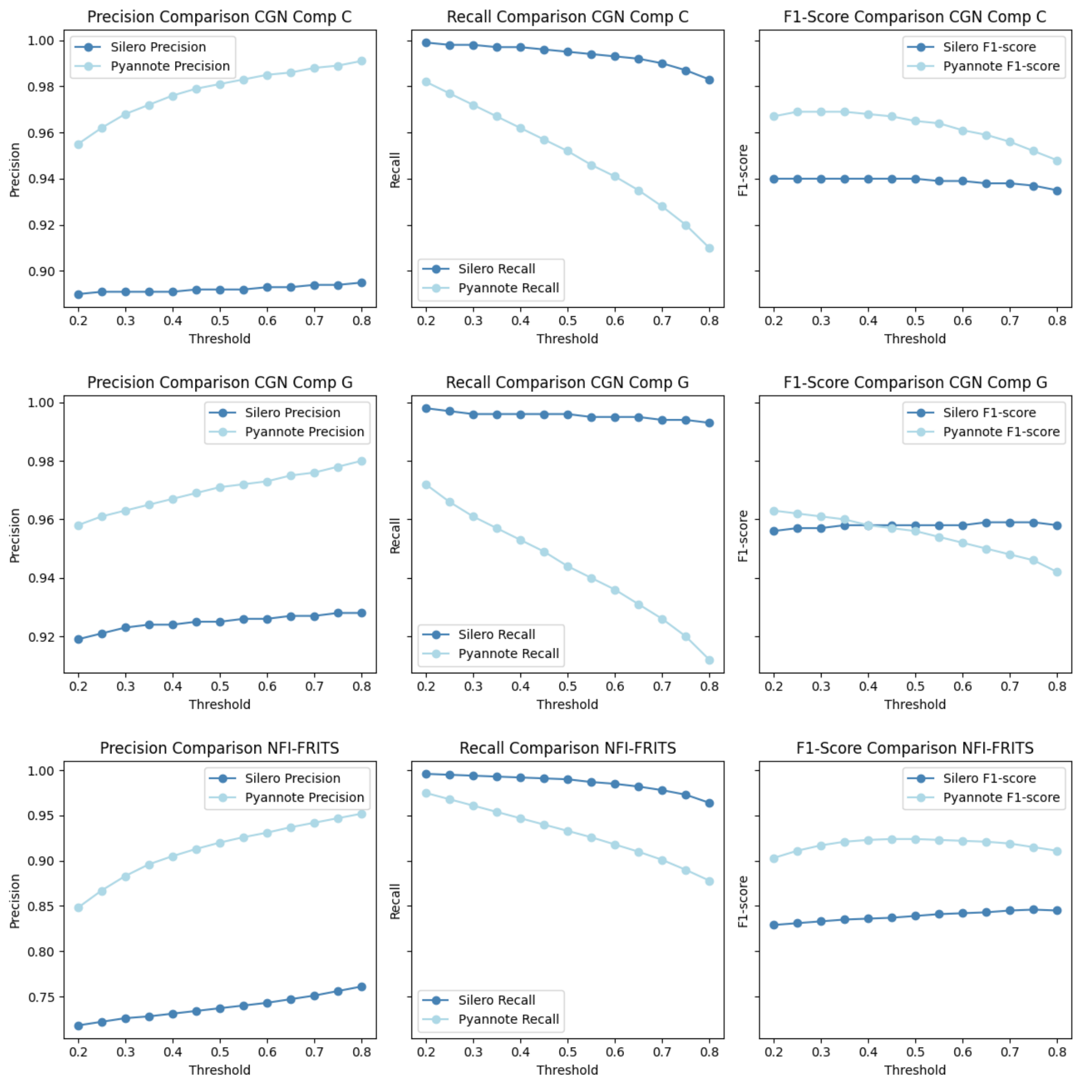
Although the adjustment of threshold values in Faster-Whisper and WhisperX has some impact across the different performance metrics, the observed variations are relatively small. For example, in CGN Component C, the most significant discrepancy between threshold values (e.g. 0.25 and 0.80) in Faster-Whisper on the WER is 0.09, while for WhisperX, the discrepancy in WER between threshold values 0.20 and 0.55 is 0.003. In CGN Component G, the differences are even smaller. For example, the precision metric shows a difference of 0.001 across all threshold values in Faster-Whisper, while in WhisperX the discrepancy in recall values is 0.002. Overall, the influence of different threshold values on the performance metrics is most outspoken in the NFI-FRITS dataset. For instance, in Faster-Whisper, there is a difference of 0.027 in precision when comparing threshold values of 0.20 and 0.70. Similarly, in WhisperX, a difference of 0.005 in recall is observed when comparing threshold values of 0.30 and 0.50.

5.5 RESULTS SILERO AND PYANNOTE

In addition to the experiments with Whisper, Faster-Whisper, and WhisperX, experiments are also conducted to test the performance of the voice activity detection models only. As aforementioned, Faster-Whisper incorporates Silero VAD, while WhisperX incorporates PyAnnote VAD. The predicted timestamps by Silero and PyAnnote VAD are compared to the original timestamps as present in the original transcriptions, using the performance metrics precision, recall, and F1-score. In these experiments, the threshold values are changed. An overview of all results can be found in Appendix D, E, and F. In this section, only the main findings are presented, and summarized in Figure 5.5.1.

As can be seen in Figure 5.5.1, regarding the precision metric, its value increases with higher thresholds in all datasets. This pattern applies for both the Silero and PyAnnote VAD models. In each dataset, the PyAnnote VAD model consistently achieves higher precision values. Conversely, the recall metric shows a decreasing trend with higher thresholds across all cases. When comparing the two VAD models, the recall value of the PyAnnote model is consistently lower than that of the Silero model across all datasets. Lastly, in terms of F1-scores, no general

Figure 5.5.1: Precision, Recall, and F1-score of Silero and PyAnnote VAD for different datasets



trend is observed across the datasets. For CGN Component C, both models exhibit decreasing F1-scores with higher thresholds. However, for CGN Component G, an opposite trend is observed, where the PyAnnote VAD model shows a decreasing recall value with higher thresholds, while the Silero VAD model exhibits an increasing trend. In the NFI-FRITS dataset, the recall value of the PyAnnote VAD model is consistently higher than that of the Silero VAD model across all threshold values.

6 DISCUSSION

The purpose of this chapter is to discuss the meaning of the research results as presented in Chapter 5. As described in Chapter 1, the goal of this research is to assess the performance of Whisper and two re-implementations, Faster-Whisper and WhisperX, that incorporate the pre-processing technique of voice activity detection on Dutch long-form audio data. This section interprets the results of the different experiments conducted in this research and discusses how the results contribute to answering the research question by assessing the performance of Whisper, Faster-Whisper and WhisperX. This chapter is organized as follows. First, in Sections 6.1, 6.2, and 6.3, the performance results of Whisper and its re-implementations are discussed in further detail. After that, Section 6.4 discusses the interpretation of the limited effects of tuning the threshold values within the voice activity detection models. Section 6.5 discusses the potential explanations for the different results of the two voice activity detection models Silero and PyAnnote. Finally, in Section 6.6, the main shortcomings and limitations of this research are elaborated on.

6.1 PERFORMANCE OF WHISPER, FASTER-WHISPER, AND WHISPERX ON WORD ERROR RATE

By incorporating a voice activity detection model, two primary effects are anticipated that both influence the WER. Firstly, as fewer segments of the audio file require transcription, it is expected that the occurrence of hits, substitutions, and insertions will decrease, while the number of deletions will increase. Secondly, by filtering out segments of poor audio quality, the likelihood of hallucination, i.e., predicting words where none exist in the audio file, will be reduced. Consequently, one can expect that the number of deletions and insertions will decrease, while the number of hits will increase. The impact on the number of substitutions remains unclear. In theory these effects can counteract each other, however, since the exact increase or decrease of the number of deletions, insertions, substitutions, and hits was not tracked accurately, it is hard to say which effect is stronger. However, since both effects result in a decrease in insertions, one could hypothesise that an overall decrease in the WER can be the result of the implementation of a voice activity detection model.

The hypothesis mentioned above is confirmed in the obtained results, as presented in Table 5.1.1, 5.2.1 and 5.3.1, since it shows that the WER decreases for Faster-Whisper and WhisperX compared to Whisper. However, considerable variations in WER still occur across the different datasets examined. Specifically, the NFI-FRITS dataset exhibits substantially higher WER values, approximately 0.5 on average, followed by CGN Component C with an average WER of around 0.332, while CGN Component G displays the lowest WER, averaging around 0.173. This deviation can potentially be attributed to differences in audio quality between datasets. In particular, the NFI-FRITS dataset contains audio files characterized by relatively reduced clarity when compared to the relatively cleaner audio files in CGN Component C and G. Furthermore, the audio files in CGN Component C and G appear to be more staged, with mostly intelligible speech, whereas the NFI-FRITS dataset comprises recordings of individuals unaware of being recorded, resulting in lower audio quality due to their lack of concern.

Regarding the contrasting WERs observed between CGN Component C and G, it appears that Component G exhibits fewer instances of overlapping speech (i.e., speeches rather than conversations) and fewer speaker switches, making the ASR model less prone to mistakes.

6.2 PERFORMANCE OF WHISPER, FASTER-WHISPER, AND WHISPERX ON PRECISION, RECALL, AND F1-SCORE

By employing the voice activity detection model, the ASR model is also expected to achieve a higher proportion of accurate transcriptions among all positive predictions, leading to higher precision. The reason for this is that by employing a voice activity detection model, low-quality segments are filtered out, making it easier for the ASR model to transcribe the audio segments leading to a higher proportion of correct positive predictions. The experimental findings presented in Table 5.1.1, 5.2.1 and 5.3.1, consistently demonstrate that, overall, Faster-Whisper and WhisperX exhibit higher precision compared to the baseline Whisper model. Moreover, when comparing the same model with only the *VAD_filter* enabled and disabled, in most cases, the enabled models demonstrate higher precision scores. These results align with the initial expectations of achieving higher precision scores by the incorporation of a voice activity detection model and indicate that out of all predicted words by Faster-Whisper and WhisperX, a high proportion of them are correct.

However, by incorporating a voice activity detection model, some spoken words in the audio file may not be recognized as speech and subsequently omitted by the ASR model. Consequently, a relatively higher number of false negatives is expected, leading to a decrease in the recall value. Contrary to expectations, the experimental results as illustrated in Table 5.1.1, 5.2.1 and 5.3.1, indicate an increase in recall for almost all experiments where the *VAD_filter* is enabled. This unexpected outcome can potentially be attributed to the voice activity detection model effectively filtering out low-quality segments of the audio file, identifying them as non-speech. Consequently, the voice activity detection models in Faster-Whisper and WhisperX avoid transcribing these degraded segments and mitigates the occurrence of failure loops. As a result, the overall recall value for the audio segment improves. These findings are supported by the observation that when the *VAD_filter* is disabled, certain audio files showed highly anomalous transcriptions at segments in the beginning of the audio file. This poses substantial challenges in attaining accurate transcription for the entirety of the audio file, thereby resulting in a relatively low recall value. By pre-segmentation of these specific segments in advance, there is a possibility of potentially achieving an overall higher recall value, as observed in the experimental results of this research.

Based on the F1-score formula, since precision and recall are both in the numerator and denominator, F1-score can either increase or decrease when the precision is increased and the recall is decreased. However, since the results show that the recall increases as well, the F1-score is also expected to increase since the effect of the numerator is larger than the effect of the denominator. This trend is clearly showed throughout the results as Faster-Whisper and WhisperX show higher for F1-scores when their precision and recall values increase compared to the baseline Whisper.

6.3 PERFORMANCE OF WHISPER, FASTER-WHISPER, AND WHISPERX ON REAL-TIME FACTOR

Finally, it is hypothesized that the integration of a voice activity detection model into Whisper would lead to a reduction in RTF since there are fewer speech segments to transcribe for the ASR model.

The experimental results presented in Table 5.1.1, 5.2.1 and 5.3.1, demonstrate that both Faster-Whisper and WhisperX exhibit lower RTF values compared to the baseline model Whisper, across various settings and in all datasets. When the *VAD_filter* is enabled (set to On) for Faster-Whisper and WhisperX, the RTF is lower than that of Whisper, aligning with the initial hypothesis. This indicates that both Faster-Whisper and WhisperX transcribe the audio files faster than the baseline model Whisper. Additionally, even when the *VAD_filter* is disabled (set to Off) for Faster-Whisper and WhisperX, the RTF remains significantly lower than that of the baseline model Whisper, which was expected initially, since Faster-Whisper and WhisperX are implemented using CTranslate. Besides, a few cases exhibit an even lower RTF when the *VAD_filter* is disabled, when compared to the same model with the *VAD_filter* enabled. A possible explanation for this is the simultaneous execution of multiple experiments effecting the RTF, as will be discussed in Section 6.6.

6.4 THE EFFECTS OF THRESHOLD TUNING WITHIN FASTER-WHISPER AND WHISPERX

In addition to comparing Whisper and its re-implementations across various settings, experiments are conducted to tune the threshold value of the voice activity detection model in Faster-Whisper and WhisperX. One can expect that increasing the threshold value would generally lead to a lower RTF, as fewer speech segments require transcription. Furthermore, it was anticipated that a higher threshold value would result in a larger decrease in WER compared to a lower threshold. Finally, as fewer speech segments require transcriptions when the thresholds increases, it was expected that the effects of an increasing precision and decreasing in recall as described before, would be stronger when compared to lower thresholds.

In general, the results show that the performance of both models remains relatively stable when evaluated at various threshold values. This implies that the selection of a specific threshold has a limited impact on the model's performance across different datasets. It seems that these results can be explained by two effects that cancel each other out. For lower thresholds, there are more audio segments considered as speech, but the quality of each segment is lower. For Whisper, this means more audio segments to transcribe (leading to a higher recall), but also more change to get stuck in failure loops or hallucinations (leading to a lower recall). At the same time, for higher thresholds, there are less audio segments considered as speech, but the quality of each segment is higher. For Whisper, this means less audio segments to transcribe (leading to lower recall), but also more less to get stuck in failure loops or hallucinations (leading to a higher recall).

Although minor performance differences exist across diverse thresholds, it is indefinable whether a relatively high or low threshold value is optimal, as the ideal threshold varies among datasets and depends on the prioritized performance metric. Consequently, finetuning the threshold hyperparameter in the voice activity detection model yields slight variations in performance outcomes at different threshold values. Notably, the differences between threshold values are most distinct in the NFI-FRITS dataset (see Appendix C), indicating that finetuning the threshold value in the voice activity detection model for this particular dataset may have a bigger impact on the performance metrics compared to the other datasets. Generally, for the NFI-FRITS dataset, higher thresholds for both Faster-Whisper (0.75) and WhisperX (0.55, 0.60, 0.70) demonstrate better performance when considering all performance metrics as equally important.

The observed difference in the impact of threshold tuning on performance metrics between the NFI-FRITS dataset and CGN, may be attributed to factors that are different between the datasets, such as the level of background noise and other interference's. While both NFI-FRITS and CGN Component C datasets involve telephone conversations, the NFI-FRITS dataset, derived from real police investigations, may exhibit higher levels of background noise and interference's compared to CGN Component C. Consequently, the NFI-FRITS dataset would likely require more precise threshold tuning in the VAD models to effectively separate speech from noise, resulting in a greater impact on performance metrics.

6.5 THE EFFECTS OF THRESHOLD TUNING IN Silero VAD AND PyAnnote VAD

Given that WhisperX generally outperforms Faster-Whisper in terms of Word Error Rate, precision, recall, F1-score, and Real-Time Factor, it was hypothesized that this discrepancy may be attributed to the usage of different voice activity detection models by Faster-Whisper and WhisperX. To further investigate this hypothesis, additional experiments are conducted to assess the performance disparities between the Silero VAD and PyAnnote VAD models, which are employed by Faster-Whisper and WhisperX, respectively. It was expected that, in general, the PyAnnote VAD model would exhibit better performance compared to the Silero VAD model, considering the integration of PyAnnote within the WhisperX system.

Overall, the precision scores for both models and across all datasets exhibit an increasing trend with higher thresholds, aligning with the aforementioned hypothesis. However, just like for threshold tuning in the section above, the differences in performance between tuning the threshold values are limited. Considering the consistently lower precision values of the Silero VAD model across all datasets, it can be inferred that the Silero VAD model tends to consider more segments as speech compared to the PyAnnote VAD model. This observation finds support in the determined detected speech ratios presented in Appendices D, E, and F. The detected speech ratio refers to the proportion of positive predictions, which correspond to segments considered as speech by the voice activity detection model, divided by the total speech segments indicated in the original transcription. A speech ratio exceeding 1 indicates that the voice activity detection model identifies more segments as speech than actually exist in the audio file. In general, the detected speech ratios for Silero VAD are consistently higher across all thresholds compared to the detected speech ratios obtained with PyAnnote VAD, supporting the aforementioned observation of higher precision values for Silero VAD. Furthermore, it can be seen that higher thresholds lead to a decrease in detected speech ratio for both models, which confirms that threshold tuning affects the number of segments that are considered as speech. In contrast to previously discussed findings in Section 6.4, adjusting the threshold value within the voice activity detection models, apart from the ASR models, has clear effects. Increasing the threshold value demonstrates an improvement in precision scores, across all datasets and for both Silero and PyAnnote VAD models.

Additionally, the recall value decreases for both voice activity detection models and in all three datasets as the threshold increases, which is in line with expectations. By higher thresholds, more segments are considered as non-speech, which leads to more false negatives, and therefore a lower recall value. However, this is in contrast with the findings in Section 5.4, which show limited to no effect on the recall when adjusting the threshold value. Additionally, there is no clear upward or downward trend observable at an increasing or decreasing threshold. As mentioned in Section 5.5, he recall value is lower for PyAnnote VAD across all datasets and thresholds than for Silero VAD, leading to the conclusion that PyAnnote VAD is more cautious about detecting segments as speech when compared to Silero VAD.

Finally, regarding the F1-scores, there is a different trend between the CGN Component C and NFI-FRITS and CGN Component G. more specific, the F1-score for CGN Component G of both Silero and PyAnnote VAD crosses at a threshold of 0.4, after which the F1-score of Silero VAD is higher than the PyAnnote F1-score, which is in contrast with the findings in the other datasets. Intuitively, one could be surprised by the results, but given the formula of the F1-score, this finding can be explained due to the relative increase and decrease between precision and recall for CGN Comp G.

6.6 SHORTCOMINGS AND LIMITATIONS

With regard to shortcomings in this research, there are three main limitations that need to be addressed. To start with, a limitation arises from the considerable amount of time dedicated to debugging the WhisperX algorithm. Throughout the execution of the experiments, it was discovered that, regardless of the hyperparameters tuned, WhisperX consistently returned identical results. It turned out that the hyperparameters were updated, but not utilized in the source code, which was a fundamental mistake by the designer of the model. Consequently, due to the substantial time consumed by this issue, there was insufficient time available to execute the experiments multiple times and compute the average output.

Another limitation of this research relates to not monitoring the memory usage of the various models employed. Although WhisperX demonstrates promising results across diverse performance metrics, it appears from preliminary experiments to consume more memory compared to Faster-Whisper. Monitoring the memory usage would have provided a more comprehensive understanding of the models' performance. However, due to the simultaneous execution of multiple programs, accurate monitoring of memory usage was not possible. The simultaneous execution of multiple programs during the research leads to another limitation, namely that the RTF may not be fully representative, as this simultaneous execution may have affected the RTF. Therefore, the RTF values in this study research not 100% reliable.

Other smaller limitations include the potential inadequacy of the JiWER package in scenarios where transcriptions are missing or numerous. Moreover, the package treats misspelled words as errors, even though the biggest part of the word is identical and the content is correct. Finally, despite pre-processing of the transcriptions, certain undesired annotations such as "peep" may remain. This leads to a potential higher number of detected errors, resulting in a blurred representation of the performance of the models .

7 CONCLUSION

This work aimed to assess the performance of Whisper and two re-implementations called Faster-Whisper and WhisperX that both incorporate the pre-processing technique voice activity detection. This chapter is organized as follows. In Section 7.1, the contributions of this work and the main findings are discussed. Next, in Section 7.2, based on the main conclusions, a number of recommendations for future work are mentioned.

7.1 CONCLUSION

Whisper, an automatic speech recognition model, demonstrates state-of-the-art performance across multiple tasks and languages. Although the accuracy of the Whisper output is quite high, the problem lies in its extensive execution time. TROI, a department of the Dutch National Police, is looking for ways to optimize this model in terms of speed without comprising its accuracy. Given Whisper's minimalist strategy for data pre-processing, it is expected that incorporating advanced pre-processing techniques could further optimize its performance. Accordingly, this study aims to investigate the influence of incorporating the pre-processing technique voice activity detection on Whisper's performance. Therefore, the main research question is:

> *How do state-of-the-art enhanced Whisper models including the pre-processing technique voice activity detection perform compared to the baseline model Whisper on Dutch long-form audio in terms of Word Error Rate, precision, recall, F1-score, and Real-Time Factor?*

In this research, a comprehensive analysis between Whisper and two re-implementations called Faster-Whisper and WhisperX, is conducted. Both Faster-Whisper and WhisperX integrate a pre-processing technique known as voice activity detection. A voice activity detection model provides a probability for each segment in an audio file, indicating the presence or absence of speech. Consequently, specific parts of an audio file are filtered out prior to being processed by the ASR model. This process reduces the number of audio segments that the ASR model transcribes. This research assesses whether the incorporation of a voice activity detection model into Whisper leads to an accelerated transcription speed, measured by the Real-Time Factor. Additionally, the research examines the impact of this approach on various performance metrics, including the Word Error Rate, precision, recall, and F1-score.

To reach the goal of this research, a literature review is conducted on Whisper and its re-implementations. Subsequently, a series of experiments is conducted to compare the performance of Whisper, Faster-Whisper, and WhisperX under default settings, utilizing various performance metrics. Additionally, different hyperparameters within the models, namely *condition_on_previous_text*, *VAD_filter*, and threshold, are tuned to evaluate their impact on the performance metrics. Moreover, the performance of the voice activity detection models, specifically Silero employed in Faster-Whisper and PyAnnote utilized in WhisperX, is assessed through experiments conducted solely on the timestamps generated by these models. The precision and recall values are used to evaluate the performance of the voice activity detection models.

In conclusion, both Faster-Whisper and WhisperX demonstrate better performance compared to the original Whisper algorithm, as they effectively enhance the execution time measured by the RTF and improve various performance metrics, namely the WER, precision, recall, and F1-score. This indicates the advantages of incorporating a voice activity detection model within the Whisper framework. Despite the expectation of lower recall values due to potential omission of speech segments, the recall value actually increases in nearly all cases for Faster-Whisper and WhisperX compared to the baseline Whisper model, contrary to initial expectations. WhisperX is, in most cases, outperforming Faster-Whisper across different datasets and settings.

The impact of tuning the threshold values Faster-Whisper and WhisperX is limited. By adjusting these threshold values, only minor differences in performance are observed. Furthermore, it was observed that finetuning the threshold values is dependent on the specific dataset. Therefore, it can be concluded that incorporating of a voice activity detection model yields more benefits in terms of improving on the various performance metrics compared to the effect of finetuning the threshold value within this model.

Finally, it can be concluded that the dissimilarities in performance between Faster-Whisper and WhisperX can be primarily attributed to their utilization of distinct voice activity detection models. The Silero VAD model as incorporated in Faster-Whisper is achieving higher recall values, whereas the PyAnnote VAD model utilized by WhisperX is achieving higher precision values.

7.2 RECOMMENDATIONS FOR FUTURE WORK

A few suggestions can be made for future work:

- **Examine alternative voice activity detection models**
  Future work could involve evaluating alternative voice activity detection models, such as SpeechBrain[1] or NeMo[2], alongside Whisper. This would enable an assessment of the impact of different voice activity detection models on the overall performance of the model.

- **Test memory usage**
  WhisperX exhibits promising results across different performance metrics. Nevertheless, additional research is needed to compare the memory usage of various implementations of Whisper. Preliminary observations suggest that WhisperX consumes a higher amount of memory compared to Faster-Whisper, thereby positioning Faster-Whisper as an attractive choice in scenarios with limitations on resources.

- **Investigate differences between the models Faster-Whisper and WhisperX**
  Based on the different results observed, where adjusting the thresholds in Faster-Whisper and WhisperX showed no convincing effects, while similar adjustments in Silero and PyAnnote VAD models, apart from the ASR models, showed clear trends, it is plausible that inherent differences between Faster-Whisper and WhisperX contribute to these differences in outcomes. However, investigating these differences was beyond the scope of the current study. It is therefore recommended to investigate these variations in future research.

1 https://speechbrain.github.io/
2 https://docs.nvidia.com/deeplearning/nemo/user-guide/docs/en/stable/asr/speech_classification/models.html

- **Incorporate other pre-processing techniques**
  As discussed in Section 2.1, various pre-processing techniques, such as noise removal, can be employed within an ASR model. Future research could focus on exploring the effects of integrating various pre-processing techniques into Whisper or a re-implementation. It is expected that incorporating additional pre-processing techniques can further improve the performance of the ASR model. However, the effectiveness of these techniques depends on the characteristics of the dataset used to test the ASR model, as was also found in this research.

BIBLIOGRAPHY

[1] A. Dhouib, A. Othman, O. El Ghoul, M. K. Khribi, and A. Al Sinani, "Arabic automatic speech recognition: A systematic literature review," vol. 12, no. 17, p. 8898. [Online]. Available: https://www.mdpi.com/2076-3417/12/17/8898

[2] Y. A. Ibrahim, J. C. Odiketa, and T. S. Ibiyemi, "Preprocessing technique in automatic speech recognition for human computer interaction: An overview."

[3] S. Alharbi, M. Alrazgan, A. Alrashed, T. Alnomasi, R. Almojel, R. Alharbi, S. Alharbi, S. Alturki, F. Alshehri, and M. Almojil, "Automatic speech recognition: Systematic literature review," vol. 9, pp. 131 858–131 876. [Online]. Available: https://ieeexplore.ieee.org/document/9536732/

[4] S. Sen, A. Dutta, and N. Dey, *Audio Processing and Speech Recognition: Concepts, Techniques and Research Overviews*, ser. SpringerBriefs in Applied Sciences and Technology. Springer Singapore. [Online]. Available: http://link.springer.com/10.1007/978-981-13-6098-5

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

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need." [Online]. Available: http://arxiv.org/abs/1706.03762

[7] M. Kadlčík, A. Hájek, J. Kieslich, and R. Winiecki, "A whisper transformer for audio captioning trained with synthetic captions and transfer learning." [Online]. Available: http://arxiv.org/abs/2305.09690

[8] R. Olivier and B. Raj, "There is more than one kind of robustness: Fooling whisper with adversarial examples." [Online]. Available: http://arxiv.org/abs/2210.17316

[9] L. R. S. Gris, R. Marcacini, A. C. Junior, E. Casanova, A. Soares, and S. M. Aluísio, "Evaluating OpenAI's whisper ASR for punctuation prediction and topic modeling of life histories of the museum of the person." [Online]. Available: http://arxiv.org/abs/2305.14580

[10] A. Baevski, H. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations." [Online]. Available: http://arxiv.org/abs/2006.11477

[11] M. Malik, M. K. Malik, K. Mehmood, and I. Makhdoom, "Automatic speech recognition: a survey," vol. 80, no. 6, pp. 9411–9457. [Online]. Available: http://link.springer.com/10.1007/s11042-020-10073-7

[12] M. Labied, A. Belangour, M. Banane, and A. Erraissi, "An overview of automatic speech recognition preprocessing techniques," in *2022 International Conference on Decision Aid Sciences and Applications (DASA)*. IEEE, pp. 804–809. [Online]. Available: https://ieeexplore.ieee.org/document/9765043/

[13] A. Keerio, B. K. Mitra, P. Birch, R. Young, and C. Chatwin, "On preprocessing of speech signals."

[14] A. Y., K. A., Y. A., and N. Pandya, "Survey paper on different speech recognition algorithm: Challenges and techniques," vol. 175, no. 1, pp. 31–36. [Online]. Available: http://www.ijcaonline.org/archives/volume175/number1/vadwala-2017-ijca-915472.pdf

[15] M. Labied and A. Belangour, "Automatic speech recognition features extraction techniques: A multi-criteria comparison," vol. 12, no. 8. [Online]. Available: http://thesai.org/Publications/ViewPaper?Volume=12&Issue=8&Code=IJACSA&SerialNo=21

[16] M. B. Akçay and K. Oğuz, "Speech emotion recognition: Emotional models, databases, features, preprocessing methods, supporting modalities, and classifiers," vol. 116, pp. 56–76. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0167639319302262

[17] A. Ivry, B. Berdugo, and I. Cohen, "Voice activity detection for transient noisy environment based on diffusion nets," vol. 13, no. 2, pp. 254–264. [Online]. Available: http://arxiv.org/abs/2106.13763

[18] "Research and development in intelligent systems XXXI: Incorporating applications and innovations in intelligent systems XXII." [Online]. Available: https://link.springer.com/10.1007/978-3-319-12069-0

[19] T. Mishra, A. Ljolje, and M. Gilbert, "Predicting human perceived accuracy of ASR systems," in *Interspeech 2011*. ISCA, pp. 1945–1948. [Online]. Available: https://www.isca-speech.org/archive/interspeech_2011/mishra11_interspeech.html

[20] R. Errattahi, A. El Hannani, and H. Ouahmane, "Automatic speech recognition errors detection and correction: A review," vol. 128, pp. 32–37. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877050918302187

[21] A. Ali and S. Renals, "Word error rate estimation for speech recognition: e-WER," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pp. 20–24. [Online]. Available: https://aclanthology.org/P18-2004

[22] M. Bain, J. Huh, T. Han, and A. Zisserman, "WhisperX: Time-accurate speech transcription of long-form audio." [Online]. Available: http://arxiv.org/abs/2303.00747

[23] S.-J. Lee and H.-Y. Kwon, "A preprocessing strategy for denoising of speech data based on speech segment detection," vol. 10, no. 20, p. 7385. [Online]. Available: https://www.mdpi.com/2076-3417/10/20/7385

[24] S. Yadav, P. A. D. Legaspi, M. S. O. Alink, A. B. J. Kokkeler, and B. Nauta, "Hardware implementations for voice activity detection: Trends, challenges and outlook," vol. 70, no. 3, pp. 1083–1096. [Online]. Available: https://ieeexplore.ieee.org/document/9976339/

[25] N. Usha Rani and P. N. Girija, "Error analysis and improving the speech recognition accuracy on telugu language," in *Advances in Communication, Network, and Computing*, V. V. Das and J. Stephen, Eds. Springer Berlin Heidelberg, vol. 108, pp. 301–308, series Title: Lecture Notes of the Institute for Computer

Sciences, Social Informatics and Telecommunications Engineering. [Online]. Available: http://link.springer.com/10.1007/978-3-642-35615-5_46

[26] V. Pratap, Q. Xu, A. Sriram, G. Synnaeve, and R. Collobert, "Mls: A large-scale multilingual dataset for speech research," *arXiv preprint arXiv:2012.03411*, 2020.

[27] 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," *arXiv preprint arXiv:1912.06670*, 2019.

[28] C. Wang, M. Riviere, A. Lee, A. Wu, C. Talnikar, D. Haziza, M. Williamson, J. Pino, and E. Dupoux, "Voxpopuli: A large-scale multilingual speech corpus for representation learning, semi-supervised learning and interpretation," *arXiv preprint arXiv:2101.00390*, 2021.

[29] A. Conneau, M. Ma, S. Khanuja, Y. Zhang, V. Axelrod, S. Dalmia, J. Riesa, C. Rivera, and A. Bapna, "Fleurs: Few-shot learning evaluation of universal representations of speech," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 798–805.

[30] C. Wang, J. Pino, A. Wu, and J. Gu, "Covost: A diverse multilingual speech-to-text translation corpus," *arXiv preprint arXiv:2002.01320*, 2020.

[31]

[32] H. Bredin, R. Yin, J. M. Coria, G. Gelly, P. Korshunov, M. Lavechin, D. Fustes, H. Titeux, W. Bouaziz, and M.-P. Gill, "pyannote.audio: neural building blocks for speaker diarization." [Online]. Available: http://arxiv.org/abs/1911.01255

[33] A. Sherstinsky, "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network," vol. 404, p. 132306. [Online]. Available: http://arxiv.org/abs/1808.03314

[34] H. Bredin and A. Laurent, "End-to-end speaker segmentation for overlap-aware resegmentation." [Online]. Available: http://arxiv.org/abs/2104.04045

[35] J. Carletta, S. Ashby, S. Bourban, M. Flynn, M. Guillemot, T. Hain, J. Kadlec, V. Karaiskos, W. Kraaij, M. Kronenthal *et al.*, "The ami meeting corpus: A pre-announcement," in *International workshop on machine learning for multimodal interaction*. Springer, 2005, pp. 28–39.

[36] J. Godfrey and E. Holliman, "Switchboard-1 release 2 ldc97s62," *Linguistic Data Consortium*, p. 34, 1993.

[37] F. Hernandez, V. Nguyen, S. Ghannay, N. Tomashenko, and Y. Esteve, "Ted-lium 3: Twice as much data and corpus repartition for experiments on speaker adaptation," in *Speech and Computer: 20th International Conference, SPECOM 2018, Leipzig, Germany, September 18–22, 2018, Proceedings 20*. Springer, 2018, pp. 198–208.

[38] I. Schuurman, M. Schouppe, H. Hoekstra, and T. Van der Wouden, "Cgn, an annotated corpus of spoken dutch," in *Proceedings of 4th International Workshop on Linguistically Interpreted Corpora (LINC-03) at EACL 2003*, 2003.

[39] D. Van Der Vloed, J. Bouten, and D. Van Leeuwen, "NFI-FRITS: A forensic speaker recognition database and some first experiments," in *The Speaker and Language Recognition Workshop (Odyssey 2014)*. ISCA, pp. 6–13. [Online]. Available: https://www.isca-speech.org/archive/odyssey_2014/vandervloed14_odyssey.html

[40] N. Oostdijk, “Het corpus gesproken nederlands.”

[41] W. Ropke, “Training a speech-to-text model for dutch on the corpus gesproken nederlands ,.”

[42] N. Oostdijk, “The spoken dutch corpus. overview and first evaluation.”

[43] W. Heeringa, “Measuring dialect pronunciation differences using levenshtein distance.”

[44] R. Haldar and D. Mukhopadhyay, “Levenshtein distance technique in dictionary lookup methods: An improved approach.” [Online]. Available: http://arxiv.org/abs/1101.1232

[45] H. Bredin, “pyannote.metrics: A toolkit for reproducible evaluation, diagnostic, and error analysis of speaker diarization systems,” in *Interspeech 2017*. ISCA, pp. 3587–3591. [Online]. Available: https://www.isca-speech.org/archive/interspeech_2017/bredin17_interspeech.html

A RESULTS EXPERIMENTS CGN COMP C

Table A.0.1: Performance metrics CGN Comp C, tuning the threshold values, condition_on_previous_text for Faster-Whisper and WhisperX set to True

		CGN Comp C				
Model	**Threshold**	*WER*	*Precision*	*Recall*	*F1*	*RTF*
Faster-Whisper	0.20	0.322	0.839	0.695	0.760	0.082
	0.25	0.320	0.838	0.697	0.760	0.083
	0.30	0.321	0.838	0.696	0.760	0.083
	0.35	0.321	0.838	0.696	0.760	0.083
	0.40	0.320	0.839	0.697	0.761	0.083
	0.45	0.323	0.838	0.694	0.759	0.083
	0.50	0.324	0.838	0.693	0.758	0.084
	0.55	0.324	0.838	0.694	0.758	0.085
	0.60	0.324	0.838	0.694	0.758	0.084
	0.65	0.325	0.838	0.692	0.757	0.085
	0.70	0.325	0.840	0.692	0.758	0.082
	0.75	0.326	0.839	0.691	0.757	0.083
	0.80	0.329	0.838	0.688	0.755	0.085
WhisperX	0.20	0.329	0.848	0.682	0.755	0.047
	0.25	0.328	0.848	0.682	0.755	0.047
	0.30	0.329	0.848	0.682	0.755	0.047
	0.35	0.328	0.848	0.683	0.756	0.048
	0.40	0.327	0.848	0.684	0.757	0.048
	0.45	0.327	0.848	0.684	0.757	0.048
	0.50	0.327	0.848	0.684	0.757	0.047
	0.55	0.326	0.848	0.685	0.757	0.047
	0.60	0.326	0.848	0.684	0.757	0.047
	0.65	0.326	0.849	0.685	0.757	0.047
	0.70	0.327	0.849	0.684	0.757	0.047
	0.75	0.327	0.848	0.684	0.757	0.047
	0.80	0.327	0.848	0.684	0.756	0.047

B RESULTS EXPERIMENTS CGN COMP G

Table B.0.1: Performance metrics CGN Comp G, tuning the threshold values, condition_on_previous_text for Faster-Whisper and WhisperX set to True

		CGN Comp G				
Model	**Threshold**	*WER*	*Precision*	*Recall*	*F1*	*RTF*
Faster-Whisper	0.20	0.172	0.922	0.837	0.877	0.077
	0.25	0.172	0.922	0.836	0.877	0.087
	0.30	0.174	0.921	0.835	0.876	0.088
	0.35	0.172	0.922	0.836	0.876	0.088
	0.40	0.173	0.922	0.836	0.876	0.088
	0.45	0.172	0.922	0.836	0.877	0.088
	0.50	0.173	0.922	0.836	0.876	0.087
	0.55	0.172	0.922	0.836	0.877	0.087
	0.60	0.173	0.921	0.835	0.876	0.087
	0.65	0.173	0.921	0.835	0.876	0.087
	0.70	0.173	0.921	0.836	0.876	0.088
	0.75	0.173	0.921	0.835	0.876	0.086
	0.80	0.174	0.922	0.835	0.876	0.087
WhisperX	0.20	0.171	0.922	0.836	0.877	0.047
	0.25	0.171	0.923	0.836	0.877	0.047
	0.30	0.172	0.922	0.836	0.876	0.046
	0.35	0.172	0.922	0.836	0.876	0.046
	0.40	0.172	0.922	0.835	0.876	0.046
	0.45	0.172	0.922	0.835	0.876	0.046
	0.50	0.172	0.923	0.836	0.877	0.046
	0.55	0.172	0.922	0.835	0.876	0.046
	0.60	0.172	0.923	0.836	0.876	0.046
	0.65	0.172	0.923	0.835	0.876	0.046
	0.70	0.173	0.923	0.835	0.876	0.046
	0.75	0.173	0.922	0.834	0.876	0.047
	0.80	0.174	0.923	0.834	0.875	0.046

C RESULTS EXPERIMENTS NFI-FRITS

Table C.0.1: Performance metrics NFI-FRITS, tuning the threshold values, condition_on_previous_text for Faster-Whisper and WhisperX set to True

		NFI-FRITS				
Model	**Threshold**	*WER*	*Precision*	*Recall*	*F1*	*RTF*
Faster-Whisper	0.20	0.504	0.654	0.606	0.624	0.092
	0.25	0.497	0.658	0.616	0.632	0.086
	0.30	0.502	0.656	0.610	0.628	0.085
	0.35	0.496	0.658	0.611	0.629	0.087
	0.40	0.487	0.666	0.616	0.635	0.086
	0.45	0.485	0.669	0.624	0.642	0.085
	0.50	0.483	0.666	0.620	0.639	0.083
	0.55	0.488	0.668	0.618	0.637	0.080
	0.60	0.485	0.671	0.620	0.640	0.077
	0.65	0.483	0.672	0.618	0.639	0.076
	0.70	0.475	0.681	0.623	0.646	0.075
	0.75	0.469	0.678	0.625	0.647	0.075
	0.80	0.476	0.678	0.615	0.640	0.076
WhisperX	0.20	0.447	0.695	0.642	0.665	0.041
	0.25	0.445	0.696	0.644	0.667	0.040
	0.30	0.443	0.698	0.641	0.666	0.040
	0.35	0.443	0.698	0.644	0.668	0.041
	0.40	0.443	0.696	0.645	0.668	0.040
	0.45	0.443	0.697	0.643	0.667	0.040
	0.50	0.443	0.697	0.646	0.668	0.040
	0.55	0.443	0.698	0.646	0.669	0.040
	0.60	0.443	0.697	0.646	0.669	0.040
	0.65	0.444	0.698	0.646	0.668	0.040
	0.70	0.443	0.700	0.645	0.669	0.040
	0.75	0.443	0.700	0.642	0.668	0.040
	0.80	0.443	0.700	0.643	0.668	0.040

D RESULTS EXPERIMENTS VAD MODELS CGN COMP C

Table D.0.1: Performance metrics CGN Comp C, tuning the threshold values in Silero VAD and pyannote VAD. *DSR: Detected Speech Ratio, calculated by the predicted segments containing speech, divided by the total speech in audio file

	CGN Comp C				
Model	**Threshold**	*Precision*	*Recall*	*F1*	*DSR**
Silero VAD	0.20	0.890	0.999	0.940	1.128
	0.25	0.891	0.998	0.940	1.127
	0.30	0.891	0.998	0.940	1.126
	0.35	0.891	0.997	0.940	1.125
	0.40	0.891	0.997	0.940	1.124
	0.45	0.892	0.996	0.940	1.123
	0.50	0.892	0.995	0.940	1.121
	0.55	0.892	0.994	0.939	1.120
	0.60	0.893	0.993	0.939	1.118
	0.65	0.893	0.992	0.938	1.116
	0.70	0.894	0.990	0.938	1.113
	0.75	0.894	0.987	0.937	1.109
	0.80	0.895	0.983	0.935	1.104
Pyannote VAD	0.20	0.955	0.982	0.967	1.030
	0.25	0.962	0.977	0.969	1.016
	0.30	0.968	0.972	0.969	1.005
	0.35	0.972	0.967	0.969	0.995
	0.40	0.976	0.962	0.968	0.986
	0.45	0.979	0.957	0.967	0.978
	0.50	0.981	0.952	0.965	0.971
	0.55	0.983	0.946	0.964	0.963
	0.60	0.985	0.941	0.961	0.956
	0.65	0.986	0.935	0.959	0.948
	0.70	0.988	0.928	0.956	0.939
	0.75	0.989	0.920	0.952	0.930
	0.80	0.991	0.910	0.948	0.919

E RESULTS EXPERIMENTS VAD MODELS CGN COMP G

Table E.0.1: Performance metrics CGN Comp G, tuning the threshold values in Silero VAD and pyannote VAD. *DSR: Detected Speech Ratio, calculated by the predicted segments containing speech, divided by the total speech in audio file

	CGN Comp G				
Model	**Threshold**	*Precision*	*Recall*	*F1*	*DSR**
Silero VAD	0.20	0.919	0.998	0.956	1.089
	0.25	0.921	0.997	0.957	1.086
	0.30	0.923	0.996	0.957	1.084
	0.35	0.924	0.996	0.958	1.082
	0.40	0.924	0.996	0.958	1.081
	0.45	0.925	0.996	0.958	1.080
	0.50	0.925	0.996	0.958	1.080
	0.55	0.926	0.995	0.958	1.079
	0.60	0.926	0.995	0.958	1.078
	0.65	0.927	0.995	0.959	1.077
	0.70	0.927	0.994	0.959	1.076
	0.75	0.928	0.994	0.959	1.074
	0.80	0.928	0.993	0.958	1.073
Pyannote VAD	0.20	0.958	0.972	0.963	1.018
	0.25	0.961	0.966	0.962	1.009
	0.30	0.963	0.961	0.961	1.001
	0.35	0.965	0.957	0.960	0.994
	0.40	0.967	0.953	0.958	0.988
	0.45	0.969	0.949	0.957	0.982
	0.50	0.971	0.944	0.956	0.976
	0.55	0.972	0.940	0.954	0.970
	0.60	0.973	0.936	0.952	0.964
	0.65	0.975	0.931	0.950	0.957
	0.70	0.976	0.926	0.948	0.950
	0.75	0.978	0.920	0.946	0.942
	0.80	0.980	0.912	0.942	0.933

F RESULTS EXPERIMENTS VAD MODELS NFI-FRITS

Table F.0.1: Performance metrics NFI-FRITS, tuning the threshold values in Silero VAD and pyannote VAD. *DSR: Detected Speech Ratio, calculated by the predicted segments containing speech, divided by the total speech in audio file

	NFI-FRITS				
Model	**Threshold**	*Precision*	*Recall*	*F1*	*DSR**
Silero VAD	0.20	0.718	0.996	0.829	1.449
	0.25	0.722	0.995	0.831	1.436
	0.30	0.726	0.994	0.833	1.427
	0.35	0.728	0.993	0.835	1.419
	0.40	0.731	0.992	0.836	1.412
	0.45	0.734	0.991	0.837	1.404
	0.50	0.737	0.990	0.839	1.395
	0.55	0.740	0.987	0.841	1.384
	0.60	0.743	0.985	0.842	1.373
	0.65	0.747	0.982	0.843	1.361
	0.70	0.751	0.978	0.845	1.347
	0.75	0.756	0.973	0.846	1.330
	0.80	0.761	0.964	0.845	1.307
Pyannote VAD	0.20	0.848	0.975	0.903	1.183
	0.25	0.867	0.968	0.911	1.145
	0.30	0.883	0.961	0.917	1.115
	0.35	0.896	0.954	0.921	1.089
	0.40	0.905	0.947	0.923	1.069
	0.45	0.913	0.940	0.924	1.051
	0.50	0.920	0.933	0.924	1.035
	0.55	0.926	0.926	0.923	1.020
	0.60	0.931	0.918	0.922	1.005
	0.65	0.937	0.910	0.921	0.990
	0.70	0.942	0.901	0.919	0.975
	0.75	0.947	0.890	0.915	0.958
	0.80	0.952	0.878	0.911	0.939