Speech-to-text models to transcribe emergency calls

Jens A. Thuestad, Øyvind Grutle

Master's thesis in Software Engineering at

Department of Computer science, Electrical engineering and Mathematical sciences,
Western Norway University of Applied Sciences

Department of Informatics,
University of Bergen

May 2023

Western Norway
University of
Applied Sciences

Acknowledgements

We would like to express our deepest gratitude to our supervisor Dr. Alexander Selvikvåg Lundervold, for his guidance and support throughout this thesis. His expertise and insights have made a significant contribution to our learning process, and we sincerely appreciate the knowledge we have gained from him.

Our gratitude extends to Professor Guttorm Brattebø, Dr. Emil Iversen, and the rest of the AI-support in Medical Emergency Calls research group for helping us to delve into such a meaningful project. Being part of this progressive development has been both enlightening and rewarding.

We sincerely thank our fellow master's thesis students at the Mohn Medical Imaging and Visualization Centre (MMIV), who contributed to a vibrant and enjoyable workspace.

Jens

I would like to express my thanks to my family for all their encouragement, love, and support. A special word of gratitude goes to my partner, Juliana. Your patience, understanding, and love have provided me with the stability and motivation I needed along the way. Lastly, I wish to acknowledge the unwavering commitment of my co-author, Øyvind. His camaraderie has made this journey an enjoyable one.

Øyvind

I want to thank my family for their consistent support throughout this project. Their backing was helpful in the completion of this thesis. Lastly, a special mention to Jens, my co-author. His commitment and our collaborative effort made this journey both efficient and enjoyable.

Abstract

This thesis is part of the larger project *"AI-Support in Medical Emergency Calls (AISMEC)"*, which aims to develop a decision support system for *Emergency Medical Communication Center (EMCC)* operators to better identify and respond to acute brain stroke. The system will utilize historical health data and the transcription from the emergency call to assist the EMCC operator in whether or not to dispatch an ambulance and with what priority and urgency.

Our research primarily focuses on adapting the Automatic Speech Recognition (ASR) model, Whisper, to create a robust and accurate ASR model to transcribe Norwegian emergency calls. The model was fine-tuned on simulated emergency calls and recordings done by ourselves. Furthermore, a proof-of-concept ASR web application was developed with the goal of streamlining the manual task of transcribing emergency calls.

After demonstrating the application to the involved researchers in AISMEC, and the potential users, both suggested optimism about the potential of this solution to streamline the transcription process. As part of our research, we conducted an experiment where we utilized the suggested transcriptions provided by the application and then corrected them for accuracy. This approach showed a notable reduction in our transcription time. We also found that establishing a machine learning pipeline to fine-tune the model on historical emergency calls was feasible.

Further work would involve training the model on actual emergency calls. To investigate the efficiency of the ASR web application further, a larger scale of the semi-automatic transcription experiment could be conducted by the professional audio transcribers at *Haukeland universitetssjukehus*.

Glossary

AI	Artificial Intelligence.
AISMEC	AI-Support in Medical Emergency Calls.
API	Application Programming Interface.
ASR	Automatic Speech Recognition.
BPE	Byte Pair Encoding.
EMCC	Emergency Medical Communication Center.
GPT	Generative Pre-trained Transformer.
HEMS	Helicopter Emergency Medical Services.
NLP	Natural Language Processing.
OHCA	Out-of-Hospital Cardiac Arrest.
SRT	SubRip Subtitle.
UI	User Interface.
WAV	Waveform Audio File Format.
WER	Word Error Rate.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The ultimate goal of this project is to improve emergency calls using Artificial Intelligence (AI) and machine learning. Today, emergency calls are handled only by the operator and their ability to make a decision with guidance from the *Norwegian Index for Emergency Medical Assistance* ("Index") [35]. A system that could use artificial intelligence to efficiently produce suggestions that can help the operator during the call could make the process much more efficient and increase the accuracy of choosing the correct resources to spend in different scenarios. Furthermore, since a computer can gather data and make computations quickly, patient records, health data about the caller, and outputs from AI models based on this and similar data could be instantly accessible to the operator. This includes beneficial information such as what medication they might be using and previous health-related complications. For a machine to extract actionable information about an ongoing call, it is also useful to analyze what is said during the call, for example, based on accurate transcriptions from speech-to-text models.

Speech-to-text is something that has been around for some time. One well-known example of speech-to-text is the auto-generated subtitles on YouTube videos. However, this system is limited because it is only available for a limited set of languages [53], which does not include Norwegian. There is a lack of good solutions tailored to the Norwegian language, especially when the audio quality is poor, as it often is in phone calls. As mentioned, a set of historical 113 calls will be used to develop the models. These recordings are per now saved in such a way that both the callers' and the operators' audio is layered in the same file. This, in addition to other implications such as background noise and imprecise speech, as it is not uncommon for the caller to be flustered in the case of an emergency situation, makes it hard to transcribe the audio unless the model is trained to tackle these problems.

Another use for models trained to analyze human speech is to have them detect information about how the caller communicates, such as having slurred or strange speech. If the person experiencing the emergency is the one calling, such irregular speech may be helpful information for the diagnostic process. It could, for example, be a sign of a brain stroke [49]. This is not explored in our

thesis, but similar methods to the ones we employ are natural candidates for such detection tasks.

1.1 Expected results

The objective is to develop a speech-to-text model specifically tailored for transcribing Norwegian emergency calls. Our focus is on enhancing the robustness of the model for handling real emergency phone recordings, rather than outperforming existing general-purpose speech-to-text models. This will ensure greater accuracy and effectiveness in critical situations where clear and precise communication is vital.

1.2 Research questions

With this master thesis, we aim to investigate existing speech-to-text models in order to develop a solution best suited for transcribing Norwegian emergency calls. While our research primarily contributes to the existing *"AI-Support in Medical Emergency Calls (AISMEC)"* research project, we also explore other application areas for the model. It is essential, regardless, to ensure that the finished product can be accessed from within hospitals and emergency call centers. The actual implementation of our system inside the hospital infrastructure is beyond the scope of our thesis. Still, we are designing and documenting our setup to ease any subsequent migration process. To clarify the scope of this thesis, we formulate the following research questions and research objectives:

RQ1	Can we construct a system for accurately transcribing Norwegian emergency calls by fine-tuning OpenAI's Whisper model using historical emergency calls?
RQ2	Which further steps, for example, source separation, can improve the resulting system referred to in RQ1?
RQ3	What technologies and infrastructure can be used to make such a transcription system available inside the hospital and emergency call centers?

1.3 Research methods

We have incorporated a loose interpretation of the prototyping research methodology [55], an iterative development cycle described below.

1. **Determine objective**: Understand the requirements that the system must fulfill.
2. **Build Prototype**: Create a prototype based on the requirements.
3. **User feedback**: Demonstrate the prototype to stakeholders and/or potential users of the system.
4. **Refinement**: Improve the prototype based on user feedback.

5. **Iteration**: Repeating the evaluation and refinement process until the prototype meets the desired requirements.

We have drawn from the prototyping research methodology, but it is important to note that we loosely implement them; it does not follow the conventional prototyping methodology to the letter. Moreover, since our goal is not to implement a complete system as suggested by the final stage of the prototyping methodology but to explore the potential viability of such a system, the last step of implementation falls outside the scope of our thesis.

See Section 5.2.2 about the chosen technology stack. To construct the machine learning solution at the core of our prototype, we have chosen a well-suited set of software tools and libraries, described next.

1.4 Machine learning methods

Python[1] is highly appropriate for machine learning due to its widely developed ecosystem of libraries, which allows us to create state-of-the-art models without necessitating the development of existing solutions from scratch. Hugging Face Transformers[2] is an especially useful tool as this library can help us build good solutions using pretrained models. Hugging Face also makes a wide variety of datasets readily available and ready to use without needing much preprocessing on the data.

We conduct several tests using OpenAI's pre-trained model Whisper[3], which is trained in a supervised manner on 680,000 hours of multilingual audio samples. Since Whisper handles Norwegian speech effectively, we choose to investigate its performance in transcribing audio from phone calls involving multiple speakers. Our initial task is to use Whisper for transcribing a simulated phone call in Norwegian between a professional Emergency Medical Communication Center (EMCC) operator and an acting caller. Next, we use this simulated phone call, along with our transcription of it, to fine-tune the model.

1.5 Evaluation

We use a combination of quantitative and qualitative research methods to evaluate the performance of the ASR model. One obvious way to assess the performance of an ASR system is the Word Error Rate (WER). WER provides a numerical metric that is a clear and standardized way to quantify the accuracy of an ASR system's transcription. While WER is a good way to evaluate the performance of an ASR system, it does not capture the entire picture, especially when it comes to understanding the content and context of the transcription [26]. We, therefore, manually inspect the transcriptions to assess the improvement in understanding important phrases and words after fine-tuning the model. It is important to note that improving these qualitative aspects does not necessarily correspond to a decrease in the WER. If the model better understands impor-

[1] `https://www.python.org/`
[2] `https://huggingface.co/docs/transformers/index`
[3] `https://huggingface.co/openai/whisper-large-v2`

tant phrases and words, it could be considered a better-performing model even if the WER does not improve.

We use a similar approach to evaluate the efficiency of the transcription web application. We measure the time taken for the application to transcribe audio clips and conduct experiments to measure the efficiency of using a semi-automatic transcription method instead of manually transcribing the whole audio. After demonstrating the application to different stakeholders and potential users, we use their feedback to further develop the application.

Part I

Background

Chapter 2

The context: emergency medicine and emergency communication centers

2.1 Prehospital emergency medicine

The Norwegian healthcare system is divided into two levels, primary care, and secondary (specialist/hospital) care. Primary care of patients involves emergency medical assistance provided by General Practitioners and the out-of-hours service, which addresses health issues at the most appropriate and effective level of care [11]. Patients requiring urgent medical assistance are mainly handled by Norway's secondary health services, which include the ambulance service and Emergency Medical Communication Center (EMCC) [29, 62].

The ambulance service provides transportation through different means, such as cars, boats, planes, and helicopters. Cars are operated by a team of two ambulance workers, planes by specialized nurses, and helicopters are managed by anaesthesiologists and rescuers [20]. Rescue and transportation of patients in Norway can be incredibly challenging because of seasonal weather changes, geographical land formations such as fjords and mountains, and poor road conditions [29]. Helicopter Emergency Medical Services (HEMS) are, therefore, specifically beneficial when it comes to geographical access, transportation time, and the ability to administer advanced critical care interventions to patients before they reach the hospital [16, 32, 30]. One crucial factor is that EMCC operators correctly dispatch HEMS to patients who actually need it. The EMCC has the authority to dispatch all of the different emergency medical transport options, except for HEMS. If EMCC notifies HEMS, a decision to accept the mission will be made by an anesthesiologist based on medical indications [38, 11].

All contacts within the secondary health care system are made with the EMCCs, which received 936 419 inquiries in 2015 [34].

2.2 Emergency Medical Communication Centers (EMCCs)

Emergency Medical Communication Centers (EMCCs), known as "AMK-sentraler" in Norway, were developed in the early 1980s with healthcare as the driving force and the police, fire department, and the former Televerket as important partners [34]. The purpose of these centers was to provide a quick and effective response to medical emergencies and to coordinate the response of different emergency services.

Today, there are 16 emergency medical communication centers in Norway [34]. EMCCs operate on a 24/7 basis and handle all types of emergency calls, from minor injuries to life-threatening emergencies. They provide medical advice and assistance over the phone and dispatch the appropriate emergency services to the scene. The operators who work in these centers are trained professionals who use medical protocols to provide instructions to the caller until the ambulance or other emergency services arrive at the scene [20, 11].

The increase in the elderly population and the prevalence of diseases are some of the challenges that EMCCs faces, leading to a greater demand for emergency medical services. Therefore, there is a need to strengthen the EMCCs' assessment, coordination, and prioritization functions by enhancing the operators' healthcare expertise, improving medical involvement, and developing better decision-support tools [34].

2.3 Brain stroke

A brain stroke occurs when the blood flow to the brain is interrupted, which can happen in two ways: a blood artery that provides blood to the brain is blocked by a blood clot, or it ruptures and bleeds into the brain. When the blood supply to the brain is interrupted, brain cells die [50]. Since more brain cells die as time passes, time is of the essence. This is also emphasized by the phrase "Time is brain", coined by Neurologist Camilo Gomez, M.D. Essentially this means that it is crucial for the ambulance to reach the patient as quickly as possible.

Dead brain cells lead to different symptoms that vary depending on the affected part of the brain [58]. Still, the most common symptoms are sudden difficulty speaking or understanding others, sudden weakness or numbness in the face, arm, or leg, especially on one side of the body, sudden vision problems, sudden difficulty walking or loss of balance, and sudden severe headache. To help recognize a stroke, the acronym F.A.S.T is useful [27, 28]:

Face	Their face may droop to one side. Ask the person to smile.
Arms	The person may have weakness or numbness in one arm. Ask the person to raise both hands over the head and see if one arm drifts downwards or cannot be raised.
Speech	The person's speech may be slurred. Ask the person to repeat a simple phrase or sentence and listen for any difficulty or changes in their speech.
Time	If any of the mentioned symptoms are present, it is time to call the emergency number since getting help fast in case of a brain stroke is critical.

There are two types of treatment for brain stroke: either medical treatment or surgical treatment. The most common medical treatment is called thrombolysis, a blood clot-dissolving treatment through medication. A standard surgical procedure is called thrombectomy, which removes a blood clot from a blood vessel by inserting a small, flexible tube (catheter) into the blood vessel. The catheter is guided to the blood clot, where a stent retriever is used to grasp and remove the clot [21, 22].

2.4 Diagnostic workflow for brain stroke

Norwegian Index for Emergency Medical Assistance [35] give an overview of different symptom criteria which gives reason to suspect acute brain stroke and possibly trigger an acute response, also called a *red response*. An essential aspect of the pre-hospital phase is establishing a system for notifying the relevant hospital. Patients with symptoms of a brain stroke should get high priority for immediate service from an ambulance. An air ambulance may be considered if it is necessary to reach the hospital within four hours from the first symptom, or the transportation time can be reduced by more than 30 minutes [37].

To ensure effective and accurate diagnosis, there should be clear procedures when the patient arrives at the hospital. Upon arrival, the time of symptom debut should be clarified. If it is less than four hours since the first symptoms, a decision should be made quickly as to whether the patient is a potential candidate for thrombolysis (and/or) thrombectomy.

Numbers from a study based on Emergency department and EMCC records from a comprehensive stroke centre in Oslo, Norway, during a six-month period (2019-2020) [1] show that 77% of brain stroke patients was identified by the EMCC and given the appropriate response. In those cases, the EMCC used on avarage 01:29 minutes to assess the need for an acute medical response in contrast to 00:55 minutes in the 23% where brain stroke was not identified. The ambulance used on avarage 06:01 and 07:39 minutes, respectively, to arrive at the scene after being alarmed. It was also found that in the cases where brain stroke was not identified, there was a median 11-minute prehospital delay. This is due to approximately half of these cases being given the degree of urgency

[1] Source: personal communication with Bjørn Jamtli, Oslo University Hospital, OsloMet, and Norwegian Directorate of Health

Urgent instead of *Acute* in addition to the ambulance personnel spending more time on site. From this, we can see that in the case of a time-sensitive condition like a brain stroke, accurate diagnosis is critical to ensure prompt arrival at the hospital and improve patient outcomes. Therefore, it is essential to prioritize developing and implementing effective diagnostic tools and protocols for healthcare professionals.

Chapter 3

Methods: deep learning for speech recognition

3.1 Machine learning and deep learning

Machine learning [31] is a subfield of artificial intelligence (AI) focused on developing algorithms that can learn from data, imitating how humans learn. These algorithms are designed to improve their performance on a specific task over time by learning from experience.

The experiences typically come as a set of labeled data, and the machine learning algorithms attempt to learn the relationship between the input data and the desired output labels. Once the algorithm has learned this relationship, it can be used to make predictions on new, unseen data.

This forms the foundation of what is known as *supervised learning*. There are other kinds of machine learning, including unsupervised learning and reinforcement learning. Supervised learning algorithms are trained on labeled examples, where the correct output is provided for each input. Unsupervised learning algorithms are trained on unlabeled data and can find patterns and relationships in the data without any prior knowledge. Reinforcement learning algorithms are trained to make decisions in a dynamic environment in order to maximize a reward.

Machine learning algorithms are used in a wide variety of applications, such as image and speech recognition, recommendation systems, and predictive modeling. These algorithms can often make more accurate predictions or suggest more effective actions than what would be possible using traditional programming techniques.

Deep learning [14] is a specific kind of machine learning that uses artificial neural networks to model and solve complex problems. The term "deep" refers to the number of layers in the network, with deep artificial neural networks having multiple hidden layers between the input and output layers. These hidden layers enable deep learning algorithms to learn and represent more abstract and

complex relationships between inputs and outputs, allowing them to perform complicated tasks such as image and speech recognition with high accuracy.

3.2 Speech recognition

Automatic Speech Recognition (ASR) models are under constant development but have existed for a long time. The history of ASR started in 1952 with a system developed by researchers at Bell Laboratories. The system was able to recognize spoken digits by a single speaker with an accuracy varying between 97 and 99 percent [10]. Even though this was very impressive at the time, it had a very limited vocabulary and required a controlled environment. During the 1960s and 1970s, researchers began to explore more complex statistical models for ASR, such as Hidden Markov Models [42], which worked by modeling the probability distribution of acoustic features and using these models to predict the most likely sequence of words. During the 1980s and 1990s, more robust and scalable ASR systems were developed. These models could recognize larger vocabularies and work in real-world environments. This led to ASR systems being used in applications like telephone-based customer service and dictation software. The introduction of neural networks and deep learning have led to significant improvements in ASR systems, and they are now being used in a variety of applications.

3.3 Deep learning for speech recognition

The robustness and accuracy have increased drastically after deep learning revolutionized the field of speech recognition. The ability to learn complex relationships in data makes it ideal for this task, and after the introduction of Transformer models [54] the development of ASR models has skyrocketed. Transformer models can process inputs in parallel, making them more efficient. It can also capture long-range dependencies using a mechanism called *"attention"*, which enables the model to focus on different parts of the input sequence when producing the output. Additional information about some selected terms in deep learning are included in the glossary 3.1.

Deep learning also faces challenges when applied to ASR. Emergency calls often tend to contain background noise, several speakers, and irregular speech given the often stressful nature of the situation. By training the model on a diverse set of data, including these challenging factors, the models can learn to understand and transcribe these kinds of calls accurately.

Utilizing pre-trained models makes it possible to significantly reduce the need for extensive fine-tuning on task-specific data. The pre-training involves training the model on a large dataset, containing audio from different contexts. This allows the model to learn features, patterns, and nuances in the audio. If this training data also includes audio with background noise, it can learn to isolate speech patterns and ignore irrelevant noise. This is useful when dealing with emergency calls as these tend to contain some level of background noise, which will vary depending on the environment of the call. Using a multilingual pre-trained model such as Whisper [44] drastically reduces the need for

fine-tuning on a specific language. This is because a multilingual model has already learned to understand and process a variety of languages. Not only is it familiar with the syntax, semantics, and structure of the languages, but it also has an understanding of the different accents and dialects that can exist within a language. A multilingual model also has the advantage of transfer learning across languages, meaning that the understanding learned from one language can be used in the processing of another. This benefits low-resource languages, such as Norwegian since they can benefit from the understanding derived from related or more resource-rich languages.

Autoregressive models	Pre-trained models tasked with predicting the next word given the previous outputs. The output at each time step is fed back to the model as input for the next time step [51].
Self-supervised learning	Type of machine learning that involves training on a large amount of unlabeled data, which removes the need for human annotation. In language modeling, the model can be trained to predict the next word based on the context of the previous words.
Transfer learning	Transfer learning is a machine learning technique where a pre–trained model is used as a starting point for a different but similar problem. Transfer learning consists of pre-training a model, typically on a large dataset where the model learns generic features. The second step is fine–tuning, where the pre–trained model is adapted to a different problem, usually with a smaller dataset. The idea behind transfer learning is to leverage the knowledge from the pre-trained model to enhance the performance on the second problem [39].
Tokenization	Tokenization is a process in Natural Language Processing (NLP) that breaks down text into smaller units (or tokens), such as words, sub-words, or characters. In this way, machines can understand the meaning of each individual part both independently and in the context in which it is used.[4]
Language model	Language models are a type of machine learning model that is trained on a large amount of text data, trained to understand, generate and manipulate human language. It can be used for different NLP tasks such as text generation, translation, question answering, and more [59]. Transformer models has significantly improved the performance of language models.
Transformers	Transformer models are a type of neural network architecture. The defining feature of Transformer models is the self-attention mechanism, which enables the model to pay attention to different parts of the input when producing the output [33]. See also Section 3.4.

GPT	Generative Pre-trained Transformer (GPT) models are a family of large language models developed by OpenAI [43]. These models are trained on a large amount of text in a self-supervised manner, where the objective is to try and predict the next word based on the preceding words. By training on large amounts of data, the models learn statistical language patterns, such as semantics, syntax, and context. These pre-trained models can be fine-tuned on specific NLP tasks, such as translation, text classification, question answering, and chatbots (for example, ChatGPT [1]).
Zero–shot	Zero-shot is the ability to classify or understand data that was not seen during the model training [61]. For an ASR model this could mean the ability to correctly transcribe words it has never seen before and in languages it was not specifically trained on.

Table 3.1: Glossary of selected terms used the following exposition. Some of these concepts are expanded upon in the following sections.

3.4 Transformer models

A transformer model is a type of neural network used to process sequential data, for example, speech or natural language. It was first introduced in 2017 by Ashish Vaswani in the paper *Attention is all you need* [54].

The Transformer model architecture consists of an encoder-decoder structure. The encoder maps the input sequence $(x_1, ..., x_n)$ to a sequence of continuous representations $Z = (z_1, ...z_n)$. Based on the sequence Z, the decoder generates an output sequence $(y_1, ..., y_n)$ one element at a time. The model is auto-regressive at each step, using the previously generated symbols as input when generating the next. The key to the transformers models is the self-attention mechanism which allows the model to attend to different parts of the input sequence while generating the output sequence. The attention mechanism is computed from the dot-product of the input embeddings with trainable parameters, and these weights capture the relevance between the input token and output token. The model can dynamically adjust these weights to focus on different parts of the input. This enables the model to capture long-range dependencies between the distant parts of the input sequence, which differs from *Recurrent Neural Networks* (RNN) [57], which rely on fixed-length context windows.

3.5 Byte Pair Encoding Tokenization

Byte Pair Encoding (BPE) is a data compression algorithm that has been adopted to use as a text tokenizer in Natural Language Processing (NLP), iteratively merging the most frequent pairs of characters until the vocabulary size is reached or the desired level of compression [6].

[1] `https://openai.com/blog/chatgpt`

The algorithm computes the unique set of words used in the corpus, building the vocabulary using all the symbols used in the corpus to write those words. As an example, let us consider a corpus of five words: "hug snug pug hans bun pun". We would then follow the following steps to construct our vocabulary:

1. Construct base vocabulary: *'h', 'u', 'g', 's', 'n', 'p', 'a', 'b'*
2. Count frequencies of adjacent character pairs: *'hu': 1, 'ug': 3, 'sn': 1, 'nu': 1, 'pu': 2, 'ha': 1, 'an': 1, 'ns': 1, 'bu': 1, 'un': 2*
3. Merge most frequent pair: *'ug'*
4. Update the vocabulary: *'h', 'u', 'g', 's', 'n', 'p', 'a', 'b', 'ug'*
5. Repeat steps 2-4 until the desired vocabulary size or level of compression is reached.

3.6 Mel-frequency Spectogram

The Mel-frequency spectrogram is a visual representation of frequencies that make up sounds over time which is commonly used in areas such as automatic speech recognition. The Mel-frequency spectrogram is constructed by taking the original spectrogram (frequency at a given time with color as amplitude) [60] and applying a mathematical function called the Mel scale [46]. The Mel scale is a logarithmic function that is designed to provide a more accurate representation of the way humans perceive sound. The Mel scale, therefore, gives more weight to sounds with lower frequencies, as humans are more sensitive to lower frequencies. This results in a two-dimensional representation of the audio signal, with frequency on the y-axis and time on the x-axis. The color intensity at a given point indicates the amplitude of the sound at the time and frequency.

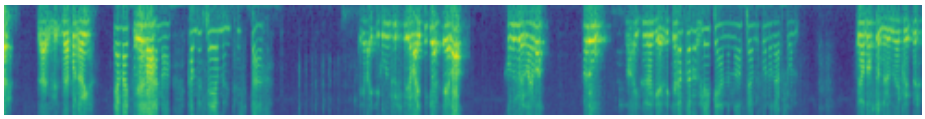


Figure 3.1: Example Mel Spectogram. This Mel-spectrogram represents sound frequency on the y-axis and time on the x-axis, with color intensity showing sound amplitude at any given point.

3.7 The Whisper system for automatic speech recognition

The Whisper model uses the basic encoder-decoder Transformer model, described in Section 3.4, since the focus was studying the capabilities of large-scale supervised pre-training for speech recognition and since this model has been validated to scale reliably [44].

The dataset is constructed by audio paired with transcripts from the internet, which results in a very diverse dataset consisting of audio from different setups, environments, languages, and speakers. Since many transcripts on the

internet are generated by existing Automatic Speech Recognition (ASR) systems, these kinds of transcripts were removed from the dataset since training on datasets consisting of mixed human and automatically generated data have been proven to impair the performance of translation systems [13]. Machine-generated transcriptions also often remove aspects that are difficult to predict, such as punctuation, commas, exclamation marks, question marks, paragraphs, and capitalization, which means that all-uppercase or all-lowercase is unlikely to be transcribed by a human.

The audio was resampled to 16,000 Hz, and the melspectogram was computed on 25 millisecond windows. The tokenizer for the English-only models is a BPE text tokenizer, which is the same tokenizer used in GPT-2 [48]. For the multilingual models, the vocabulary is refitted but kept the same size.

The Whisper model is created to perform the whole speech recognition pipeline, not only the recognition part. It is able to perform several tasks such as translating and transcribing several languages, predict no speech, and time stamps. For this to work in a single model, some task specification is required. A simple format is used to specify all tasks and conditioning information as a sequence of input tokens to the decoder. The start of a prediction is indicated by the <|startoftranscript|> token. The model starts by predicting the spoken language, which is represented as a special token (for example <|no|> for Norwegian bokmål). If no speech is detected, the model is trained to predict *no speech* represented as the special token <|nospeech|>. The next token is the given task, which is either <|transcribe|> or <|translate|>. To specify if the model should predict timestamps or not, the <|notimestamps|> is included. The prediction ends with a <|endoftranscript|> token.

Since the start of this project, a new speech recognition model has been introduced, the Conformer-1 by AssemblyAI [8]. The model is trained on 650,000 hours of labeled English audio. This model, similar to Whisper, is proven extremely robust to noise, which could improve the accuracy of emergency calls since these tend to contain some background noise. It also can do several different tasks, including labeling speakers and entity detection, which could be useful when transcribing emergency calls. However, since it is currently only trained on the English language, it is not a viable option for this project.

Chapter 4

Related work

Even though Automatic Speech Recognition (ASR) has been around for some time, it is in recent years that the performance of ASR systems has skyrocketed. This includes the application of ASR technology in emergency call centers, driven by the need to address the time-sensitive nature of such calls, requiring rapid and accurate decision-making. Given the rapid growth and widespread application of ASR worldwide, we have chosen to narrow the scope and concentrate on the development and applications of ASR in emergency call centers and public safety organizations in the Nordic region. We choose to focus on the development of ASR in Nordic languages because ASR technology for these languages is still lagging behind its English counterparts. This is because English is a widely spoken and dominant language in the technology sector, and therefore has naturally received considerable attention and resources in the development of ASR.

CORTI AI

A machine learning framework was developed to predict Out-of-Hospital Cardiac Arrest (OHCA) to increase the chance of survival by quickly recognizing OHCA [5]. The mean incidence of OHCA is 84 per 100,000 population per year [17], with the corresponding incidence rate of resuscitation is 56 per 100,000 population and a survival rate of 8% [18]. By performing cardiopulmonary resuscitation (CPR) and using an automated external defibrillator, the survival rate can increase in 7 of 10 cases of OHCA. Since time is critical in cases of OHCA, it is important that these cases are recognized as soon as possible. The aim of this study was to examine if a machine learning framework can increase the proportion of recognized OHCA calls within the first minute compared to dispatchers.

The machine learning framework consists of two machine learning models: an ASR model that transcribes the speech to text and a detection model that predicts OHCA from the transcribed text. The ASR model is based on Connectionist Temporal Classification [3], and it was trained on 45 hours of uniform randomly selected Swedish emergency calls from 2015 to understand the Swedish

language. The OHCA detection model is a densely connected deep neural network predicting OHCA based on the transcribed text from the ASR model. The model is trained on 3944 calls labelled as OHCA and 39,888 calls labelled as no OHCA.

We use a very similar approach to solve our problem, but there are still some differences. We use Whisper to transcribe Norwegian emergency calls, which is already trained on the Norwegian language and therefore does not require as much fine-tuning to get a good ASR model. Since CORTI is engaged in another domain, OHCA, several sections of the call might influence the model's prediction. While CORTI combines transcriptions and raw audio to predict OHCA, our approach concentrates solely on the call transcription.

Given the success of CORTI AI's ASR model for predicting OHCA, researchers in Denmark researched *how an ASR, at the EMS Copenhagen, could contribute to a more accurate stroke detection and impact the stroke-related treatment* [47]. Under the assumption that stroke detection rates would increase by the same amount as the OHCA detection rate through CORTI AI, patients treated with thrombolysis will rise by 5% within the group of patients calling within the time-to-treatment for thrombolysis. They found that using ASR for stroke detection would be particularly relevant for females, younger stroke patients, calls received through the 1813-Medical Helpline (an out-of-hours number providing direct contact with trained nurses and physicians), and on weekends. These are promising results given that the goal of the larger project, AISMEC, is to detect stroke patients using ASR.

AI4INTERVIEWS

Oslo Police District is currently developing an AI solution to streamline the task of transcribing interviews. The objective is to make the task of transcribing interviews more effective by designing, creating, and implementing AI-powered solutions for ASR and text analysis. [2]

Annually, the Norwegian police conduct thousands of interviews of various types, including organized crime, violence, child abuse, and financial crime. Typically, these interviews are either manually transcribed in whole or in part, or condensed into a summary, which is time-consuming and tedious work for investigators and police. Furthermore, the demand for interviews linked to these types of crimes is expected to increase continuously. Therefore it is crucial to address these challenges by employing automation technologies such as machine learning.

Even though this is a different domain, it is interesting for our project because AI4INTERVIEWS uses deep learning techniques to automatically transcribe Norwegian interviews.

AMK Simulator

RAKOS and Headroom Life Science AS are developing a tool called *EmergencyPerform* to increase the competence level of emergency call operators [1]. The tool is a 1:1 training simulator where the health personnel interacts with virtual patients developed using artificial intelligence. The project got granted

access to real emergency calls in November of 2021 and has been able to improve their speech-to-text algorithms. This project is very relevant to our work, not only because they are developing speech-to-text algorithms using real emergency calls but also because they are affiliated with the same health region as the current project, *"Helse Vest"*. Unfortunately, we have not been able to find more information about this project or establish any collaboration with them.

Part II

Experimental work

Chapter 5

Design and implementation

5.1 Fine-tuning Whisper

To make Whisper suitable for emergency calls in Norwegian, we are fine-tuning Whisper on simulated emergency calls. The Whisper model already performs well on Norwegian audio, so our goal is to make the model more robust for medical terms and improve performance on the poor audio quality from phone calls. How well this works mainly depends on the amount and quality of data we have available to train the model.

5.1.1 Data

The training data for the model consists of Waveform Audio File Format (WAV) files with a sampling rate of 16,000 Hz. It's important to specify the format of the audio files, as the characteristics of different file formats, such as compression, bit depth, and sampling rate, can significantly impact the quality and size of the audio data.

Actual emergency call recordings would be ideal for fine-tuning the model. Unfortunately, this could not be done because it would require access to the secure location of the recordings, in addition to the whole training loop would need to be running from inside that same location. This means we cannot access large amounts of data usable for training.

5.1.2 Augmentations and simulation

As a result of insufficient data, we devised an alternative to the original emergency calls. Initially, we acquired a simulated call by having a physician act as a patient and an actual EMCC operator answering. This simulated call was generated using the same system as a genuine emergency call, guaranteeing the same audio format and quality. The result was a six-minute audio clip that imitated a real-life emergency. Later, we got another call made in the same way. However, we would need more data than just these two simulated calls to enhance the model's robustness to such audio.

To replicate noise and variations that may arise in actual emergency calls, it may be beneficial to do some augmentation on the data. Our first step was to record noises from outside traffic to layer on top of our simulated call. This is to provide audio examples with background noise to ensure the model can perform well in such cases.

Even if we split up our one emergency call into smaller bits to get more clips to train on, they will all have the same speakers. This may create some bias as dialects and language quality do not change much across all the training data. The best solution to prevent this is to acquire more data from different speakers. As we can't train on actual emergency calls at this time, we chose to create more audio ourselves based on the simulated call. This provides a small amount of additional data containing speakers with different voices and dialects. This audio was created using an audio recorder on the phone with both speakers in the same room. To create some diversity between the two speakers, one was put closer to the microphone than the other.

5.1.3 Training

Fine-tuning is done by first preparing the small amounts of available data. The simulated calls are first manually transcribed so that they are labeled. Then the audio clips are split into smaller chunks, shorter than 30 seconds per chunk. This is necessary as the Whisper feature extractor will truncate longer audio samples to 30 seconds. The new audio samples are then loaded into the training loop with an AudioFolder[1] dataset loader from Hugging Face. This dataset now contains all the audio segments with their matching transcriptions.

A small part of the dataset is set aside for evaluation after the model is fine-tuned. The rest is sent through the processing steps to make the data usable for the Whisper model. The audio must be converted to log-Mel spectograms, and the transcriptions must be translated to label ids with the tokenizer before training starts.

5.1.4 Evaluation

The effectiveness of fine-tuning Whisper on our own Norwegian audio may be evaluated in different ways, both qualitative and quantitative. The most obvious way to measure the quality of a speech-to-text model is to look at the Word Error Rate (WER) on audio samples the model has not seen before. This is also true in our case, as it is important that the model maintains a high accuracy after fine-tuning. What this does not tell us anything about is the accuracy of which it can correctly guess more important medical terms or words that might be unseen by the vanilla Whisper model. One key consideration is that there may be multiple correct ways to transcribe a single audio clip. Two transcriptions could have different formats, or one might include more filler words than the other, yet both could still be considered correct by humans. This becomes particularly relevant for models like Whisper, which are designed to function across various fields with different transcription formats. Whisper addresses this challenge to a degree by standardizing the text prior to calculating

[1] `https://huggingface.co/docs/datasets/audio_load`

the WER, thus reducing the negative impact of non-semantic differences [44]. Given our limited amount of training data, we cannot expect any major impact on the quantitative metrics. Therefore, human inspection of the results might be a more effective way to evaluate the quality of the fine-tuned model in our scenario.

We are also evaluating the feasibility of the fine-tuning process. This involves the exploration of practical aspects such as the required resources, the complexity involved, and the time commitment for fine-tuning. Understanding these factors is important in order to facilitate a fine-tuning pipeline inside the same secure environment where the data is stored.

5.2 Transcription web app

As a proof-of-concept, we developed a web application to showcase how Automatic Speech Recognition (ASR) could streamline the manual task of transcribing historical emergency calls. The result was a web application where the user is able to upload an audio file, which is then transcribed using ASR.

Since this web application is mainly developed for streamlining different manual transcription tasks within *"Helse Vest"*, it is important to focus on functionality that could be useful for the different tasks at hand. For example, *Seksjon for medisinsk dokumentasjon, Dokumentasjonsavdelingen ved Haukeland universitetssjukehus* is currently manually transcribing historical emergency calls formatted to indicate which person is speaking at a given time. This could also be useful in the case of supervisory cases, in addition to timestamps which makes it possible to play the audio clip together with subtitles.

Since this application will be used to transcribe real emergency calls, the solution must run within the same secure environment as the recordings are stored, which comes with some constraints the system must abide by. OpenAI provides an Application Programming Interface (API) for the Whisper system, but there are several reasons why using this API is not a viable solution. First, because of the sensitive nature of the data, we cannot upload the recordings to servers outside the environment where recordings are stored. This API also does not allow us to fine-tune the model, which means adapting the model to Norwegian emergency calls is impossible. Emergency calls also vary in length, and since the API is limited to 25MB file uploads, this may be a limiting factor. Hugging Face also provides an inference API, but this comes with the same restrictions as OpenAI's API, the only difference being that you can make inferences using your own fine-tuned models. With these constraints in mind, we developed a web application that should be able to run inside the *"Helse Vest"* environment with minimal adjustments.

5.2.1 Architecture and implementation

The web application consists of a front-end developed using React and TypeScript that communicates with the back-end, created using the Python framework Flask. The front-end is responsible for handling the representation of logic and is also the part of the application the client can interact with. The back-

end is responsible for handling the business logic. This includes doing speech recognition and constructing the response.

To fetch resources from the back-end, we use the Fetch API [12] to access resources across the network. When the user uploads an audio file, the function will set the variable *audio* with the uploaded audio file, create an object URL, and assign this URL to the source property making it possible to play the audio in the browser, see Listing 5.1. When pressing the *Transcribe* button, provided there is an audio file, the function sends the file and the file name to an API, see Listing 5.2, where a new FormData object is constructed using the audio file and the file name. Then the Fetch API sends a POST request to the back-end using the FormData object as the request body. This is done with an asynchronous function using the async/await syntax in TypeScript. Since the await keyword is present, the function is paused until the request is completed. If an error occur during the transcription, an alert will be presented to the user.

When the back-end receives a POST request with the audio file in the request body, we can use ASR to transcribe the audio. When done, the back-end responds with a JSON object with two elements: the raw transcription with speaker tags and the transcription with timestamps formatted as SubRip Subtitle (SRT).

```
1  const uploadAudio = async (audio: File) => {
2          if (audio) {
3              setAudioFile(audio);
4              const objectUrl = URL.createObjectURL(audio);
5              console.log(objectUrl);
6              if (audioRef.current) {
7                  audioRef.current.src = objectUrl;
8              }
9          }
10     };
```

Code Listing 5.1: This code snippet defines an asynchronous function, 'uploadAudio', which accepts an audio file as input. If a valid file is provided, the function will set the audio file, create an object URL for it, and then assign this URL to the current source property of a referenced audio object.

```
const transcribeAudio = async (file: Blob) => {
        setLoading(true);
        setDocTranscript("");
        if (!file) {
            setLoading(false);
            return;
        }
        try {
            const response = await API.transcribeAudio(file,
                audioFile!.name);
            setLoading(false);
            setDocTranscript(response.doc_text);
            setSrt(response.srt_text);
        } catch (error) {
            toast({
                title: "An error has occurred.",
                description: "An error occurred during the
                    transcription. Try again later.",
                status: "error",
                duration: 5000,
                isClosable: true,
              });
        }

    };
```

Code Listing 5.2: This code snippet defines the 'transcribeAudio' asynchronous function, which takes an audio file as a Blob object as input. Initially, it sets the loading state to true and clears any previous transcript. If no file is provided, it stops the loading state and returns. If a file is provided, it sends the file to an API for transcription. After receiving the response, it stops the loading state and sets the received text transcript and SRT format to the corresponding state variables.

To do the ASR with speaker diarization, we load the Whisper and pyannote.audio speaker diarization model [40] into memory from local storage using Hugging Face's pipeline function. This is to avoid loading the models each time a request is made, which would significantly increase response time. Using Hugging Face's pipeline, we create a pipeline object that abstracts most of the complexity from the library.

```
whisper_pipeline = pipeline(
  "automatic-speech-recognition",
  model="<PATH_TO_TRANSCRIPTION_MODEL>",
  chunk_length_s=30,
  device=device,
  generate_kwargs={
        "language":"<|no|>",
        "task":"transcribe"
  }
)

dz_pipeline = Pipeline.from_pretrained(
    '<PATH_TO_DIARIZATION_MODEL>',
    use_auth_token="<AUTH_TOKEN>"
    )

@app.route('/transcribe', methods=['POST'])
def transcribe_audio():
    audio_file = request.files['file']
    file_name = "temp-data/"+audio_file.filename

    audio_file.save(file_name)

    transcription = whisper_pipeline(file_name, return_timestamps=
        True)["chunks"]
    dz = dz_pipeline(file_name, min_speakers=2, max_speakers=5)

    os.remove(file_name)

    speaker_list = get_speaker_list(dz)

    labeled_transcriptions = label_transcriptions(transcription,
        speaker_list)

    srt_text = generate_srt_text(labeled_transcriptions)
    doc_text = generate_doc_text(labeled_transcriptions)

    data = {}
    data['doc_text'] = doc_text
    data['srt_text'] = srt_text


    return jsonify(data)
```

Code Listing 5.3: ASR and speaker diarization model is loaded into memory. The end point receives an audio file, transcribes it, and returns transcription with speaker tagging and as SRT.

The speaker tagging of each text segment that the Whisper model produces is done with the function *label_transcriptions()* as shown in Listing 5.5. It takes a list of transcribed segments from the Whisper model and a list of segments with labeled speakers from the *get_speaker_list()* and matches each transcription to a speaker. The speaker segments serve as "buckets" so that if a transcription is within the range of one speaker segment, that transcription is labeled with that speaker. Each transcription segment is represented by the median of start and end times. This ensures that each transcription segment can only be put into one "bucket".

The speaker list, as displayed in Listing 5.4, transforms the output generated

by the pyannote.audio diarization model into more meaningful speaker labels, such as “MO” (“Medisinsk Operator”; eng: “Medical Operator”) and “I” (“Innringer”; eng.: “Caller”). It also tags additional speakers with “I” followed by a number.

```
def get_speaker_list(diarization):
    speaker_list = []
    speaker_dict = {}


    for track in diarization.itertracks(yield_label=True):

        # Find the speaker labels
        if len(speaker_dict) == 0:
            speaker_dict[track[2]] = "MO"

        elif track[2] not in speaker_dict.keys():
            if len(speaker_dict) > 1:
                speaker_dict[track[2]] = "I" + str(len(speaker_dict
                    ))
            else:
                speaker_dict[track[2]] = "I"


        speaker_list.append([track[0].start, track[0].end,
            speaker_dict[track[2]]])
    return speaker_list
```

Code Listing 5.4: This Python code snippet defines the function *get_speaker_list*, which takes the diarization output from the pyannote.audio model as input. This function iterates over the tracks in the diarization output and assigns meaningful speaker labels to each track. The first speaker is labeled as *MO* (Medical Operator), and subsequent speakers are labeled as *I* (Caller) or *I* followed by a number, for additional speakers. The function ultimately returns a list of speaker segments, each represented by start and end times and the corresponding speaker label.

```python
def label_transcriptions(transcriptions, speakers):
    labeled_transcriptions = []

    for transcription in transcriptions:
        # If the median of transcription is inside of timerange of speaker, then label it with that speaker
        transcription_median = (transcription["timestamp"][0] + transcription["timestamp"][1]) / 2
        closest_speaker_index = find_bucket(speakers, transcription_median)
        if closest_speaker_index == 0:
            # Do the same but round down the median to closest integer
            transcription_median = int(transcription_median)
            closest_speaker_index = find_bucket(speakers, transcription_median)


        labeled_transcriptions.append({
            "timestamp": [transcription["timestamp"][0], transcription["timestamp"][1]],
            "text": speakers[closest_speaker_index][2] + ": " + transcription["text"],
            "speaker_label": speakers[closest_speaker_index][2]
        })

    return labeled_transcriptions

def find_bucket(speakers, transcription_median):
    closest_speaker_index = 0
    for i, speaker in enumerate(speakers):
        if transcription_median >= speaker[0] and transcription_median <= speaker[1]:
            closest_speaker_index = i
            break
    return closest_speaker_index
```

Code Listing 5.5: This Python code snippet defines two functions, *label_transcriptions* and *find_bucket*. The *label_transcriptions* function accepts two parameters: transcriptions and speakers. It assigns speaker labels to each transcription segment based on the median timestamp of the segment. If the median timestamp falls within the time range of a speaker's speech, the corresponding speaker label is appended to the transcription. The function returns a list of labeled transcriptions, each with start and end times, the transcribed text with speaker label, and the speaker label itself. The *find_bucket* function finds and returns the index of the speaker whose speech segment's time range includes the given median timestamp. If no such speaker is found, it returns the index 0.

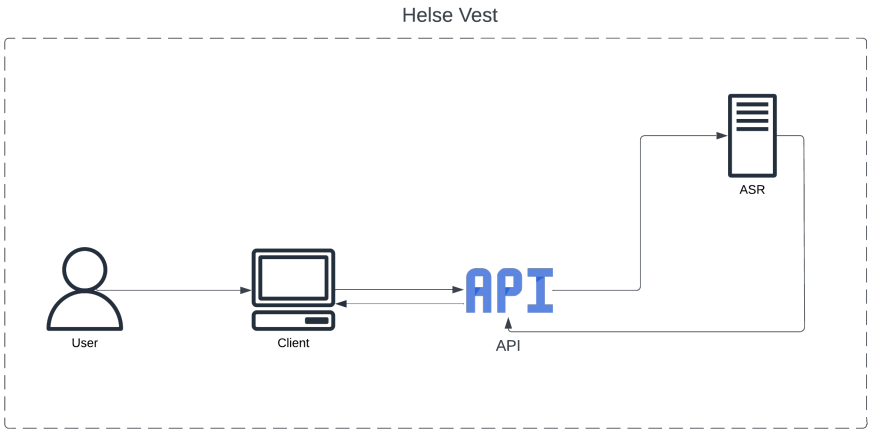


Figure 5.1: Web application architecture. The user uploads audio, which is sent to the back-end for ASR. The back-end returns the transcribed audio.

5.2.2 Technology stack

To develop a web application with the features we want to implement, we need to decide on a technology stack that is able to provide us with the functionality we need. We need front-end technologies to develop the front-end that the user can interact with, and we need back-end technology that can handle requests from the front-end and give an appropriate response. We also need technology to load the pre-trained Whisper model and to fine-tune the model on our data. Therefore we have decided on the technology stack described below.

React

React is an open-source library for JavaScript released in 2013 by Facebook for building User Interfaces (UIs) and web applications. React applications are based on components, which are reusable UI pieces that can be composed together to make web applications [45].

Chakra UI

Chakra UI is an open-source component library for React. It provides a variety of components that are customizable, accessible, and reusable. These components are highly customizable and can be manipulated by passing properties to the components. Using Chakra UI eliminates the need to write CSS [7].

TypeScript

TypeScript is a subset of JavaScript that was created to be a static typechecker for JavaScript programs, which means that it is a tool that runs before the code (static) and ensures that the types are correct. This is a powerful tool because the most common errors that occur are type errors: a value was used where a different kind of value was expected [19]. Since JavaScript is a dynamically typed language, types are determined at runtime and are prone to runtime errors, while errors are uncovered during compilation in TypeScript.

Hugging Face

Hugging Face is a platform where you can build, train and deploy machine learning models. Hugging Face's Transformer library is an open-source library for building state-of-the-art deep learning models built on top of PyTorch and TensorFlow frameworks. It includes pre-trained models that can be fine-tuned for specific tasks such as text classification, question answering, and speech recognition. The Transformer library can be installed as a Python package, making it possible to train models with as little as three lines of code [52]. These models can then be shared to the Hugging Face Hub [23] where you easily can do inference using your model via the Hugging Face Inference API [25] using simple API calls.

PyTorch

PyTorch is a machine learning library for building and training deep learning models developed by Facebook's AI Research group [41]. In addition to the Python interface, PyTorch supports a range of hardware platforms, such as GPUs, CPUs, and TPUs. PyTorch also supports distributive training, which allows parallel processing across multiple GPUs.

Flask

Flask [56] is a Python web framework based on the Wergzeug[2] toolkit and Jinja2[3] templating engine. With Flask, it is easy to define routes and bind them to functions that is executed when the route is accessed.

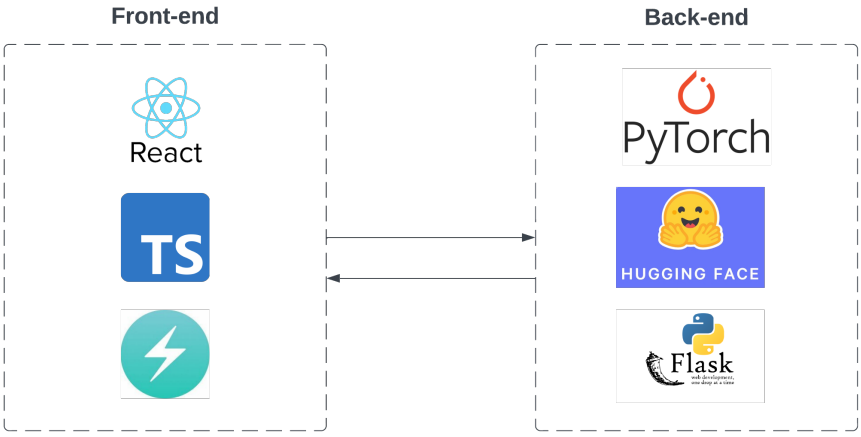


Figure 5.2: Web application technology stack. Front-end consisting of react, TypeScript, and Chakra UI. Back-end consisting of PyTorch, Hugging Face, and the Python framework Flask.

[2] https://werkzeug.palletsprojects.com/en/2.3.x/
[3] https://palletsprojects.com/p/jinja/

Chapter 6

Results

This chapter presents the evaluation and results of the research and experiments conducted as part of this master's thesis. The primary objective of our evaluation was to assess the effectiveness and usability of the application. The evaluation was performed using quantitative and qualitative methods, including performance metrics, manual inspection, usability tests, and user feedback.

6.1 Fine-tuning

Whisper is already trained on enormous amounts of data, and because of that, little fine-tuning is required to create a robust model, even though Norwegian is considered a low-resource language. After fine-tuning the model on approximately six minutes of simulated audio, we notice some minor changes (illustrated in Table 6.1). To evaluate performance on unseen data more extensively, we calculate the WER on the simulated call not included in the training. The results (as seen in Table 6.3) show a marginal improvement in the fine-tuned version of the Whisper large-v2 model compared to the original.

Ground truth	Fine-tuned	Whisper
MO: Ja. Er det sånn at hun føler hun holder på å besvime?	MO: Ja, er det sånn at hun selv holder på å besvime?	MO: Er det sånn at hun selv holder på å befime?

Table 6.1: Example results from comparing a vanilla Whisper model against Whisper fine-tuned on emergency call data. The results are generated by uploading the same audio to the application two times. Each time with a different model.

Ground truth	Fine-tuned	Whisper
I: Jeg er pjuskete selv, så jeg klarer ikke løfte opp Margit. Hun er stor dame. MO: Svarer hun ordentlig når du snakker til henne? I: Margit? Ja, Margit? Ja da, hun svarer. Hun sier hun har vondt da. Hun har slått seg litt. Hun har vondt i hofta si sier hun.	I: Jeg husker ikke selv, så jeg klager ikke å løfte opp Margit, hun er en store dame. MO: Ja, svar hun ordentlig når du snakker med henne? I: Margit, ja, Margit. Ja da, hun svarer. Hun sier hun har vondt da hun har slått seg litt. I: Hun har vondt i hofta, sier hun.	I: Jeg husker det selv, så jeg klarer ikke å løfte opp Margit, sin store dame. MO: Ja, og svarer hun ordentlig når du snakker til henne? I: Margit, ja, Margit. I: Ja, da, hun svarer. Hun sier hun har vondt da hun har slått seg litt. Hun har vondt i hofta sitt i fyrhund.

Table 6.2: Example results from the same test as in Table 6.1. These are longer examples that show the two models' ability to reproduce the original message.

Model	**WER**
Fine-tuned	32.9443 %
Whisper large-v2	33.4829 %

Table 6.3: Resulting Word Error Rate (WER) from testing both the fine-tuned and original Whisper large-v2 models on an unseen simulated emergency call.

The results from training (illustrated in Table 6.4) show how the model improves on the training set during training. We stopped the training after ten epochs, as we observed no further improvements beyond this point. Continuing to train for a longer period, especially with a limited volume of training data, risks leading to overfitting, which occurs as the model becomes increasingly specialized in fitting the training data, reducing its ability to generalize to unseen data.

The training was done on a machine with an AMD Ryzen Threadripper Pro 5975WX CPU, 256GB memory, and an NVIDIA RTX A6000 GPU. Because of the model size, training consumed almost the entirety of the GPU's 48GB VRAM. It took 4 minutes and 18 seconds to train the model over 10 epochs using roughly 7 minutes of audio data.

Training Loss	Epoch	Step	Validation Loss	WER
1.2935	1.0	1	1.0842	38.2653
1.2935	2.0	2	1.0339	35.7143
1.1811	3.0	3	0.9826	33.1633
0.9953	4.0	4	0.9103	28.5714
0.7994	5.0	5	0.8852	28.0612
0.7257	6.0	6	0.8624	28.5714
0.6560	7.0	7	0.8423	28.8265
0.5976	8.0	8	0.8274	28.5714
0.5572	9.0	9	0.8110	27.8061
0.5096	10.0	10	0.8049	28.0612

Table 6.4: Performance metrics for each epoch during model training.

6.2 Transcription web app

The web application was constructed as an easy and accessible way to access and use the fine-tuned Whisper model. The end product is a web application that can transcribe a single audio file.

6.2.1 Functionality

The core functionality of the web application is to transcribe an uploaded WAV file using the fine-tuned Whisper model. The application has a simple interface, shown in Figure 6.1.

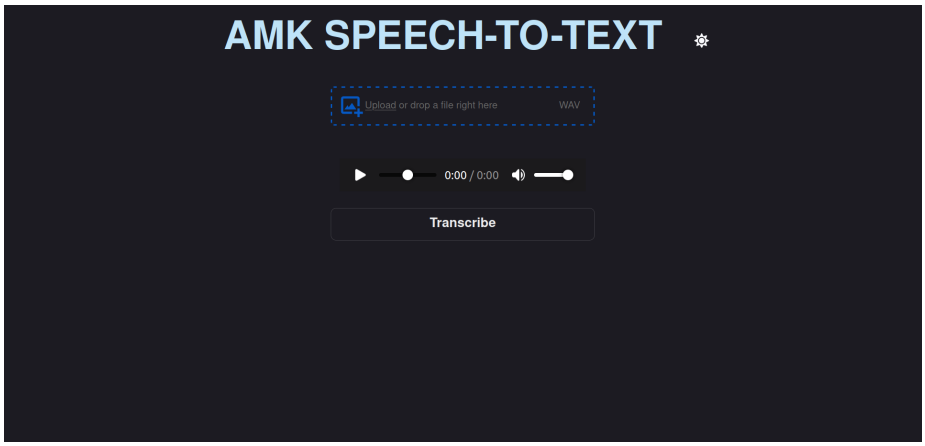


Figure 6.1: Web application home page, with the option to upload a WAV file, listen to it, and transcribe it.

After hitting the *Transcribe* button, the transcribed audio is returned, together with speaker tags, and displayed in an editable text window. The user can then download the transcription, either as a DOC or SRT file, shown in Figure 6.2.

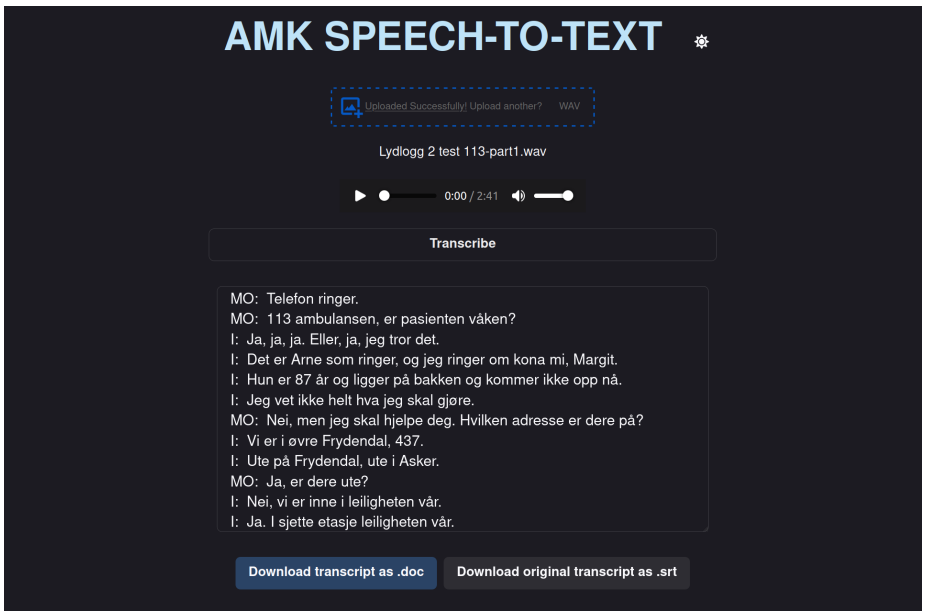


Figure 6.2: Web application with transcribed audio, with the option to download the transcription as a DOC or SRT file.

The source code for this application and a video demonstration can be found in Appendix A.

6.2.2 Speaker Tagging

In order to incorporate speaker tagging into the application, we matched the results from the pyannote.audio speaker diarization model with the transcriptions from Whisper and got a suggested speaker for every transcription segment from Whisper, visible in Figure 6.2. Evaluating the accuracy was done by manually counting the number of correct and incorrect speakers, presented in a confusion matrix in Figure 6.3.

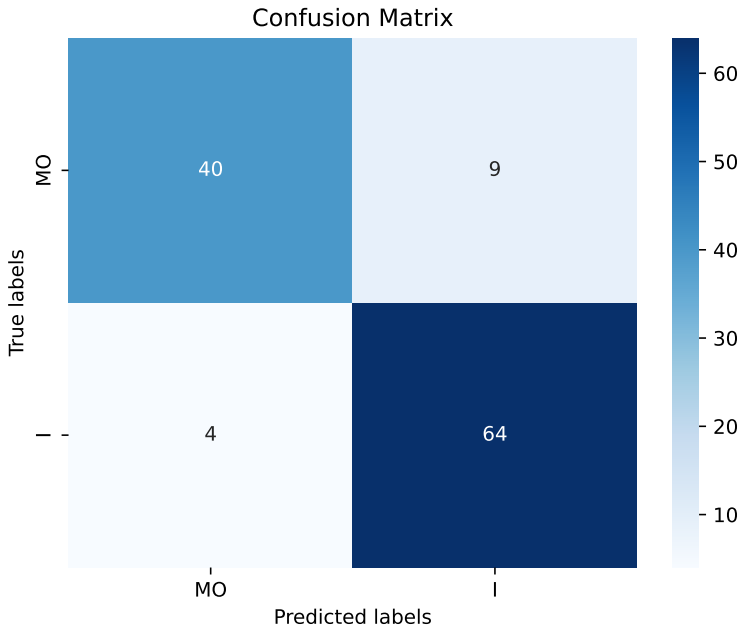


Figure 6.3: Confusion matrix illustrating the accuracy of speaker tagging by comparing manual annotations with the suggested speakers obtained from the integration of the pyannote.audio speaker diarization model and Whisper transcriptions.

6.2.3 Metrics

To evaluate the application's performance, we have timed the execution time for audio clips whose lengths varied from 0.75 minutes to 4 minutes, as shown in Figure 6.4. Given these inputs, the execution time varies from 10.2 seconds to 58.8 seconds. By observing the graph, we can see that as the input size increases, the run-time increases, as anticipated. We can also see that the increase in run-time seems relatively linear, indicating that the model maintains a consistent performance level as the input size increases. This was done on a machine with an AMD Ryzen Threadripper 1950X CPU, 64GB memory, and an NVIDIA GeForce GTX 1080 Ti GPU, with the models running on the GPU.

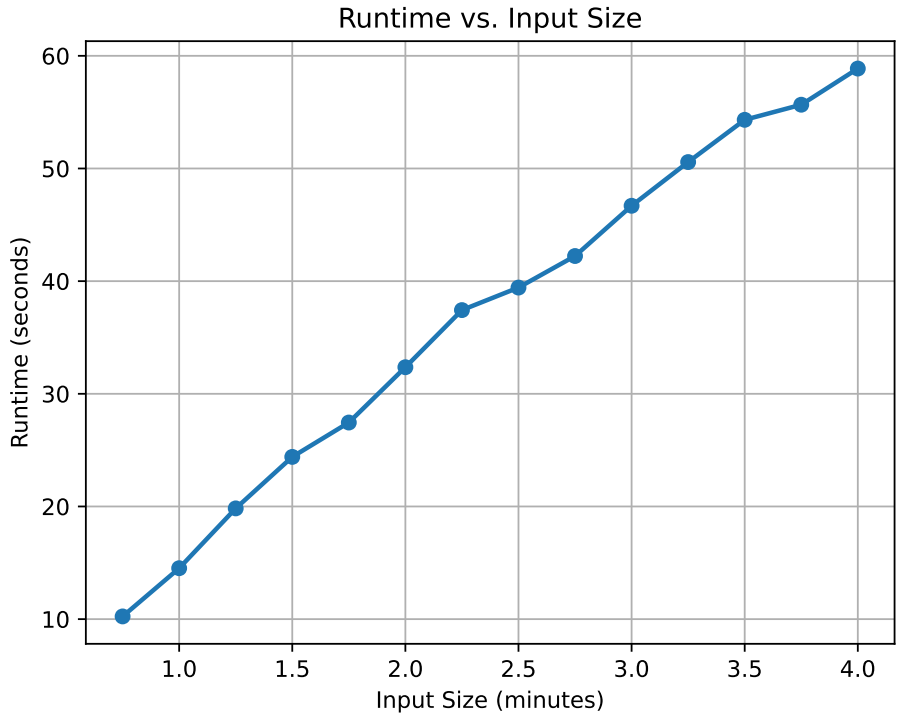


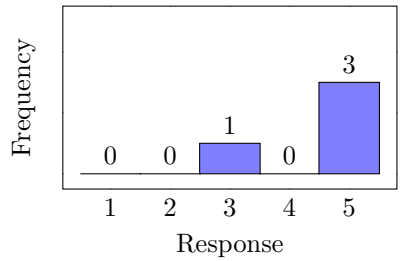
Figure 6.4: This line plot depicts the relationship between input size (in minutes) and computational runtime (in seconds). As input size increases, the computational runtime also increases.

6.2.4 Feedback from demonstrations of application prototypes

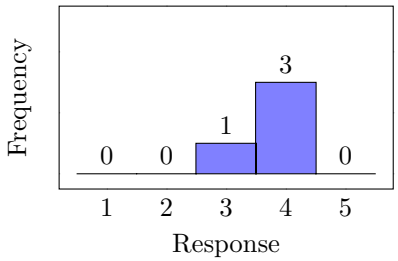
When developing a web application, it is vital to get feedback from the potential users of the application. This could be feedback about what features to prioritize, validate design choices, and enhance the user experience. Therefore, we organized two demonstrations for two sets of stakeholders: the researchers and developers involved with the research project *"AI-Support in Medical Emergency Calls (AISMEC)"* and personnel from a group tasked with transcribing audio at the hospital.

6.2.5 Demonstration for researchers involved with the AISMEC project

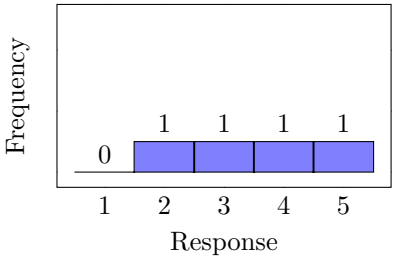
Early in the development, we had a demonstration for researchers in the "AI-Support in Medical Emergency Calls (AISMEC)" project. Although they were not the application's target users, their familiarity with *Seksjon for medisinsk dokumentasjon's* work enabled them to provide valuable feedback. We gave them three questions and the participants could respond on a scale from 1 to 5, where 1 representing strong disagreement and 5 indicating strong agreement, shown in Figure 6.5.



(1) How likely would you be to use such a solution to assist with the manual transcription of audio logs?



(2) How satisfied are you with the design of the application?



(3) How satisfied are you with the functionality of the application?

Figure 6.5: Answers from involved researchers in the AI-Support in Medical Emergency Calls (AISMEC) project.

Question 4: Do you have any other comments or feedback on the application? Feel free to provide a brief explanation for your answers as well.

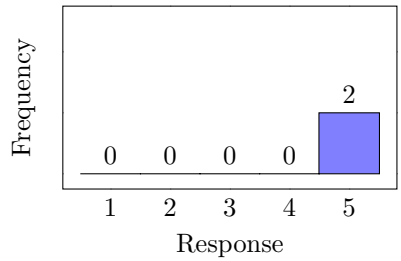
"Such a tool can be used for many different things and will be of great help and streamlining of various work processes."

"I've only seen a demo on the screen, but it looks very promising!"

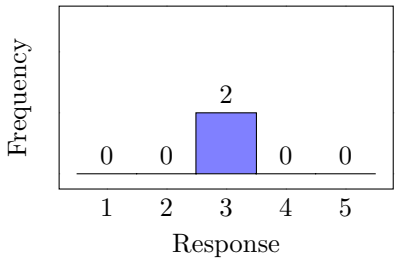
"It seems to be efficient and accurate. Simple design with a black screen, but probably just fine, as long as the user interface is easy to understand and straightforward. Nice with the box that came up with the text, as well as the option to download."

6.2.6 Demonstration for professional audio transcribers, HUS

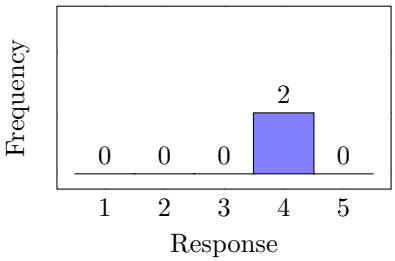
After demonstrating our applications to *Seksjon for medisinsk dokumentasjon*, we got some helpful feedback, shown in Figure 6.6. For the three first statements, the participants could respond on a scale from 1 to 5, where 1 representing strong disagreement and 5 indicating strong agreement.



(1) The use of this technology will lead to faster transcription.



(2) The use of this technology will lead to more accurate transcription.



(3) The implementation of this solution will contribute to a better workday for you.

Figure 6.6: Answers from Seksjon for medisinsk dokumentasjon.

Comments question 1

"Absolutely, it will facilitate the task. We have speech recognition in Helse Bergen, assisted speech recognition, doctors dictate, secretaries edit. It seems to be the same solution. Work on building up a vocabulary list."

Comments question 2

"Yes, that might be the case. There are different people transcribing at Seksjon for medisinsk dokumentasjon. There is some variation in how they perceive things."

"I chose 3 because if the word 'WILL' is asserting something that's going to happen, I'm unsure about it based on the experience that the company's speech recognition wasn't as good as we initially thought."

Comments question 3

"The solution will enable us to spend less time on the task and accomplish more tasks, as well as most likely make it somewhat easier for secretaries to listen to a suggested text rather than having to write the entire text themselves. We are dependent on work tasks, so considering that we might have less work, it's not necessarily a better workday."

"Yes, that could be the case. Secretaries do enjoy writing. As for time usage: yes, absolutely."

Question 4: Even though the solution is developed for EMCC, do you think it can be useful for other tasks? If so, which ones?

"It's difficult to answer, as they already have a speech recognition system for patient documentation. I don't know if one is better than the other. What they have now has been in use for a very long time. It's only in the last few years that it has become more similar to what was assumed it could do 10 years ago. I see that more and more people are using it. However, the existing speech recognition system has its weaknesses. Score 5 if there wasn't already a speech recognition system, 3 since there is one. It depends on whether it's better or not."

Question 5: If you were to use this application, do you have any suggestions or proposals for improvements?

"As simple as possible; the fewest possible buttons. Those that are there should be very clear. No mix of Norwegian and English; either one or the other. As simple as possible; no flashing lights, etc."

6.2.7 A preliminary semi-automatic transcription experiment

In order to evaluate the efficiency of utilizing the web application to assist in transcribing emergency calls, we conducted a preliminary experiment where we compared the time spent on transcribing both with and without the assistance of the application. The experiment was performed by ourselves, who had not previously listened to the full audio recordings we were going to transcribe. Both of us shared similar prior experiences, having each transcribed a single simulated emergency call, and we also both used the same setup for transcribing throughout the experiment. The audio clip was split into two equal parts and each of us had assistance on a different part. This was done to minimize the problem of one part being easier to transcribe. The timed results (illustrated in Table 6.5) show that both improved significantly when using the application.

	Semi-automatic	**Manual**	**Improvement**
Author 1	23:07 (Part 2)	31:17 (Part 1)	8:10 (35.33 %)
Author 2	15:49 (Part 1)	24:43 (Part 2)	8:54 (56.26 %)

Table 6.5: Results from the transcription experiment in minutes, including transcription time and improvement. The time the model used to generate a transcription is taken into account in the semi-automatic results. This means the results reflect the time used in a real scenario, from loading an audio clip to producing the corresponding transcription.

Part III

Discussion and conclusion

Chapter 7

Discussion, conclusion, and further work

In this chapter, we will discuss the findings of our study, which centered around our three research questions. In our first research question (RQ1), we tried to determine the feasibility of constructing a system for accurately transcribing Norwegian emergency calls by fine-tuning OpenAI's Whisper model on actual emergency calls. Our second research question (RQ2) focused on which other steps, such as source separation, could improve the resulting system developed in RQ1. Finally, research question three (RQ3) explored the technological and infrastructural requirements for implementing such a system inside the hospital and emergency call centers. We will also discuss further steps that can be done to build upon this research.

7.1 Fine-tuning Whisper

To answer RQ1 regarding building a system for accurately transcribing Norwegian emergency calls, we first needed a model that would be good enough for this task. For this, we used the Automatic Speech Recognition (ASR) model Whisper, which can already transcribe Norwegian audio out of the box. We researched the potential to improve this model by fine-tuning it so that it would perform better on audio from Norwegian emergency calls.

Fine-tuning Whisper was done by using the Hugging Face library. As previously mentioned, Hugging Face streamlines the process of utilizing advanced technologies needed to load and train a model. This lets us focus on the more important task of providing a sufficiently accurate model rather than spending time setting up the fine-tuning. Unfortunately, we had minimal amounts of data available, making it hard to produce good fine-tuning results. However, exploring this to demonstrate the feasibility and discuss the possible effects of fine-tuning using larger-scale real emergency call transcriptions was still an important part.

Information about what hardware is needed is important in order to set up the fine-tuning loop inside the same secure environment where the real emergency

calls are stored. Whisper is a rather large model in terms of the number of parameters and thus resource needs. We found that fine-tuning required a lot of GPU memory, consuming approximately the entire 48GB VRAM we had available. However, it is hard to pinpoint the exact amount of memory that is needed. It might also be a possibility to tune the training arguments in order lower the memory requirements. This could come at the cost of the time it takes to train the model, which is even more relevant when training with much larger sets of data than what we managed to do in this research. To evaluate and use the model, all we need to do is load it, which requires roughly 10GB of GPU memory. This means that getting transcriptions for real emergency calls is less demanding when we're simply using an already-trained model.

Unsurprisingly, given the small amount of data, the results from the fine-tuning show that the performance difference compared to the original model is not substantial. It is challenging to determine whether the variations in the resulting transcript have positively impacted the model or if they simply are the product of the systems' randomness. The phrases that differ between the models are not medical phrases in particular or phrases that were widespread in the training data. The small positive change in WER also does not necessarily mean that the model has improved. As previously explained, the WER may not always serve as a reliable metric. In fact, in some instances, it might yield poorer results for a model that performs better in practical applications. This leads us to believe that while our small-scale fine-tuning experiment altered the model slightly, it did not significantly improve its performance.

Although the data used for fine-tuning is simulated, it was based on real emergency calls, and the majority was recorded with the same system employed by Emergency Medical Communication Center (EMCC). Moreover, an actual EMCC operator acted out the part of the medical operator. This implies that the performance should closely resemble the actual outcomes when testing the model on genuine emergency calls. However, the only data available for our analysis were two simulated emergency calls, along with the recordings we created ourselves. A limitation of this approach is that both simulated calls were performed by the same individuals, which could potentially introduce bias when evaluating the model using the second simulated call. Another potential issue is the variability in speech patterns among individuals in real emergency calls. Our research primarily focuses on *"Helse Vest"*, thereby limiting the distribution of different Norwegian dialects likely to be encountered. However, we must still account for these dialects and accommodate non-native Norwegian speakers. Addressing this concern would likely involve fine-tuning the model on a diverse range of real emergency calls to improve its adaptability and effectiveness, an effort that is currently underway.

In the paper *"Robust Speech Recognition via Large-Scale Weak Supervision"* [44] accompanying the release of OpenAI's Whisper model, they clearly state that the goal of Whisper is to create a system that can perform well without the need for dataset-specific fine-tuning. The results from the same paper show that zero-shot Whisper models come very close to human robustness. In contrast, the supervised models trained on LibriSpeech make about twice as many errors as humans on datasets they have not been trained on. This shows that models based on Whisper hold a unique advantage when transcribing a vari-

ety of real emergency calls where unpredicted speech patterns can occur. This, however, does not eliminate the effect of fine-tuning completely. The same paper accompanying the Whisper model also reported that fine-tuning directly correlates with the model's zero-shot performance on that language. From this, in addition to how we have seen Whisper perform in our research, this model seems to fit very well for this specific task.

7.2 Transcription web app

To fully answer RQ1, we decided to develop a simple web application using OpenAI's Whisper model, fine-tuned on our own data. To also try and answer RQ3, we had to make deliberate choices of what technologies to use in building the system. We developed a front-end using React, TypeScript, and Chakra UI. Initially, we hosted the model on Hugging Face and accessed the model using Hugging Face's Inference API. However, we quickly became aware of the limitations of this approach, which included loading time for the model to be initialized, input size limitations, and the fact that real emergency calls can not be uploaded to a remote server for ASR. We then decided to develop and host our own back-end using the Python framework Flask and Hugging Face's pipeline functions.

There are several benefits to developing and hosting our own back-end. First, we have complete control over the features and functionality, allowing us to tailor the system to this project's requirements. In the early stages of the development, we used Hugging Face's Inference API, which came with limitations that made it unsuitable for this project, such as the file size being restricted to 25MB and requiring frequent model loading, which took around two minutes. We are also able to integrate it with the existing systems at *"Helse Vest"* since it has to run within the same environment as where the emergency calls are stored. Hugging Face's Inference APIs work by uploading the audio to a remote server, which is impossible with emergency calls since they can't be stored outside of the secure environment. By building our own back-end, we eliminate the restrictions that came with Hugging Face, resulting in no loading time since the models are already loaded into memory, no file size limitations, and we can integrate it within a secure environment with access to the emergency calls.

Further steps to improve the system, as requested by our RQ2, primarily consisted of implementing speaker tagging with the pyannote.audio speaker diarization model. The results indicate that the speaker tagging performs reasonably well, and its few errors do not significantly impact the performance for its current intended use, namely as an assistance for manual transcription. The biggest weakness of our implementation, and what makes up the majority of errors, is the identification of speakers when they speak in short or single-word utterances. For instance, if one speaker is saying affirmative words like 'yes' while the other is mid-sentence. Addressing these problems might require an improved speaker diarization model as it struggles to separate such tiny segments. It is still worth mentioning that our algorithm that matches Whispers transcriptions with a speaker segment might also have room for improvement. One limiting factor caused by how we have implemented speaker tagging is that we base all tags on the segments Whisper produces using its own voice activity detection.

If Whisper detects speech from two people as one speaker, we will still only give that segment a single speaker tag. We also know that Whisper can do multiple tasks by specifying input tokens to the decoder [44], which means that it could potentially support speaker tagging in the future. Further development and improvements in the EMCC systems might also lead to the emergency calls being recorded on two tracks instead of one, meaning that the two speakers will already be separated. Because of these considerations, we did not spend much time researching speaker tagging as the preliminary results proved satisfactory.

After demonstrating our first application prototype to the involved researchers in the *"AI-Support in Medical Emergency Calls (AISMEC)"* project, we felt that they were optimistic about our solution. Furthermore, they were enthusiastic about such a semi-automatic transcription system since it appeared to streamline the process of transcribing emergency calls, which is currently done manually.

After further developing our application, we demonstrated it to members of *Seksjon for medisinsk dokumentasjon* at Haukeland University Hospital. As illustrated in Figure 6.6, they were optimistic that the solution would streamline the task of transcribing emergency calls since they could correct a suggested transcription instead of manually transcribing the whole audio.

However, they raised some concerns about the possible consequences of increased efficiency. As their roles rely on work tasks, they questioned whether this solution would improve working conditions since the increased efficiency brought by such a solution could potentially lead to reductions in the workforce.

They were also curious about how the model would handle other languages, such as Swedish and Danish. Since Whisper is a multilingual model, it can also transcribe different languages, such as Swedish and Danish, with Word Error Rate (WER) 8.5 and 13.8, respectively, on the FLEURS dataset [15].

7.3 Semi-automatic transcription

To evaluate the efficiency of the application's primary use case, we manually transcribed two parts of an emergency call, one with and the other without the application's assistance. As illustrated in Table 6.5, we significantly reduced our transcription time. This is probably because correcting a suggested transcription is more efficient than transcribing the whole audio manually from scratch, even though the suggested transcription is not 100% accurate. Our experience was that having some initial text made it easier to follow along with the audio without having to pause or rewind as frequently. It also proved beneficial when speakers talked simultaneously, as the suggested text would help focus on each voice separately. Without its assistance, one might have to rewind and listen to the clip multiple times to comprehend and accurately transcribe the conversation in the proper sequence order. But this can also introduce some unwanted effects. Given the suggested transcription, the professional audio transcribers might be inclined to accept the automated transcription as correct, even when it contains errors, also known as *automation bias* [9]. We believe this will be particularly true when there are words or phrases in the audio that are very unclear, and the suggested transcription sounds about correct, but in reality, it

is not.

As with any research endeavor, this study has limitations, which should be considered when interpreting the results. As outlined in the introduction, our research employs a prototyping research methodology, though without the capacity to develop a fully functional prototype for use beyond our own. This constraint affects the extent and quality of the evaluation we can conduct on the resulting application, like having our target group perform a test. A significant part of this is due to the logistical challenge of moving the system from our local setup to the secure location with access to emergency calls, a task that requires effort from *"Helse Vest"* and is beyond our control. Thus, our research has aimed to deliver a proof-of-concept system rather than a fully implemented prototype.

7.4 Conclusion

In this study, we aimed to examine the feasibility of constructing a Automatic Speech Recognition (ASR) system to accurately transcribe Norwegian emergency calls by fine-tuning OpenAI's Whisper model on historical emergency calls, in response to our first research question (RQ1) (see Section 1.2). We found that it is feasible to construct a machine learning pipeline to train the model on actual emergency calls, and it demonstrated potential in improving the efficiency of transcribing emergency calls, hence showing promise for practical applications.

Regarding research question two (RQ2), we investigated source separation as a potential step to further improve the resulting system referred to in RQ1. We implemented speaker tagging to identify the different speakers in the transcription automatically. The results proved satisfactory for the current intended use, where most errors occur when they speak in short or single-word utterances.

Considering research question three (RQ3), we explored viable technologies and infrastructure to integrate such a transcription system within the hospital and emergency call centers. This resulted in the development of an ASR web application designed with modern technologies that could be migrated to these environments, streamlining the process of manually transcribing emergency calls.

It is also important to acknowledge the limitations of our research, such as the model only being trained on simulated emergency calls and the limited volume of the training data. We will discuss potential further work in the following section.

7.5 Further work

At the beginning of this project, the aim was to fine-tune the model on real emergency calls to create a robust and accurate speech-to-text model for Norwegian emergency calls. Unfortunately, this proved difficult since the fine-tuning had to happen in the same environment where the emergency calls were stored. Late in our research process, we gained access to a development environment equipped with the necessary packages. As a result, we have obtained some very

early results on how the original Whisper large-v2 model performs on real emergency calls. While it is too early to provide specific details on this, we can now confirm that it is feasible to run Whisper inside this environment. In addition, the preliminary examination of the resulting transcript also indicates promising potential. A natural next step would be to continue this evaluation using our fine-tuned model instead of the original. Then, to increase the model's accuracy and robustness, fine-tune the model using the real emergency calls that is available in this environment. *Seksjon for medisinsk dokumentasjon, Dokumentasjonsavdelingen ved Haukeland universitetssjukehus* has transcribed several hundred audio logs and continues to transcribe more.

Integrating spellchecking into the transcription process could improve the quality of the resulting transcript without needing to enhance the transcription model itself. This could be accomplished by running a spellchecker, for example, Hunspell [24], to rectify misspellings in the transcription output from Whisper. However, additional pre-processing steps, like censoring personal and sensitive information in the transcripts, would also need to be considered. We have tried using GPT-4 [36], requesting it to redact sensitive data from a Whisper-generated transcript. Preliminary results suggest that it effectively conceals names, social security numbers, and addresses, signifying that integrating a GPT model can be a promising approach.

Evaluating the effectiveness of semi-automatic transcription using our solution was only done by ourselves. To realistically tell how well the system would work in a real scenario would be to conduct a more extensive evaluation with professional audio transcribers. We can assume that these professionals are much faster writers than us and have more experience and better routines regarding this type of task. This might reduce the beneficial impact shown by our experiment on a first-time test without practicing with the system beforehand.

AI-generated suggestions might negatively and incorrectly impact a human's decisions. For example, sometimes there are words that, when you hear them, depending on what you believe is being said, are what you will hear. We have yet to test if our system could suggest the wrong word in such cases. It could be beneficial to do more research on that topic and see whether this would be a problem to the system's performance and, if so, to what extent.

While we mainly focused on AI-generated transcriptions using Whisper models, we also made it possible to tag the different speakers with the pyannote.audio diarization model. For now, our implementation of this model in the system seems to provide satisfactory results. The primary weakness related to speaker tagging in our system seems to be when the speaker speaks in short or single-word utterances. This weakness might depend on the accuracy of the models' output and also the algorithm we use to match the speaker with transcription. The extent to which available technology could enhance this aspect requires further investigation.

Bibliography

[1] *AMK SIMULATOR: Et komplett verktøy for kompetansebygging for medisinsk nødmeldetjeneste, basert på simuleringstrening og kunstig intelligens - Prosjektbanken*. Prosjektbanken - Forskningsrådet. URL: https://prosjektbanken.forskningsradet.no/project/FORISS/309676 (visited on 04/19/2023).

[2] *Artificial Intelligence in Innovation of Investigative Interviews Speech-To-Text and Text Analysis Using Machine Learning - Prosjektbanken*. Prosjektbanken - Forskningsrådet. URL: https://prosjektbanken.forskningsradet.no/project/FORISS/331893 (visited on 04/15/2023).

[3] Lasse Borgholt et al. "Do end-to-end speech recognition models care about context?" In: *arXiv preprint arXiv:2102.09928* (2021).

[4] Anni Burchfiel. *What is NLP (Natural Language Processing) Tokenization?* tokenex. May 16, 2022. URL: https://www.tokenex.com/blog/ab-what-is-nlp-natural-language-processing-tokenization/ (visited on 05/24/2023).

[5] Fredrik Byrsell et al. "Machine learning can support dispatchers to better and faster recognize out-of-hospital cardiac arrest during emergency calls: A retrospective study." In: *Resuscitation* 162 (2021), pp. 218–226.

[6] *Byte-Pair Encoding tokenization - Hugging Face Course*. URL: https://huggingface.co/course/chapter6/5 (visited on 03/01/2023).

[7] *Chakra UI - A simple, modular and accessible component library that gives you the building blocks you need to build your React applications.* Chakra UI: Simple, Modular and Accessible UI Components for your React Applications. URL: https://chakra-ui.com (visited on 02/27/2023).

[8] *Conformer-1: a robust speech recognition model*. News, Tutorials, AI Research. Mar. 15, 2023. URL: https://www.assemblyai.com/blog/conformer-1/ (visited on 04/13/2023).

[9] Mary L. Cummings. "Automation and Accountability in Decision Support System Interface Design." In: *The Journal of Technology Studies* 32 (2006), pp. 23–31.

[10] Ken H Davis, R Biddulph, and Stephen Balashek. "Automatic recognition of spoken digits." In: *The Journal of the Acoustical Society of America* 24.6 (1952), pp. 637–642.

[11] Eirin Nybø Ellensen. "Norwegian Index for Emergency Medical Assistance. Studies on the Use and Precision of the Emergency Medical Dispatch Guidelines in Norway." eng. Accepted: 2017-10-03T11:01:34Z ISBN: 9788230839829. Doctoral thesis. The University of Bergen, Aug. 2017.

URL: https://bora.uib.no/bora-xmlui/handle/1956/16736 (visited on 02/28/2023).

[12] *Fetch API - Web APIs — MDN*. Feb. 20, 2023. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API (visited on 03/23/2023).

[13] Behrooz Ghorbani et al. *Scaling Laws for Neural Machine Translation*. 2021. DOI: 10.48550/ARXIV.2109.07740. URL: https://arxiv.org/abs/2109.07740.

[14] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[15] *google/fleurs · Datasets at Hugging Face*. Feb. 1, 2023. URL: https://huggingface.co/datasets/google/fleurs (visited on 05/05/2023).

[16] Katherine Gordon et al. "The Wellington life flight helicopter emergency medical service (HemS): a retrospective audit against new Ministry of health criteria." In: *NZ Med J* 127.1402 (2014), pp. 30–43.

[17] Jan-Thorsten Gräsner et al. "EuReCa ONE 27 Nations, ONE Europe, ONE Registry: A prospective one month analysis of out-of-hospital cardiac arrest outcomes in 27 countries in Europe." In: *Resuscitation* 105 (2016), pp. 188–195.

[18] Jan-Thorsten Gräsner et al. "Survival after out-of-hospital cardiac arrest in Europe-Results of the EuReCa TWO study." In: *Resuscitation* 148 (2020), pp. 218–226.

[19] *Handbook - The TypeScript Handbook*. URL: https://www.typescriptlang.org/docs/handbook/intro.html (visited on 02/28/2023).

[20] Ministry of Health and Care Services. *Akuttmedisinforskriften*. no. Rundskriv. Publisher: regjeringen.no. Apr. 2015. URL: https://www.regjeringen.no/no/dokumenter/akuttmedisinforskriften/id2409330/ (visited on 02/28/2023).

[21] *Hjerneslag*. NHI.no. URL: https://nhi.no/sykdommer/hjernenervesystem/hjerneslag-og-blodninger/hjerneslag/ (visited on 05/23/2023).

[22] *Hjerneslag*. Helsebiblioteket. URL: https://www.helsebiblioteket.no/innhold/artikler/pasientinformasjon/hjerneslag (visited on 12/14/2022).

[23] *Hugging Face Hub documentation*. URL: https://huggingface.co/docs/hub/index (visited on 03/08/2023).

[24] *hunspell/hunspell: The most popular spellchecking library*. URL: https://github.com/hunspell/hunspell (visited on 05/30/2023).

[25] *Inference API - Hugging Face*. URL: https://huggingface.co/inference-api (visited on 03/08/2023).

[26] *Is Word Error Rate Useful?* News, Tutorials, AI Research. Sept. 9, 2021. URL: https://www.assemblyai.com/blog/word-error-rate/ (visited on 05/19/2023).

[27] Dawn O. Kleindorfer et al. "Designing a Message for Public Education Regarding Stroke." In: *Stroke* 38.10 (2007), pp. 2864–2868. DOI: 10.1161/STROKEAHA.107.484329. eprint: https://www.ahajournals.org/doi/pdf/10.1161/STROKEAHA.107.484329. URL: https://www.ahajournals.org/doi/abs/10.1161/STROKEAHA.107.484329.

[28] Rashmi U. Kothari et al. "Cincinnati Prehospital Stroke Scale: Reproducibility and Validity." In: *Annals of Emergency Medicine* 33.4 (1999), pp. 373–378. ISSN: 0196-0644. DOI: https://doi.org/10.1016/S0196-

0644(99)70299-4. URL: https://www.sciencedirect.com/science/article/pii/S0196064499702994.

[29] Audun Langhelle et al. "International EMS Systems: the Nordic countries." en. In: *Resuscitation* 61.1 (Apr. 2004), pp. 9–21. ISSN: 0300-9572. DOI: 10.1016/j.resuscitation.2003.12.008. URL: https://www.sciencedirect.com/science/article/pii/S030095720300460X (visited on 02/28/2023).

[30] Nicola Littlewood et al. "The UK helicopter ambulance tasking study." en. In: *Injury* 41.1 (Jan. 2010), pp. 27–29. ISSN: 0020-1383. DOI: 10.1016/j.injury.2009.04.002. URL: https://www.sciencedirect.com/science/article/pii/S002013830900223X (visited on 02/28/2023).

[31] *Machine Learning*. July 6, 2022. URL: https://www.ibm.com/cloud/learn/machine-learning (visited on 12/14/2022).

[32] Carl McQueen et al. "Does the use of dedicated dispatch criteria by Emergency Medical Services optimise appropriate allocation of advanced care resources in cases of high severity trauma? A systematic review." en. In: *Injury* 46.7 (July 2015), pp. 1197–1206. ISSN: 0020-1383. DOI: 10.1016/j.injury.2015.03.033. URL: https://www.sciencedirect.com/science/article/pii/S0020138315001709 (visited on 02/28/2023).

[33] Rick Merritt. *What Is a Transformer Model?* NVIDIA Blog. Mar. 25, 2022. URL: https://blogs.nvidia.com/blog/2022/03/25/what-is-a-transformer-model/ (visited on 05/24/2023).

[34] Ministry of Health and Care Services. *Sentrale elementer vedrørende organisering av AMK-sentralene*. no. Rapport. Publisher: regjeringen.no. Sept. 2016. URL: https://www.regjeringen.no/no/dokumenter/sentrale-elementer-vedrorende-organisering-av-amk-sentralene/id2511460/ (visited on 03/01/2023).

[35] NAKOS. *Norsk indeks for medisinsk nødhjelp*. https://www.helsedirektoratet.no/veiledere/norsk-indeks-for-medisinsk-nodhjelp/Norskindeksformedisinskndhjelp.pdf.

[36] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

[37] *Organisering av prehospitale tjenester*. Helsedirektoratet. URL: https://www.helsedirektoratet.no/retningslinjer/hjerneslag/behandlingskjeden-ved-hjerneslag/organisering-av-prehospitale-tjenester (visited on 05/23/2023).

[38] Ø. Østerås, G. Brattebø, and J.-K. Heltne. "Helicopter-based emergency medical services for a sparsely populated region: A study of 42,500 dispatches." en. In: *Acta Anaesthesiologica Scandinavica* 60.5 (2016). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/aas.12673, pp. 659–667. ISSN: 1399-6576. DOI: 10.1111/aas.12673. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/aas.12673 (visited on 02/28/2023).

[39] Barak Or PhD. *Learning Transfer Learning*. Medium. Jan. 6, 2023. URL: https://towardsdatascience.com/learning-transfer-learning-31b4b05f5a1a (visited on 05/24/2023).

[40] *pyannote/speaker-diarization · Hugging Face*. Jan. 17, 2023. URL: https://huggingface.co/pyannote/speaker-diarization (visited on 04/19/2023).

[41] *PyTorch*. URL: https://www.pytorch.org (visited on 05/31/2023).

[42] L. Rabiner and B. Juang. "An introduction to hidden Markov models." In: *IEEE ASSP Magazine* 3.1 (1986), pp. 4–16. DOI: 10.1109/MASSP.1986.1165342.

[43] Alec Radford et al. "Improving Language Understanding by Generative Pre-Training." In: *OpenAI* (2018).

[44] Alec Radford et al. *Robust Speech Recognition via Large-Scale Weak Supervision*. en. arXiv:2212.04356 [cs, eess]. Dec. 2022. URL: http://arxiv.org/abs/2212.04356 (visited on 02/22/2023).

[45] *React – A JavaScript library for building user interfaces*. URL: https://reactjs.org/ (visited on 02/17/2023).

[46] Leland Roberts. *Understanding the Mel Spectrogram*. Analytics Vidhya. Aug. 17, 2022. URL: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53 (visited on 05/22/2023).

[47] Mirjam Lisa Scholz et al. "Artificial intelligence in Emergency Medical Services dispatching: assessing the potential impact of an automatic speech recognition software on stroke detection taking the Capital Region of Denmark as case in point." In: *Scandinavian Journal of Trauma, Resuscitation and Emergency Medicine* 30.1 (2022), pp. 1–17.

[48] Rico Sennrich, Barry Haddow, and Alexandra Birch. *Neural Machine Translation of Rare Words with Subword Units*. en. arXiv:1508.07909 [cs]. June 2016. URL: http://arxiv.org/abs/1508.07909 (visited on 02/22/2023).

[49] *Slik kjenner du att eit hjerneslag*. Jan. 31, 2020. URL: https://www.helsenorge.no/sykdom/hjerneslag/slik-gjenkjenner-du-et-hjerneslag/ (visited on 10/12/2022).

[50] *Stroke - What Is a Stroke? — NHLBI, NIH*. Mar. 24, 2022. URL: https://www.nhlbi.nih.gov/health/stroke (visited on 05/31/2023).

[51] *Summary of the models*. URL: https://huggingface.co/docs/transformers/model_summary (visited on 03/01/2023).

[52] *Transformers*. URL: https://huggingface.co/docs/transformers/index (visited on 03/08/2023).

[53] *Use automatic captioning - YouTube Help*. URL: https://support.google.com/youtube/answer/6373554?hl=en#zippy=%5C%2Cautomatic-captions-on-long-form-videos-and-shorts (visited on 05/22/2023).

[54] Ashish Vaswani et al. *Attention Is All You Need*. Dec. 5, 2017. arXiv: 1706.03762[cs]. URL: http://arxiv.org/abs/1706.03762 (visited on 02/21/2023).

[55] Jeannine Volchko. *Prototyping Methodology: Steps on How to Use It Correctly*. URL: https://www.lumitex.com/blog/prototyping-methodology (visited on 05/20/2023).

[56] *Welcome to Flask — Flask Documentation (2.2.x)*. URL: https://flask.palletsprojects.com/en/2.2.x/ (visited on 03/22/2023).

[57] *What are Recurrent Neural Networks? — IBM*. URL: https://www.ibm.com/topics/recurrent-neural-networks (visited on 05/31/2023).

[58] *What Happens During a Stroke? — Premier Health*. URL: https://www.premierhealth.com/your-health/articles/women-wisdom-wellness-/what-happens-during-a-stroke- (visited on 05/31/2023).

[59] *What Is a Language Model? — deepset*. URL: https://www.deepset.ai/blog/what-is-a-language-model (visited on 05/24/2023).

[60] *What is a Spectrogram?* Pacific Northwest Seismic Network. URL: https://pnsn.org/spectrograms/what-is-a-spectrogram (visited on 05/31/2023).

[61] *What is Zero-Shot Classification? - Hugging Face*. Nov. 16, 2022. URL: https://huggingface.co/tasks/zero-shot-classification (visited on 05/31/2023).

[62] Erik Zakariassen, Elisabeth Holm Hansen, and Steinar Hunskaar. "Incidence of emergency contacts (red responses) to Norwegian emergency primary healthcare services in 2007 – a prospective observational study." In: *Scandinavian Journal of Trauma, Resuscitation and Emergency Medicine* 17.1 (July 2009), p. 30. ISSN: 1757-7241. DOI: 10.1186/1757-7241-17-30. URL: https://doi.org/10.1186/1757-7241-17-30 (visited on 02/28/2023).

Appendix A

Additional resources

A.1 GitHub project

The GitHub project for the web application can be found here:
https://github.com/AMK-MsC/amk-s2t

A.2 Video demonstration

A video demonstration of the project can be viewed here:
https://www.dropbox.com/s/1h8qdfbqvhyej81/demo-lyd.mp4?dl=0

Appendix B

Questions and answers from *AISMEC*

1. **Hvor sannsynlig er det at du ville tatt i bruk en slik løsning for å hjelpe med manuell transkribering av lydlogger?**

   (Ingen tekstsvar)

2. **Hvor fornøyd er du med designet av applikasjonen?**

   (Ingen tekstsvar)

3. **Hvor fornøyd er du med funksjonaliteten til applikasjonen?**

   (Ingen tekstsvar)

4. **Har du noen andre kommentarer eller tilbakemeldinger til applikasjonen? Skriv også gjerne en liten begrunnelse av svarene dine.**

   - **Svar:** Et slikt verktøy kan brukes til mye forskjellig og vil være til god hjelp og effektivisering av ulike arbeidsprosesser.
   - **Svar:** Har jo kun fått demo på skjerm, men det ser veldig lovende ut!
   - **Svar:** Ser ut til å være effektiv og nøyaktig. Enkelt dsign med svart skjerm, men trolig helt OK, så lenge brukergrensesnittet er lettfattelig og greit. Fint med boksen som kom opp med teksten, samt mulighet for download.

Appendix C

Questions and Answers from *Seksjon for medisinsk dokumentasjon*, HUS

1. **Bruken av denne teknologien vil føre til raskere transkribering.**
   - **Svar:** Absolutt, det vil lette oppgaven. Vi har talegjenkjenning i Helse Bergen, assistert talegjenkjenning, legene dikterer, sekretærene redigerer. Det virker å være samme løsning. Jobb med å bygge opp ordliste.
2. **Bruken av denne teknologien vil føre til mer korrekt transkribering.**
   - **Svar:** Ja, det kan være. Det er ulike personer som transkriberer hos Skrivetjenesten. Det varierer litt hvordan de oppfatter ting.
   - **Svar:** Valgt 3 fordi dersom ordet VIL er en påstand om at det blir og det er jeg usikker med tanke på den erfaringen Talegjenkjenning i foretaket ikke var så bra som man hadde trodd.
3. **Implementeringen av denne løsningen vil bidra til en bedre arbeidshverdag for dere.**
   - **Svar:** Ja, det kan være. Sekretærer liker å skrive. Tidsbruk: ja, absolutt.
   - **Svar:** Løsningen vil gjøre at vi bruker mindre tid på oppgaven og får utført flere oppgaver samt at det mest sannsynlig vil bli noe enklere for sekretærene å lytte på en forslagstekst enn å måtte skrive hele teksten selv. Vi er avhengig av arbeidsoppgaver, så med tanke på om vi får mindre arbeid, så er jo ikke det nødvendigvis en bedre arbeidshverdag.
4. **Selv om løsningen er utviklet for AMK, tror dere den kan være nyttig for andre arbeidsoppgaver? I så fall, hvilke?**

- **Svar:** Vanskelig å svare på, for de har allerede et talegjenkjenningssystem for pasientdokumentasjon. Vet ikke om det ene er bedre enn det andre. Det de har nå har vært i bruk veldig lenge. Først de siste årene at det har blitt mer tilnærmet det en antok at det kunne gjøre for 10 år siden. Ser at flere og flere bruker det. Men det har sine svakheter det eksisterende talegjenkjenningssystemet. Score 5 hvis det ikke fantes et talegjenkjenningssystem fra før, 3 siden det allerede finnes. Avhengig av om det er bedre eller ikke.

5. **Dersom dere skulle benyttet denne applikasjonen, har dere noen innspill eller forslag til forbedringer?**

    - **Svar:** Så enkelt som mulig; færrest mulig knapper. De som er der er veldig tydelige. Ikke blanding av norsk og engelsk; enten eller. Så enkelt som mulig; ikke blinkende lys etc.