Speech to Text Using Whisper

A study of the automatic speech recognition model on Danish radio broadcasts

Astrid Hansen

EC Utbildning

Data Science Project

202410

Abstract

The report focuses on the Whisper ASR model, which has shown promising results compared to other ASR models. The study aims to experiment with Whisper models to gain knowledge that can enhance the model's transcription performance on Danish radio broadcasts. Among the seven baseline models tested, the small and the large-v3 models yielded the best results in terms of word error rate (wer) and transcription time on a small data sample

The large-v3 model was fine-tuned on a small dataset created from another radio broadcast to test the fine-tuning process and its effect. The fine-tuned large-v3 model was then evaluated on a 2-hour audio file against the OpenAI implementation, the Hugging Face implementation of the large-v3 model, and a pretrained small model obtained from the Hugging Face hub. The Hugging Face implementation of the turbo model gave the best results in terms of a normalized wer-score on 20% when transcribing a long audio file. The fine-tuned model gave a norm. wer-score on 40% which indicates that further experiments of fine tuning the model are needed.

Table of Contents

1 Introduction

It may be an uneven match between the rapidly advancing field of AI and the tech industry on one hand, and a cultural institution like a library, which can come across as somewhat outdated and conservative on the other. However, the fact is that libraries contain an enormous amount of data in various formats, such as text, audio, video, etc., spanning several decades. This knowledge base is something any AI-model - if it had consciousness or desire - would be eager to digest.

On the other hand, libraries continue to generate more and more data – scraping Danish website, saving all tv and radio broadcasts as well as preserving all music and books that are produced. Managing all this new data, along with digitizing all the old data, requires an immense amount of work, but it can be significantly enhanced through the use of machine learning and AI models. The tasks where AI can contribute are virtually endless when it comes to handling this data. These tasks range from data processing, digitalization, generating metadata, clustering material, to improved and personalized search capabilities.

The specific aim of the Data Science team at the Royal Library (KB) in Denmark is a project focused on digitalizing radio broadcasts, from the oldest ones dating back to the 1930s to the most recent ones from 2005. The goal is to reconstruct what was aired on each particular day, apply metadata, and ultimately cluster the material to make it searchable for the library's users. In KB they have conducted experiments with the Whisper models because they saw a potential in using AI models is several situations like producing subtitles for hearing disabled, topic extraction, meta data generation as well as automatically quality checks of the model itself and its output.

1.1 Whisper

Whisper was released September 2022 and analysis of the different ASR models showed a better performance than other ASR models at the time.[1] The performance of the Whisper models in comparison to other ASR-models makes it an obvious choice for further investigation.

Since its release, Whisper has evolved both in terms of more baseline models as well as the different implementations of the models providing different outputs. A key focus of these optimizations of the Whisper models' performance, is not just increasing the accuracy but also increasing the speed. Speed is indeed an important parameter for processing even small amounts of audio, since even 10 seconds of an audio file at a sampling rate of 16 kHz will be fed into a model as an array containing 10x16.000 = 160.000 input features.

1.2 Questions

The aim of this study is to examine the Whisper model and pave the way for further use of the model for transcriptions of audio files. The following questions will seek to be answered:

1. How do the baseline Whisper models perform in terms of wer and transcription time on a small audio sample from a Danish radio broadcast, and what insights can be drawn from the segments that the openai implementation provides?

[1] https://deepgram.com/learn/benchmarking-top-open-source-speech-models

2. Can we increase the accuracy of the transcriptions by using different implementations, fine tuning a model on a small, representative dataset or by using a pretrained model from Hugging Face Hub?

1.3 Structure

The first part of the study provides an overview of the theory needed to answer the questions posed above. To work with audio data requires an understanding of what digitalized sound is and how audio can be fed into ML models. The first section describes the characteristics of audio data, while the second part of the theory section focuses on the Whisper model, its architecture, implementations and characteristics.

The next section revolves around the methodology of the study, detailing what and how the research questions will be answered. There will be an exploratory data analysis of the audio files with visualizations as well as an audio classification. A small section with processing of the manual transcription will also be conducted to be able to use these as reference text when evaluating the models' performances.

Following this comes a section with examination of the baseline Whisper models performance on a small audio sample. The winning model will afterwards be investigated further by transcribing a longer audio file done with the openai as well as with the Hugging Face implementation. Afterwards the winning model will be trained on a small dataset created for the occasion. Another fine-tuned model from Hugging Face, that is trained on the common voice dataset, will as well be tested and the models' performance will be evaluated.

The findings will be presented and discussed in the last section, followed by conclusions and reflections on further possibilities for study.

2 Theory

The first section discusses the characteristics of audio files and what to consider when working with them. The second section focuses on the Whisper model, its architecture and the implementations.

2.1 Working with Audio

Digitalizing sounds means converting sound waves, which by nature are continuous signals with infinite values over time, into a finite number of signals. The process of converting continuous sound waves into discrete electrical values is known as digital representation, and the process is called sampling. This means that the sound waves are converted and then compressed into formats like WAV or MP3, etc.[2]

The number of discrete values is determined by the sampling rate or sampling frequency. This is measured in Hz and represents the number of samples taken in one second. Music typically has a sampling rate of around 32 kHz, while speech is around 8-16 kHz.[3]

One thing to be aware of when working with audio files is the large amount of data that is stored in just one second of audio. With a sampling rate of 8 kHz, there are 60 seconds x 8000 = 480,000 electrical signals to be processed. This means that 10 minutes of radio broadcast contains 4,800,000 signals to be processed. The sheer amount of data stored in a single second of audio makes working with audio files a complex matter. Additionally, there are challenges such as background noise, poor data quality, accents in speech, mumbling, silence, people speaking at the same time or over music playing in the background.

2.1.1 Sampling rate

Since the sampling rate controls the amount of input to the model, it is important to ensure that the data has the correct sampling rate and to resample it if necessary. Pre-trained models have been trained on specific input data, which the new audio data must match. Both downsampling and upsampling of the audio files can be necessary preprocessing steps. Where upsamling is a matter of calculating missing values that lie in between the existing ones, while downsampling requires more advanced calculations that are provided by libraries such as librosa.

2.1.2 Amplitude, Frequency and Bit Depth

Amplitude is measured in decibels and represents the air pressure in a sound wave. The higher the amplitude, the louder the sound. In non-digital audio then 0 decibel represents silence, but for digital audio signals, 0 dB represents the loudest possible amplitude, while all other values are negative.

The same way that the sampling rate tells us about the digital audio files contents per second, then the amplitude in a digital audio file is also represented as discrete values at specific points in time. The precision of the amplitude in digital audio is determined by the bit depth of the

[2] https://huggingface.co/learn/audio-course/chapter1/audio_data

[3] https://huggingface.co/learn/audio-course/chapter1/audio_data

recording. A higher bit depth provides greater precision and less noise, and with this also follows better performance of the ML models. While 16-bit and 24-bit audio are expressed as integers, 32-bit audio is represented in floating-point format. Preprocessing steps may therefore be necessary, as machine learning models work with floating-point values.[4]

Frequency is measured in Hz, and it represents the sound vibrations in the air per second. The human ear can perceive sounds from frequencies down to 20 Hs and up to 20 kHz, where the lower tones correspond to lower frequencies.

2.1.3 Visualizations

Visualizing audio data can be helpful since audio signals are difficult to study as acoustic signals. A typical visualization is the waveform that portrays the sample values over time. The amplitude (sound pressure) is shown on the y-axis with values as floating points between –1 and 1, while the x-axis represents time. This can give indications of the variance of the audio file, indicating where there are silent periods and more intense periods, maybe indicating a presense of music instead of speech.

Another visualization is the frequency spectrum, known as the frequency-domain representation. Instead of amplitude it is visualizing the strength of the frequencies over time. In the plot, frequency is shown on the x-axis, and amplitude in decibels is on the y-axis, where peaks indicate the amplitude of a given harmonic.

To see both amplitude and frequency and amplitude over time, we can use a spectrogram. This is calculated by taking the DFT (discrete Fourier Transform) and stacking the results together in a spectrogram. This results in a plot that shows how amplitude and frequency change over time. The x-axis represents time, the y-axis represents frequency, and color represents the amplitude of a given frequency at a specific point in time, measured in decibels.

2.1.3.1 Log-Mel Spectrogram

A common version of the spectrogram used in ML models is the log-mel spectrogram. The difference between the spectrogram and the log-mel spectrogram is that the frequencies on the y-axis are converted to a logarithmic scale to provide a better representation of the actual human perception of sound frequencies. This is since the human ear is more sensitive to changes in lower frequencies than in higher frequencies.[5]

The log-mel spectrograms can be used as images and be processed by CNN Layers. The Whisper model has CNN-layer in the beginning of the encoder block and uses log-mel spectrograms as input features. One thing to remember is that the spectrograms appear as images but can't be treated in the same way as images. Doing augmentation, cropping or resizing will influence the frequencies portrait in the spectrogram and the model will be fed with wrong data.

[4] https://huggingface.co/learn/audio-course/chapter1/audio_data

[5] https://huggingface.co/learn/audio-course/en/chapter1/audio_data

2.1.4 Preprocessing Audio Files

Raw audio data is represented as an array of values. Typically, the data needs to be processed in some manner and converted into input features that the chosen model accepts. The format of the input features depends on the model's architecture and the data it was trained on.

The Whisper models are trained on audio that are sampled at 16 kHz, and it is limited to processing 30 seconds of audio. The default setting for handling longer audio files is done by slicing the audio in 30 second slices and processing these sequentially. There is also the possibility that Whisper handles chunks of 30 second sound individually and then pastes these together for the final output. The default setting gives better results, while the latter process is faster.

2.1.4.1 Whisper Processor

The Hugging face implementation of Whisper comes with both a processor, tokenizer and feature extractor that processes the input features and corresponding labels. The input features are sliced into 30 seconds chunks and converted into log-mel spectrograms making the audio preprocessing steps easy and accessible.

2.2 Whisper

The whisper models are created by the company Openai. The models are open source and is trained for both speech recognition, translations tasks and language recognition. It was first released in September 2022 and the latest update to the model was done in September 2024 when the large-v3-turbo was released.

2.2.1 The Baseline Models

There are several versions of the Whisper model each with its own characteristics. Below are listed the 9 baseline models and characteristics such as number of parameters, speed and language. The large model comes both as large-v2 and large v3, and the turbo model is a version of the large-v3 model.

Size	**Parameters**	**English-only model**	**Multilingual model**	**Required VRAM**	**Relative speed**
tiny	39 M	tiny.en	tiny	~1 GB	~10x
base	74 M	base.en	base	~1 GB	~7x
small	244 M	small.en	small	~2 GB	~4x
medium	769 M	medium.en	medium	~5 GB	~2x
large	1550 M	N/A	large	~10 GB	1x
turbo	809 M	N/A	turbo	~6 GB	~8x

Figure 1: The baseline Whisper models. https://github.com/openai/whisper

2.2.2 Implementations

The baseline Whisper models can also be implemented in different ways that each have their own advantage. Many of the implementations, like for example Whisper JAX, focus on optimizing the speed of the transcription. Openai also offers their own basic implementation of the model.

This study will use the basic implementations from Openai as well as the implementation provided by Hugging Face. The choice of the Hugging Face implementation is both since it gave the lowest WER when tested in KB, but also because it is easy to install and uses transformers that preprocess the input before running the model. Besides this the implementation is well documented and there are possibilities for fine tuning the model as well.

2.2.3 Training Data

The models are trained on 680.000 hours of audio collected from the internet, with the majority being English audio with corresponding English transcriptions. 18% is non-English audio with English transcriptions and 17% is non-English audio with the corresponding language transcriptions.[6] As a natural consequence then the models perform better in English. But with that said then Whisper large-v3 achieved a WER (word error rate) on 12% on a Danish dataset, compared to a WER on 4.1% on an English dataset. Studies show though that there is a direct positive correlation between the amount of data for a particular language and the quality of the transcriptions.[7]

But it is not just the amount of data that is important. The dataset also needs to be diverse, if the model should be able to handle accents, slang, older language, dialects and other characteristics that influence the spoken words.

2.2.4 Model Card

The whisper model is a transformer sequence-to-sequence model. This implies that it takes an audio sequence as input and outputs a sequence of text, The text is either translated into English or as default transcribed in the language spoken in the audio file.

Below we see that the input features are in the form of log-mel spectrograms. These are fed into 2 Conv1D layers with GELU activation, that converts the spectrograms into embeddings (vectors) by the CNN layers before running through the encoding process. After the encoding process then the features are processed by the decoding layer that works on both cross attention and self-attention. This will be explained in more detail later.

[6] https://github.com/openai/whisper/blob/main/model-card.md
[7] https://github.com/openai/whisper/blob/main/model-card.md

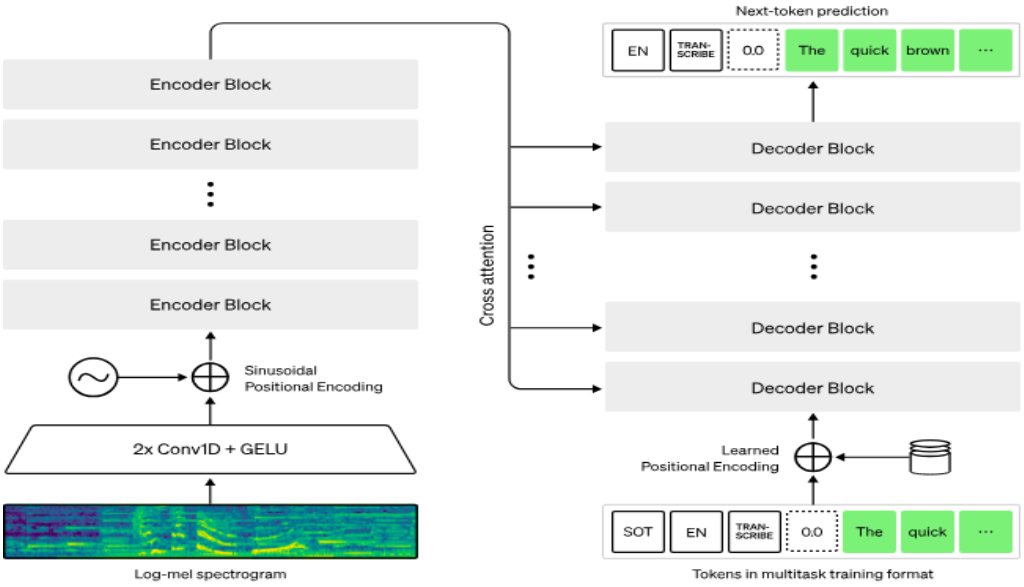

Figure 2: Model Card from https://huggingface.co/learn/audio-course/en/chapter3/seq2seq

2.2.5 Transformer Models

The architecture with first an encoding layer followed by a decoding layer forms the basis of the transformer models. The encoding layer works on the input sequences and uses word embeddings of the tokens both to process them separately as well as in reference to each other. A positional encoding of the tokens is added to ensure that the information about the tokens positions in the meanings is not overlooked.[8]

An essential part of the big language models like ChatGPT is the attention mechanism in the neural network architecture. The self-attention mechanism works by comparing the tokens in the inputs and assigning higher weights to tokens that are semantically related. Based on the calculated weights of the tokens, then the model can distinguish between relevant and irrelevant information.

The decoding block of the model processes input generated from the encoding layer. Like the encoding layer the decoding layer uses word embeddings (word vectors), positional encoding as well as using self-attention before outputting the sequence of tokens for the transcription.

Besides the self-attention mechanism, the decoder also has a cross-attention block. Whereas self-attention calculates each token's weight based on the token itself and its key, or relation to other tokens, in the sequence. Then the cross-attention block works by taking the encoded input tokens and combining these with the output tokens provided by the decoder.[9] The attention mechanism can be repeated multiple times during the encoding and decoding process to refine the results.

Besides the inputs from the encoding layer then the decoder is also fed with special tokens that are obtained via a multitask training format. The decoder then outputs a sequence based on calculating the next token prediction in a sequence.

[8] https://medium.com/@stefanbschneider/understanding-attention-and-transformers-d84b016cd352

[9] https://medium.com/@stefanbschneider/understanding-attention-and-transformers-d84b016cd352

The output layer consists of a softmax activation function that provides probabilities for the tokens and outputs the next token based on which token achieved the highest probability.

2.2.6 Evaluation of Whispers Performance

When measuring the accuracy of the predictions for ASR models, then the predictions are compared to the actual transcriptions. The errors that can occur are substitutions of words, insertions of extra words or removal of words. These errors can be tracted both on word and character level. The most used metric for ASR models is the WER (Word Error Rate). It takes the number of word errors divided by the total number of words. This calculation means that there is no upper limit for how high the error rate can be.[10] The predictions and reference texts can be normalized before calculating the wer which often gives a slightly better wer score since.

2.2.7 Fine Tuning of Whisper Models

Besides varying results for some languages then characteristics such as dialects, accents, gender and age of the person speaking influence the quality of the transcription. To achieve the best possible output then it is possible to train Whisper on representative data. Dealing with radio broadcasts from the 1930's might not give the best transcriptions in a model trained on internet material where the majority likely is produced after the millennium. Language is a dynamic expression where new expressions are created and others are forgotten, as well as the way of speaking can change over just a decade. The process of fine tuning a Whisper model will be explored later in the study.

2.2.8 Other Challenges

Another problem that occurs in transcriptions is hallucinations. This also happens in processing English audio and occurs for example when there is silence or music. It is suggested that it occurs because of the week supervised training manner. The hypothesis is that the model gives more weight to create a coherent structure meaning, because that is what is most prominent in the training data, instead of "listening" to the actual audio itself.

Repetitiveness in the model can also pose issues. Whisper is inclined to be repetitive because it can put too much weight on previously generated tokens and conclude that what has previously been generated, also is more likely to be generated later. It can also be an issue if Whisper is fine-tuned on a dataset that does not contain some diversity, since Whisper can have a tendency to generate tokens that appear more often in the training data.

2.2.9 Tuning Hyperparameters

To avoid repetitiveness then it is possible to employ beam search in the model, which causes more diversity because it allows for broader exploration of possible outputs. Another tool is temperature scaling that controls the probability distribution from which tokens are sampled. Higher temperature thresholds increase randomness and with that the possibility for less likely tokens to be

[10] https://huggingface.co/learn/audio-course/en/chapter3/seq2seq

selected. Lower temperature reduces randomness and leads the model to choose the most likely tokens based on their probability curve.

When faced with uncertain transcriptions then it is possible to test different temperatures during the decoding phase. This is done by initiating the generate_kwargs as an input to the pipeline. It is possible to list both which temperature values that are to be tested as well as setting thresholds on other parameters to refine the transcriptions.

2.2.10 Compression Rate, Avg. Log Prop and No Speech Prop

Other hyperparameters are compression rate and average log probability. The compression rate can prevent the model from getting stuck and repeating the sentences, weighing down sequences with repetitions, on behalf of more unique sequences. The ratio in the Whisper code for the compression rate is set to 2.4. A higher ratio will follow another decoding strategy. If the transcript contains a lot of repetitions, then it can be an idea to test lowering the compression ratio threshold.

The average log probability can be interpreted as a confidence measure of the model's predictions. The threshold for the original Whisper model is set to –1.0. If the result of the decoding strategy does not meet the heuristics of the compression rate or the avg. Log probability, then the model tries to do the decoding based on another strategy.[11]

There is yet another heuristic that is based on identifying no speech. The Whisper authors use a threshold of 0.6 to detect non-speech. This heuristic is ignored if the avg. log probability exceeds its threshold. Interestingly then a small test shows a tendency for larger models to hallucinate when there is silence, whereas the smaller models detect the no-speech sections easier.[12]

[11] https://deepgram.com/learn/exploring-whisper
[12] https://deepgram.com/learn/exploring-whisper

3 Method

First in the section is a presentation of the audio data conducted with some visualizations as well as an audio classification. The data for the study consists of two audio files—one used for fine-tuning, and one used for testing and evaluating the Whisper models. Both audio files have manually written transcripts saved in CSV files. These transcripts will need to be processed to serve as ground truths when calculating the WER metric. A dataset consisting of approximately 2.30 minutes of audio and transcripts will be created to serve as the training and validation dataset for fine-tuning a Whisper model.

In the model testing section then all of the base models will be tested on a small amount of data. Subsequently, the winning model will be examined through a segment analysis that provides insights into Whisper's evaluation of the transcription. The winning model will also be tested using the Hugging Face implementation of the model. The best-performing model will be trained on the created dataset, and its performance will be evaluated on the original audio file using the WER metric. Finally, a fine-tuned model from Hugging Face Hub will as well be tested and evaluated for comparison.

3.1 Exploratory Analysis

The data for the study consists of two audio files that contain about 2 hours of radio broadcasts. One from the channel P1 channel, broadcast on the 3rd of June from 1996, between 6 am and 8 am, and one from the channel P3, broadcast on the 8th of Aug. 2024 between 4 am and 6 am. The audio files are taken from the larger set of audio files in the DR collection.

3.1.1 The Audio Data

The audio file is loaded and inspected using the librosa library. Many of the audio files in the DR catalog have a sample rate of 48,000 Hz. The two audio files that are used in this study has already been downsampled to 16 kHz to serve as input for the Whisper model.

The file is loaded with the sample rate stored as a variable, and the audio data is represented as an array. The numbers in the array are in float32 format, which is necessary for machine learning. The first audio file from 1989 is 2 hours long, resulting in a one-dimensional audio array with 117,134,677 samples. This exemplifies the computational load of handling even smaller segments of audio data.

3.1.2 Visualizations

Inspecting the waveform of the audio file, we can see the changes in amplitude over the duration of the recording. Wider sections of the wave indicate higher sound pressure. This suggests that there is silence at the beginning of the audio file. The wider sections of the waveform could represent music, while the quieter sections indicate lower sound levels.

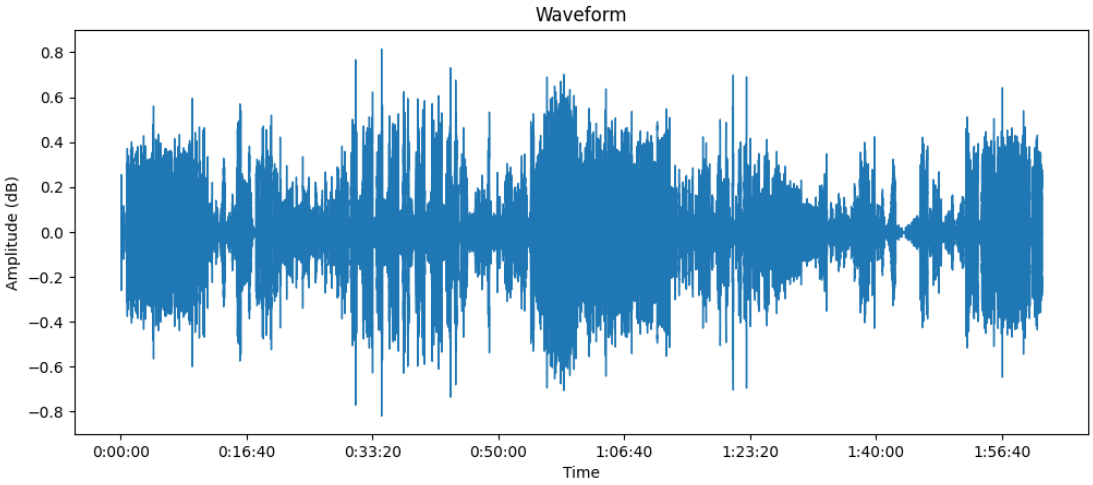

Figure 3: Waveform of the audio file from 1989

3.1.3 Log-Mel Spectrogram

Below is a log-mel spectrogram of the first 2 minutes of the audio file, where we can also see that the sample starts with silence. The first warm-colored dots around 0.15 seconds represent a "dong" sound. This is an introductory sound used before the speaker begins, which occurs just around the one-minute mark. Before the speaker starts, there is a short music jingle, which we also see as a deviation from the speaking pattern in the rest of the spectrogram.

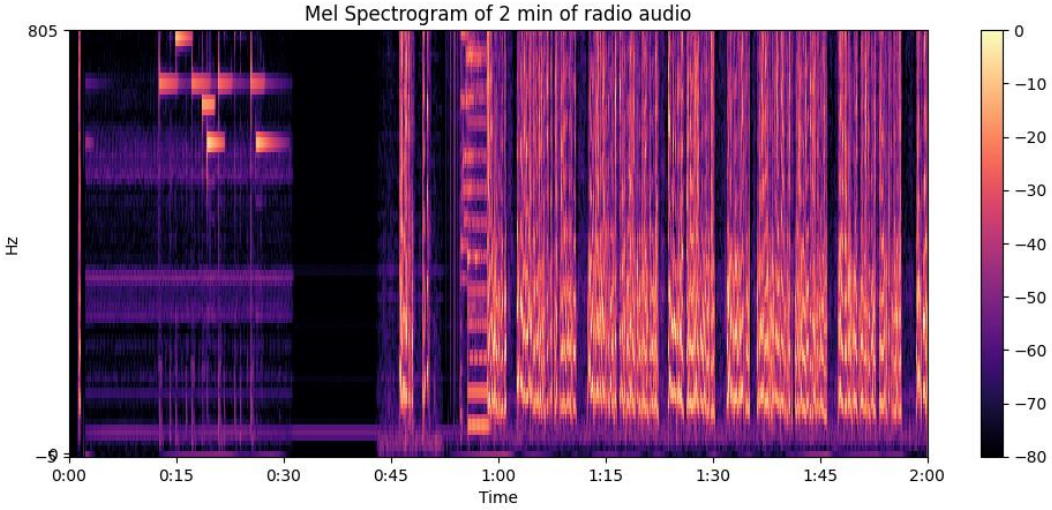

Figure 4: Log-mel spectrogram of the first 2 minutes of the audio file

3.1.4 Audio Classification

Audio Classifications are conducted using a pretrained model from Hugging Face. The model provides predictions about the contents of the two audio files based on determined labels. Using this model, we get a glimpse of the differences in the characteristics of the two audio files. The one from P3 contains more music than the one from P1. This corresponds to the conception of the two channels, where P1 is a speech-oriented channel and P3 indeed plays more music. The two files differ both in content as well as also representing two different decades and are directed to different

segments of the population. Differences that question the representativeness of the dataset that will be created using the audio file from P3.

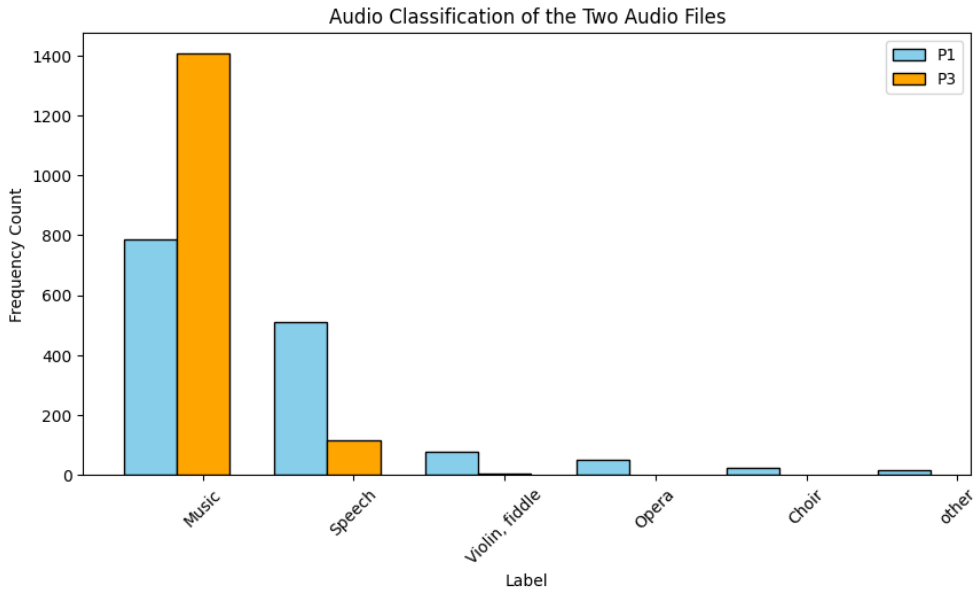

Figure 5: Audio Classification of the two Audio FIles

3.2 Preprocessing the Data

As mentioned earlier then it is not needed to perform a lot of preprocessing for these audio files. The Whisper model takes care of the process of converting the input features to log-mel spectrograms as well as text token processing. This is done by a processor that consists of a feature extractor and a tokenizer. The processor can also be used to downsample the audio to 16.000 hz so it fits the model.

Whisper handles the audio in slices of 30 seconds in two ways. The default way is sequential but for long audio files then it can be set to handle the audio in chunks. The chunks process is faster but not as accurate as the sequential. As this study aims to transcribe two hours of radio then the chunk method will be applied.

Other preprocessing steps like noise reduction, removal of silent periods as well as music sections could be conducted as well and could enhance the accuracy of the Whispers transcription. Since this is an initial study of the Whisper model, then it is of interest how Whisper handles sections with music, noise and silence which is why these sections are not removed.

3.2.1 Text Files

The two CSV files with timestamps and manual transcripts are uploaded and will serve as ground truths when evaluating the models' transcriptions. Besides texts and timestamps then they contain additional comments from the ones who did the transcription. There are notes when there is music as well as other valuable insights. A few cleaning steps are conducted, and the texts are saved as txt files for easy import when doing evaluation. A small sample of the first 5 minutes of text from the P1 file is saved as a small sample to test the baseline models with.

3.2.2 Creating the Dataset

To test the fine-tuning of a Whisper model, we need a dataset. Since this is primarily an exercise and a study of fine-tuning a Whisper model, a small dataset will suffice. Since the audio file from P3 contains a lot of music, we will extract the first 2 and a half minutes which contains speech from the CSV file and use this for fine-tuning. The timestamps will be used to extract the corresponding audio segments from the audio file that match the text in the CSV file. The extracts from the P3 audio and CSV file are afterwards created and saved as a TSV file that can be uploaded as a dataset dictionary using the load_dataset function from the datasets package.

3.3 Testing the Baseline Models

To get an impression of the base line models transcription of the Danish radio broadcast then all the baseline models will be tested on a 5 min audio slice of the original audio file. They will be evaluated on transcriptions time as well as the word error rate. The basic implementation from Openai is used for this examination and the result is shown in the figure below.

The transcription time increases as the models become larger and contain more parameters; however, the accuracy does not improve correspondingly. The two best-performing models on the small sample are the small model and the large-v3-turbo. If we consider transcription time and are willing to accept slightly lower accuracy than what the turbo model offers, the small model demonstrates the best results. The small models wer score is 39.5% while the turbo model is 31.6.

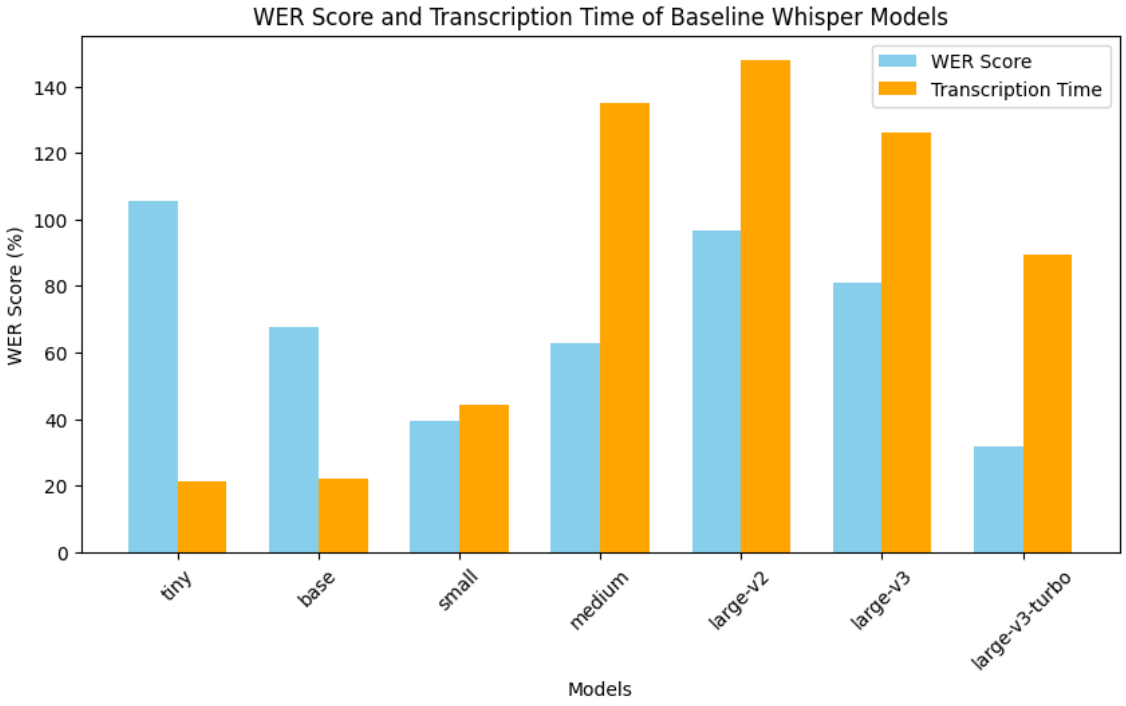

Figure 6: Bar Chart over the performance of the baseline Whisper models

3.3.1 Whisper Segment Analysis

To inspect the performance of the turbo model in further detail, then the openai implementation of the large-v3-turbo model is used for transcribing the 2-hour audio file from P1 and the segments from the transcription process are saved to a csv file for further analysis. The

segments that are provided with the openai implementing of the Whisper models are metrics such as average log probability, no-speech probability, and compression rate. Below is the head of the provided DataFrame, which includes some of the segments generated by Whisper, along with a confidence column calculated from the average log probability for easier interpretation of the numbers. Values close to 1 indicate that Whisper is confident in its predictions. Inspecting the average values of the segment analysis shows that Whisper was 76% confident in its predictions in this transcription. A temperature close to 1 indicates that the model is guessing. Since the first minutes of the audio file is silence and dong sounds then it corresponds well with Whispers numbers below for that time section indicating low confidence and increased guessing.

start	end	temperature	avg_logprob	compression_ratio	no_speech_prob	confidence
0.00	1.76	1.0	-2.364.930	0.466667	3,34E-04	0.093956
30.00	48.24	0.0	-0.397382	0.936508	2,55E-05	0.672077
49.44	50.60	0.0	-0.397382	0.936508	2,55E-05	0.672077

Figure 7: Table over the segment values from the Whisper turbo transcription

Inspecting the values in the confusion matrix, we indeed observe a negative correlation between the confidence and temperature values. Additionally, there is a somewhat negative correlation between the compression ratio and the temperature, indicating that lower temperature (more deterministic output) provides a slightly higher compression rate, meaning that the model is better at compressing or summarizing the input.

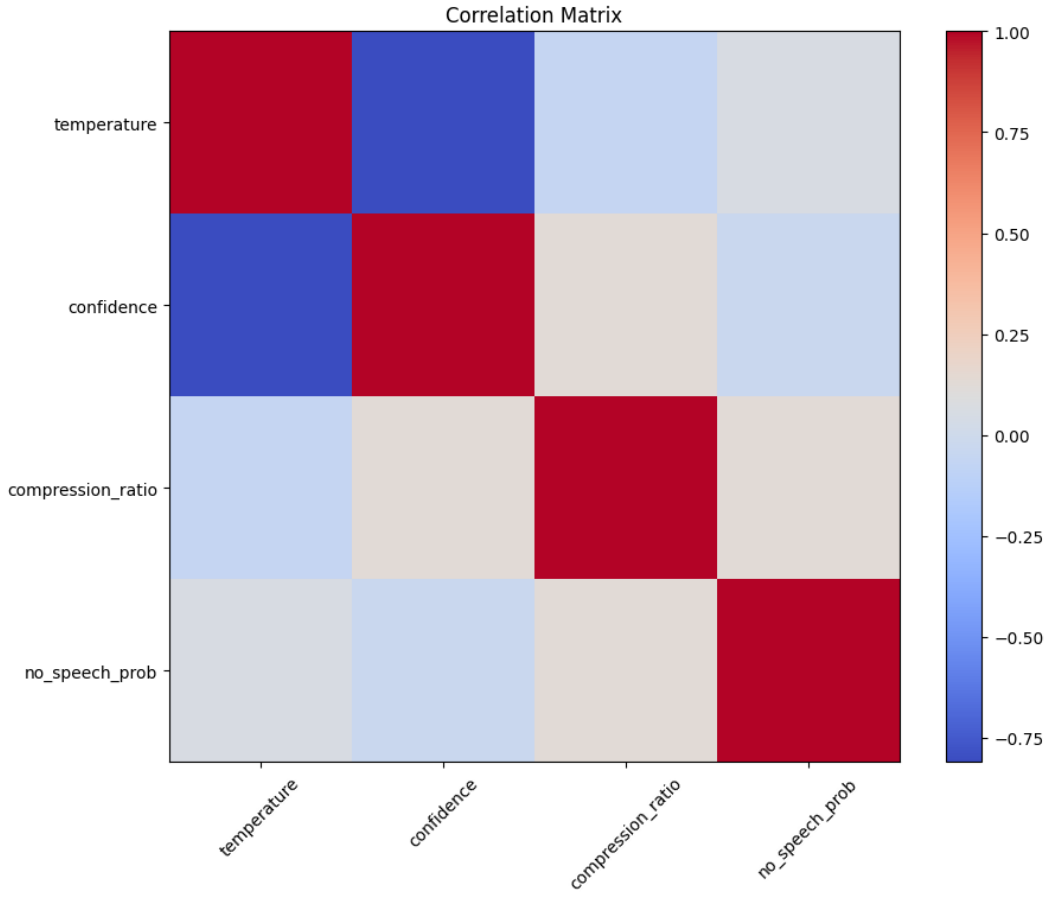

Figure 8: Confusion Matrix over Segments Provided by Whisper Turbo when transcribing the audio file

3.3.2 Hugging Face Implementation

As previously mentioned, the Hugging Face implementation provides an easy and accessible way to use the Whisper model. The turbo model will be implemented to transcribe the 2-hour audio file from P1, and the performance will be evaluated using the WER metric. The implementation involves loading the model (either on CPU or GPU, if available) as well as loading the processor that handles the preprocessing of the audio file. A pipeline is instantiated with the tokenizer and feature extractor, and the audio file is loaded in 30-second chunks in batches of 16.

Below are the evaluation metrics for the transcription of the audio using the turbo model with the Hugging Face implementation. The wer score is not perfect but can be acceptable based on the use of the transcript. It could be sufficient for extracting metadata or doing entity analysis. Compared to the implementation with openai that were conducted for the segment analysis then the Hugging Face implementation shows a good increase in terms of accuracy.

model	wer	wer_norm	substitutions	deletions	insertions
large-v3-turbo	0.226488	0.200717	635	834	224
openai-large-v3-turbo	0.456281	0.432879	1981	765	505

Figure 9. Table over the performance of the openai and the hugging face implementation of the large-v3 model

3.3.3 Fine Tuning the Turbo Model

The next experiment to be conducted involves fine-tuning the large-v3-turbo model using the small dataset we created. Since the dataset is quite small, we do not expect a significant improvement in the WER score. However, this experiment will serve as a test for fine-tuning the model and observing its effects.

The fine-tuning will be performed in Google Colab on a GPU to enhance performance, and most of the code is adapted from the Hugging Face documentation. The fine-tuning process utilizes the Transformers library to load the model, tokenizer, feature extractor, and processor, while the Datasets package is used to import the dataset as a DatasetDict. The dataset is prepared using a small function that generates log-mel spectrograms and processes the text into token IDs, with the audio array sampled at 16 kHz.

A data collator is instantiated to perform further processing on the dataset, returning the log-mel spectrograms as PyTorch tensors and padding the tokens so that they all have the same length. A function that calculates the WER metric will be used, as it will be one of the key metrics for evaluating the best model after training. Training arguments are specified, along with the trainer itself.[13]

3.4 LoRA

Since training language models can take a long time, even with GPU access, we incorporate an adapter from the PEFT package called LoRA (Low-Rank Adaptation of Large Language Models). This method involves freezing the layers in the base model and applying two matrices on top of it. Only these two layers are trained, rather than adjusting all the weights in the entire model. Implementing

[13] https://huggingface.co/blog/fine-tune-whisper

these layers can significantly reduce training time, but it may come at the expense of prediction accuracy.[14]

3.5 Training Results

Even with the training on GPU in colab, on a small dataset and with the use of LoRa, then the training took a little under two hours. Below are the statistics for the 5 epoch of training. Epoch 4 has the lowest wer score and low training loss as well as the validation loss is acceptable in comparison to the other epochs. The checkpoints for epoch 4 is saved as the best model.

Epoch	Training Loss	Validation Loss	Wer
1	1.478.400	1.563.958	37.234.043
2	0.529500	1.110.326	34.042.553
3	0.155100	1.099.420	42.553.191
4	0.078200	1.116.672	32.978.723
5	0.009800	1.110.293	40.425.532

Figure 10: Table over the training results per epoch

3.5.1 Evaluation of the Fine-Tuned Model

The fine-tuned model is uploaded using the Hugging Face implementation and transcribed via the pipeline. Inspecting the evaluation metrics shows an increase in the WER metric instead of a decrease, and there are significantly more deletions in the transcript as well. The training was not successful, which could be attributed to several factors, with one central question being whether the dataset used for training was too small or not representative for the assingment.

model	wer	wer_norm	substitutions	deletions	insertions
Fine-tuned large-v3-turbo	0.427976	0.403128	859	2993	257

Figure 11: Table over the performance of the fine tuned model

3.5.2 Pretrained Model from Hugging Face

The last model that will be tested is a pretrained model from Hugging Face. Since the small model did show good results on the small sample with the baseline implementation, then we will test loading a fine-tuned small whisper model from the hub.[15] It has been finetuned on the Danish version of the common voice 13 dataset from Mozilla and achieved a normalized wer score of 23.4%. The dataset consists of 12 hours of audio and corresponding transcripts with an overweight of male voices and people in the ages 20 to 50.

[14] https://www.youtube.com/watch?app=desktop&v=G51AHmGGrys
[15] https://huggingface.co/WasuratS/whisper-small-da

3.5.3 Evaluation of the Models

The last section in the jupyter notebook contains a function that calculates different metrics as well as a function that iterates over the predictions and models, before outputting a dataframe with the result. The results will be presented in the following section.

4 Results and Discussion

This section will present the findings of the study as well as discuss and reflect on the results.

4.1 The Baseline Models

Running the baseline models on a small audio sample showed that the Whisper small model also provided good performance in terms of both transcription time and WER score. The transcription time increased with the model sizes without improving the WER score. The turbo version of the large-v3 model demonstrated a significant decrease in transcription time compared to the other large models, as well as provided the best WER score. Therefore, the turbo model was selected for further examination.

Inspecting the segments provided by the transcription with the openai implementation gave insights into the different metrics that Whisper outputs when generating the predicted tokens. Further inspection of the transcript per timestamp in relation to the manual transcription is recommended to further understand where Whisper makes wrong predictions, how confident it is and if it indeed recognizes no speech when there is silence.

It is possible to adjust the threshold for the metrics as well as testing different temperature when running the Whisper models. These thresholds can in other words be adjusted so it fits the material that it is meant to process. A deeper analysis of these figures and experiments with different thresholds for the heuristics could help improve accuracy in the transcripts.

4.1.1 Evaluation of the Models

Four full transcriptions were performed using the large-v3-turbo model: one with the Hugging Face implementation, one with the basic OpenAI implementation, one after fine-tuning the turbo model and one using a pretrained model from the Hugging Face hub. The Hugging Face implementation yielded the best results, showing a significantly better WER score than the other models. The deletions of words in the predictions of the fine-tuned model from the Hugging Face Hub are noteworthy and should be investigated. Inspecting the transcription of this model, there is a high degree of repetitions.

model	wer	wer_norm	substitutions	deletions	insertions
openai-large-v3-turbo	0.456281	0.432879	1981	765	505
hf-large-v3-turbo	0.226488	0.200717	635	834	224
fine-tuned-large-v3-turbo	0.427976	0.403128	859	2993	257
WasuratS_whisper_small_da	0.851353	0.813139	3772	4365	769

Figure 11: Table over the models performance

4.2 Discussion

Running the Whisper model on the Danish radio broadcasts yielded a maximum accuracy of 20% for normalized texts. While this may be sufficient for some purposes, such as clustering or extracting metadata, there is significant room for improvement.

Since Whisper is primarily trained on English material and Danish is a smaller language, the Danish material that Whisper is trained on constitutes only a small fraction of the overall 17% of the

multilingual dataset that Whisper was trained on. This limitation will indeed affect the WER metric, as studies show that the more material available for a language, the better the WER score from the Whisper models. The experiment with fine-tuning the model on a small dataset did not yield impressive results. Several factors contribute to this, where size and representativeness are two of them. The pretrained small model from the Hugging Face hub, which was trained on more Danish material, demonstrated worse accuracy than the baseline small model. Future experiments with fine-tuning a model are recommended, both with a larger dataset as well as using a more representative audio sample from the same decade and channel as the test data.

The PEFT adapter LoRA was employed to decrease training time, as training language models typically requires a significant amount of time. Training on just 2.5 minutes of audio took around 2 hours, even with LoRA. While using the LoRA adapter did enhance the speed of training, experiments indicate that increased speed can lead to a decrease in transcription accuracy. [16] Speed is a critical factor, as processing even small amounts of audio data requires significant time and energy. The importance of various parameters depends on the intended use of the transcripts, as different applications demand varying levels of accuracy. For instance, subtracting metadata may allow for lower accuracy, while creating subtitles for video requires high accuracy. The balance between training time and accuracy is a consideration that must be made based on the specific assignment at hand.

The audio files that were used did not undergo further processing. The accuracy could maybe been improved by detecting and removing the silent periods as well as the ones containing music from the audio files. Initiatives like these should be pursued in future experiments.

5 Conclusions

This section will recap on the questions posed in the beginning and give a resume over the results as well as reflect upon the use of Whisper and processing audio material.

5.1 Questions

The questions that we posed in the beginning of the study were as follow:

1. How do the baseline Whisper models perform in terms of wer and transcription time on a small audio sample from a Danish radio broadcast, and what insights can be drawn from the segments that the openai implementation provides?

2. Can we increase the accuracy of the transcriptions by using different implementations, fine tuning a model on a small, representative dataset or by using a pretrained model from Hugging Face Hub?

5.2 Results

The Whisper small and large-v3 models showed the best performance compared to the other baseline models. The small model transcribed faster, but the large-v3 model gave the best Word Error Rate (WER) score. The large-v3 model achieved a confidence level of 76% when inspecting the segment analysis. Further analysis of the segments per timestamp is needed if hyperparameters are to be adjusted for a given task.

Out of the four models tested on a large audio file then the large-v3 model with the Hugging Face implementation gave the best results in terms of WER score. The fine-tuned model that was trained on similar radio broadcast did not yield better results than the baseline model. The lack of improvement in the WER score could be due to the training dataset being both small and lacking diversity, or that it simply was not a good representative sample for the audio files to be transcribed. The two audio files differ by 15 years and were broadcasted on two different channels targeting different segments of the population. It did perform better than the small pretrained model from the Hugging Face Hub, indicating that it may not be the size of the data set used for training that was the cause of the decrease in the WER score.

5.2.1 Reflections and Recommendations

The study serves as an initial examination of the performance of Whisper models on Danish radio broadcasts. The report provide insights into the model's capabilities and the challenging task of fine-tuning it for improved accuracy. However, the study also raises several questions and highlights the need for further investigation and experimentation.

At KB, researchers have experimented with using the numerical data provided by the segments to analyze the outputs of Whisper when it hallucinates. They implemented a Decision Tree model to learn from these numbers and predict when Whisper makes accurate predictions. Approaches like these, along with further segment analysis, are recommended. Whisper exhibits a tendency to be both repetitive and hallucinatory, underscoring the necessity for some form of automated quality control for the transcriptions.

5.2.2 Closing remarks

The rise of AI-generated content presents both opportunities and challenges. Many institutions, such as KB, are eager to utilize these tools but face ethical, practical as well as legal constraints. Key issues include potential bias in the models, the data on which they have been trained, legal limitations, and the environmental impact of operating large language models. This along with keeping track of the fast-paced development of the Whisper models and their implementations requires both time and effort, both for updating code pieces and documentation as well as performing experiments.

Reference List

Deepgram, Dec 2022. Article from AI & Engineering, “Benchmarking Top Open Source Speech Recognition Models: Whisper, Facebook wav2vec2, and Kaldi”. Retrieved Oct 2024 https://deepgram.com/learn/benchmarking-top-open-source-speech-models

Deepgram, Oct 2023. Article from AI & Engineering, “Exploring OpenAI Whisper Speech Recognition”. Retrieved oct. 2024. https://deepgram.com/learn/exploring-whisper

Github. Openai/Whipser. “Model Card: Whisper”. Retrieved Oct. 2024 https://github.com/openai/whisper/blob/main/model-card.md

Hugging Face. June 2023. Audio Course, Chapter 1. “audio_data”. Retrieved Oct. 2024 https://huggingface.co/learn/audio-course/chapter1/audio_data

Hugging Face. June 2023. Audio Course, Chapter 3. “seq2seq”. Retrieved Oct. 2024 https://huggingface.co/learn/audio-course/en/chapter3/seq2seq

Hugging Face. 2023. “WasuratS/whisper-small-da". Retrived Oct. 2024. https://huggingface.co/WasuratS/whisper-small-da

Hugging Face, Nov 2022. “Fine-Tune Whisper For Multilingual ASR with 🤗 Transformers”. Retrieved Oct. 2024. https://huggingface.co/blog/fine-tune-whisper

Medium, Nov 2022. “Understanding Transformers and Attention”. Retrieved oct. 2024. https://medium.com/@stefanbschneider/understanding-attention-and-transformers-d84b016cd352

Youtube, Trelis Research, Jan 2024. “Fine Tuning Whisper for Speech Transcription”. Retrieved Oct. 2024. https://www.youtube.com/watch?v=anplUNnkM68

Youtube, AI Tinkerers, Oct 2024. “(AI Tinkerers Ottawa) Fine tuning Whisper with PEFT LORA w/ Rishab Bahal”. Retrieved Oct. 2024. https://www.youtube.com/watch?app=desktop&v=G51AHmGGrys