A Work Project presented as part of the requirements for the Award of a Master's degree in

Business Analytics from the Nova School of Business and Economics.

DEVELOPMENT OF AN INNOVATIVE AND TRANSPARENT SYMPTOM CHECKER

Development of an Innovative and Transparent Symptom Checker
with a Focus on Automatic Speech Recognition

-

Florentin von Haugwitz (48174)

Work project carried out under the supervision of:

RONGJIAO JI

14/12/2022

Abstract

This paper presents the creation of a medical symptom checker with state-of-the-art machine and deep learning technologies. It examines the use and development of a speech to text model which is trained on medical datasets. The model is discussed, highlighting its advantages and disadvantages for the study. Moreover, the paper introduces a web application which provides a user-friendly interface, allowing users to interact with the models and showcase the results. Finally, the paper offers an outlook on the future use cases of the application and how it may improve healthcare outcomes.

Keywords: Deep learning, Machine Learning, Speech to text, Medical symptom checker, Streamlit

This work used infrastructure and resources funded by CMU Portugal program explorative research project MD2TRUST (CMU/TIC/0016/2021).

Table of Contents

1 Introduction

Unnecessary emergency department visits, causing longer waiting times, can have life-threatening consequences for other patients (Costa and Godinho 2016). Most patients visit the emergency department due to the fear of serious health problems or lack of knowledge about which department is most suitable (Honigman et al. 2013). The thesis aims to use Artificial

Intelligence (AI) to develop a prototype symptom checker application that patients and doctors can use for medical department and disease detection based on the patient's oral description of their symptoms. The benefit of this application is the reduction of the number of emergency department visits as patients are more informed which department to consult. It is to be proven whether such a symptom checker can be developed and whether this is a value-adding process for a user.

The problem patients are facing is the lack of knowledge of where to seek help for their health problems and the symptoms they are experiencing. Often, they are stuck in the phone queue for a consultation or sent from medical department to department. This not only costs the health system money but is a waste of valuable time for the patient. Our study presents a customer-friendly solution to facilitate and speed up the initial process of diagnosing a patient to solve this issue which could be later expanded to also work as an assisting tool for doctors.

We built an end-to-end solution with an easy-to-use interface, as illustrated in figure 1, starting with the audio description of a patient's symptoms, resulting in prediction probabilities for the three most relevant medical departments for the experienced symptoms. A distinctive feature of the product is that the patient has the comfort of expressing his symptoms in spoken language instead of choosing them from a given list of symptoms which adds to a good user experience. Furthermore, the entire process is very transparent, from the initial input to the final output, compared to other existing solutions. The patient will also be able to understand why the model made a prediction which increases its trustworthiness.

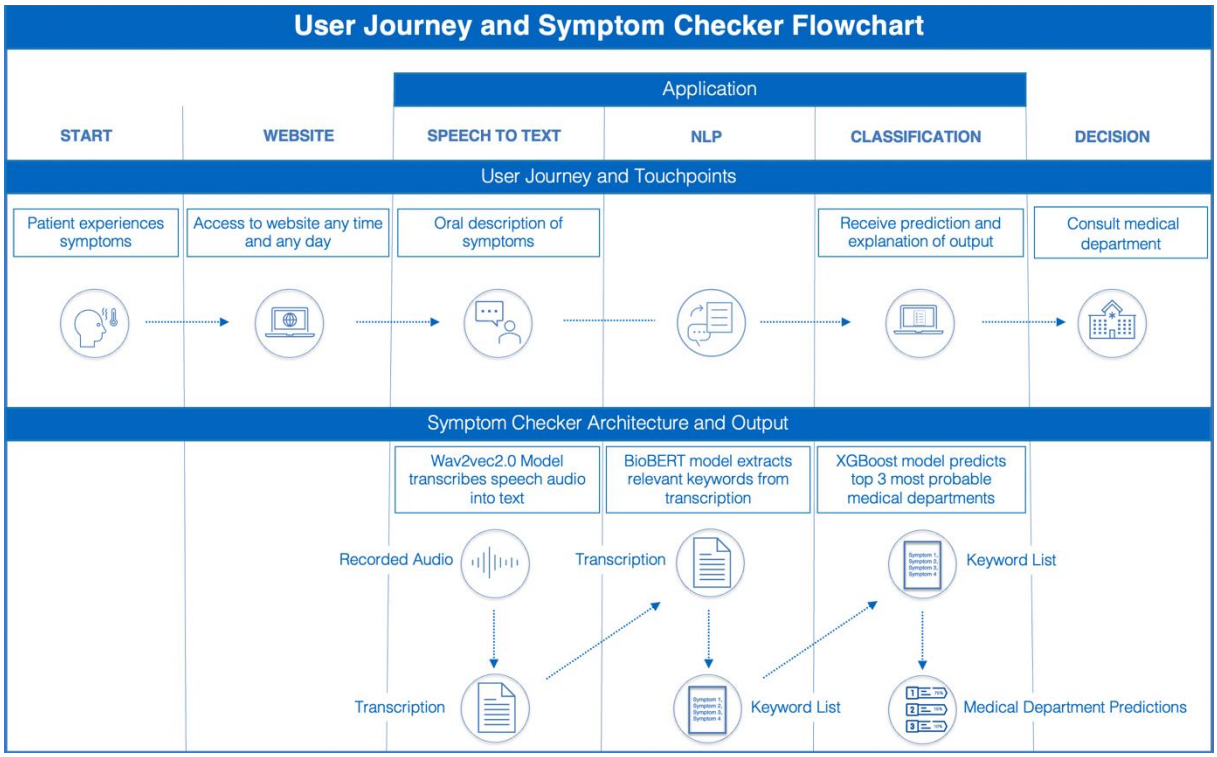

Figure 1: An illustration of the user journey and the architecture behind the web interface. The architecture is divided into three parts for which a final model was built, namely speech-to-text recognition, transcribing the oral patient's description into text, natural language processing which filters the transcription for keywords, and classification, which predicts medical departments based on the keywords. (Source: Figure created by authors).

Symptom checkers are ready to be accepted by users. A recent study examines the benefits and challenges of online symptom checkers and reveals that most healthcare professionals advocate the use of them. They allow patients to quickly connect with a healthcare professional and refer to self-treatment instructions, regardless of time and place. Moreover, medical professionals are pleased that the symptom checker provides more helpful information before meeting with the patient (Kujala et al. 2020). Hence, there is a market ready to be entered, also because patients increasingly turn to online resources to self-diagnose their symptoms before consulting a medical professional (Hochberg, Allon, and Yom-Tov 2020). If a symptom checker empowers patients through complementary information to make informed decisions, this contributes to increasing the efficiency of the healthcare system and advancing disease prediction.

Furthermore, through our work, we aim to draw attention to the use of AI and machine learning in health care since it has the power to revolutionize the healthcare system. Moreover, deep learning, which uses artificial neural networks to identify patterns and learn from large amounts

of data, is used in this study for the tasks of speech recognition and natural language processing.

Machine and deep learning represent a tremendous opportunity for healthcare since it has the potential to improve the accuracy of predictions of both health and disease. Algorithms can be trained to make predictions based on data from large populations and can learn to identify too difficult or more time-consuming patterns for humans to discern. Additionally, they are used for personalized predictions based on an individual's characteristics, like our symptom checker. Machine learning is already used in healthcare to improve disease risk predictions and develop personalized treatment plans (Son et al. 2020). Other categories of machine learning in health care involve patient engagement and adherence along the care continuum or administrative activities such as medical records management (Davenport and Kalakota 2019). It is important to emphasize that machine learning in health care must not be seen as a threat to the profession of physicians but rather as an opportunity to move routine tasks to automated technology, which will lead to a shift in the scope of physicians' tasks. It will allow them to dedicate more time to high-quality tasks like building meaningful relationships with their patients, making their human capabilities far more valuable than their diagnostic skills (Li, Kulasegaram, and Hodges 2019).

This paper is organized into four main sections. First, a literature review is conducted to highlight existing applications providing a background to this study and illustrating how our approach is innovative. Second, the accomplishments regarding our programming and web application are explained. Third, the methodology and results are described in detail. This section covers the different models, our academic approach, and our evaluation. Finally, the findings and implications are examined in the discussion part.

To conclude, having described the issues in a patient's diagnosis process and the current lack of autonomy and tools for a patient to make informed decisions, clearly, our symptom checker, developed with the use of AI, helps to solve those problems, and assists in the process of consulting the correct medical department.

2 Literature Review and Existing Applications

This section first lists and describes several existing symptom checkers which can be used as a comparison for this study, highlighting the special features of this prototype. Secondly, it looks at the existing literature on the datasets to take useful derivations for this study.

2.1 Existing Symptom Checker

Comparing this thesis' symptom checker to existing symptom checkers provides an understanding of the competitive landscape and what makes this product unique. A comparison can help to refine the product. Additionally, a comparison to existing products can aid in identifying market gaps and innovation opportunities and understanding what customers are looking for to develop a successful product. It is especially useful since it is a new market area, and there are not too many ready-made products on the market yet.

People can quickly find potential causes for their medical problems using symptom checkers. Users typically describe their symptoms or choose similar symptoms from a list and sometimes answer specific follow-up questions that help to focus on the most likely causes (Müller et al. 2022).

When searching online for publicly available symptom checkers, WebMD, one of the most popular online symptom checkers in the United States, appears at the top of the search page. According to a study published in the British Medical Journal, the WebMD symptom checker has a diagnostic accuracy of 35.5%, where the top three condition suggestions contain the primary diagnosis (Gilbert et al. 2020). The WebMD symptom checker only allows one to select from a given list of symptoms instead of describing them. It is, however, possible to add biodata and select symptoms by body location. As an intermediate step, one must rank the selected symptoms and can answer some optional questions to exclude certain conditions like pregnancy, for example. This symptom checker outputs a list of conditions with the tag of how likely the match is from strong to low that match the selected symptoms. One receives additional

information about condition details and treatment options. The tool lacks information about the most important symptoms to predict a diagnosis (“Symptom Checker with Body from WebMD - Check Your Medical Symptoms” 2022).

The online symptom checker of the Mayo Clinic also appears frequently in web searches. The Mayo Clinic is a not-for-profit group medical practice based in the United States and offers health information on its website (Putnam 2010). One can choose from a list of 45 symptoms and select related factors for the chosen symptom. Afterward, a list of diseases and conditions appears that match at least one of the factors selected. The diseases with the most matches are listed first. The user is provided with information on which factors relate to which disease and other factors related to the disease which have not been mentioned. If a disease is clicked on, a more detailed description is displayed, such as symptoms, advice on when to see a doctor, or risk factors. The Mayo Clinic also states that its symptom checker is not a diagnostic tool but a tool helping to narrow down the search (“Symptom Checker - Mayo Clinic” 2022).

According to studies from several medical journals referenced by the health search engine openmd.com the best free symptom checker regarding condition coverage, diagnostic accuracy, and appropriateness of urgent care advice is the smartphone app Ada which requires registration for usage (“5 Best Medical Symptom Checkers” 2022). Founded in 2011, the German start-up now has over 12 million users, and its medical intelligence achieves a diagnostic accuracy of 70.5% (Gilbert et al. 2020). A chatbot interface guides the user through a questionnaire about the symptoms they are experiencing. For every symptom, it is asked about its length, the exact location, the intensity, and potentially related symptoms. After gathering enough information, a report with potential conditions is created. There is a more detailed description for every condition, and the probability of a condition is given by indicating how many people out of 10 with the same symptoms also experience this condition (“Health. Powered by Ada.” 2022). According to an article by Andrea Park, Ada is a much more powerful version of the standard

WebMD symptom checker. It is continuously trained on a vast database of millions of clinical observations (Park 2021).

However, none of the symptom checkers mentioned disclosing their models or the data they use for their algorithms as we did. Other research dealing with symptom checkers often only focuses on a specific medical field and does not build a viable end-to-end product (Akrout et al. 2019; Hügle et al. 2020). A viable end-to-end product consists not only of a proposed and tested algorithm for a problem as presented by various papers but also of the implementation of the solution in an interface that a user can work with, as is the case with the symptom checker of our project.

2.2 Existing Work on the Datasets

Automation is becoming increasingly important in the health sector, and more and more research is being conducted in this area. The healthcare industry faces problems of cost, spending, and increasing waiting times. Therefore, there is a high motivation for hospitals and researchers to conduct research and analyze medical data to find efficiencies and cost reduction potentials (Nuthakki et al. 2019). There are different data sets in the medical field which are usable for free. This thesis focusses on three different ones. The MIMIC-III dataset and the MT Samples dataset, and the Medical Speech, Transcription, and Intent (MSTI) dataset. The datasets enjoy different scientific recognition. All datasets have already been used in significant studies or programming projects, but only the MIMIC-III dataset has relevant studies to compare with our task. Therefore, this analysis refers only to the MIMIC-III dataset.

The **MIMC-III** is one of the biggest publicly available datasets in the medical domain. It has been a valuable resource for researchers in a variety of fields and has enabled a wide range of studies that would not have been possible without its availability. Its use has provided insights into the risk factors, costs, and effectiveness of various medical interventions, as well as the performance of decision support systems. Most of the research follows a similar goal as ours,

trying to predict ICD codes based on the doctor's report. Our approach, in addition, finds a connection between the ICD-9 code to the department. Recent works use NLP algorithms to process the MIMIC-III database. For instance, a study by Huang et al. compares NLP methods to traditional classification methods with manually created rules to extract ICD-9 codes from the patient's text (Huang, Osorio, and Sy 2019).

A paper by Xianghao Zhan et al. compares five different NLP techniques which improve the quality of diagnostic codes by extracting structured diagnostic codes from unstructured notes concerning cardiovascular diseases (Zhan et al. 2021). Siddhartha Nuthakki et al. focus on mapping clinical notes to medical codes and predicting the final diagnosis for the top ten most occurring diagnoses. They use the deep learning method, ULMFiT, and achieve an accuracy of 80.5% for the top ten diagnoses (Nuthakki et al. 2019). The paper "multi-label natural language processing to identify diagnosis and procedure codes from MIMIC-III inpatient notes" by AK Bhavani Singh et al. tries to predict the top 10 and top 50 ICD codes with the BERT model that is also being used in our study. They achieve an accuracy score of 87% (Singh et al. 2020, 2).

In conclusion, this literature review has provided a comprehensive overview of the study of medical symptom checkers and the MIMIC-III dataset. A variety of research approaches and methods have been employed to investigate this topic. However, there is still a lack of a baseline for the community to reliably assess different algorithms on benchmark datasets. Also, it highlights the need to further explore the topic as there is room for further improvement for both the work on the MIMIC-III dataset and how to best develop a user-friendly symptom checker.

3 Data Sources

This section is concerned with the data finding and exploration of our project. The data finding is explained to help readers comprehend the research process and to demonstrate the validity and dependability of the data utilized in this study. Additionally, it provides readers with information about the selection of the data and the decision-making process. Secondly, data exploration is an

essential part of any machine learning project since it helps to determine the characteristics of the data and features that will be used for training. It helps to identify any potential issues with the data that need to be addressed before building a model. By understanding our data better, our model can be built more accurately and better suited to the problems of a symptom checker.

3.1 Data Finding

In any data science project, one of the biggest challenges is finding the right data which fits the problem that should be solved and is large enough to be statistically significant even after data cleaning and subset selection. Within the time frame of our master thesis, it was only possible to work with publicly available electronic healthcare data because negotiations with a private Portuguese hospital to support us with their electronic health records (EHR) were unsuccessful. Therefore, the data-finding process was managed by the team of this work project, and access to three different datasets was gained in the end. However, one must note that due to its nature, the availability of public healthcare data is extremely limited because of legal barriers like privacy regulations and the need to apply for access to specific databases (Humbert-Droz, Mukherjee, and Gevaert 2022). Furthermore, although healthcare data is very valuable for academic research and technology development, most of the existing datasets are relatively small (Maier 2020; van Panhuis et al. 2014). For our project, specific types of healthcare data are needed consisting of audio files, written text that includes the description of symptoms, and any assigned class labels such as medical departments or diseases. For the speech to text part of the project, labeled medical audio data is required. This means that every audio file needs to have a transcription. The data used for the project is called Medical Speech, Transcription, and Intent (MSTI) and is hosted on Kaggle.com (Mooney 2018). For the NLP and classification part, the work was started based on a scraped version from MTSamples.com, a public community platform website that contains sample medical transcription reports for different medical specialties initially created to teach clinical coders and transcriptionists (MTHelpLine 2022). At the beginning of the

project, we applied for access to the third version of the Multi-parameter Intelligent Monitoring in Intensive Care (MIMIC-III), an extensive clinical database including anonymized health-related data from more than 40,000 patients treated in the intensive care unit at Beth Israel Deaconess Medical Center between 2001 and 2012 (Johnson, Pollard, and Mark 2015). This process took over six weeks as the necessary credentials first had to be obtained, and a training course had to be completed before receiving access to the dataset. Although the time to work with the dataset was limited, it was decided to incorporate MIMIC-III into the project because the value-add is significant, as MIMIC-III is not only much larger than MT Samples but also contains codes from the international classification of diseases (ICD). Originally developed by the World Health Organization (WHO) to monitor mortality and morbidity, today, the ICD-codes are widely used by healthcare providers to accurately diagnose and document medical conditions, procedures, and treatment and to determine reimbursement and eligibility for health insurance coverage (Wu et al. 2019).

3.2 Data Exploration

Data exploration is a process of inspecting, cleansing, and transforming data with the goal of discovering hidden patterns, suggesting conclusions, and supporting decision-making. For our study, it is crucial to understand the existing dataset in order to know how to clean the data best, how to map the ICD codes to medical departments, how to optimize the resources for our training, and how to detect unwanted outliers.

3.2.1 Speech to Text

The MSTI dataset, which is used for fine-tuning the speech to text models, provides labeled audio snippets containing medical terminology (Mooney 2018). The exploration of the dataset is split into parts, as the first part focuses on the analysis of the metadata, while the second part focuses on the analysis of the audio data. The metadata is a CSV file which lists the paths to all files, including further meta information regarding the quality of the audio, the transcribed label,

the id of the speaker and transcriber, and the category of the audio. The second part depicts and generates key features of the audio files for analysis.

The metadata file consists of 6661 samples and 13 columns. Since the columns "background_noise_audible, "background_noise_audible:confidence", "audio_clipping", "audio_clipping:confidence", "quiet_speaker:confidence", "background_noise_audible", "quiet_speaker", "file_download", and "prompt" do not provide any added value. Since the categories from the "prompt" column, which contains medical categories for the phrase, are not transferable to the categories from the other dataset, this column is no longer needed. For further feature exploration, the number of words each phrase contains was added to the dataset. The dataset contains zero duplicate samples and zero empty values, meaning no samples are dropped before the exploration of the dataset.

The "phrase" column contains the transcription of the audio. On average, each phrase consists of 10.5 words, reaching from a minimum of two words to a maximum of thirty words (figure 14, appendix). A composition of the most frequent words, excluding general stop words, is shown in the word cloud beneath (figure 2).

Figure 2: Word cloud of most frequent words based on the "phrase" column. The larger the word is written, the more often the word occurs in the mentioned column. The simple choice of words indicates that the training data

rather consists of phrases by patients than professional doctors. (Source: Figure created by authors).

The "overall_quality_of_the_audio" column assesses each sample and marks them between 1 and 5. In this dataset, the quality ranges between 3 and 5 points, with an average value of 3.67. After reviewing samples with a quality of 3.5 and lower, it was decided that these samples remain in the dataset as the quality is sufficient. An overview of the quality is depicted in figure 13 in the appendix. In total, the audio files were spoken by 124 people and transcribed by 45 people. The number of audio files spoken per person ranges from 1 to 75, while the number of transcriptions ranges from 15 to 445. The boxplot below underscores the distribution of the speakers (figure 3). It becomes clear that most of the speakers have spoken most of the audio files, which is not ideal for training as training the data with a broad range of voices culminates in better results when using the model in applications.

Unfortunately, it is not possible to extract any more information about the speaker, such as gender, age, or if they are a native speaker. This would help to train and evaluate the quality of the predictions. In addition, it is not possible to retrieve information about the quality of the label. Exploring random samples proved that misspellings occur within the labels, which could lead to inaccurate results.

The data analysis is split between the analysis of the meta data and the audio files. To explore the audio files, which are stored in WAV format, the python package librosa was applied. This allows the extraction of individual characteristics from the audio files (McFee et al. 2015). Since the most important features regarding the audio data for this study are the audio length and the sample rate, both features were extracted from each audio sample. The sample rate defines how often per second a sound is sampled, thus the frequency of samples in a digital recording. The higher the sample rate is, the better quality of the sound sample (Weik 2012). The MSTI dataset provides audio samples with rates between 8000 and 192000 kHz. The exact sample rate distribution is represented in figure 16 in the appendix. As speech recognition models must be

trained with audio data of the same sample rate, the audio files of the incorrect format must be resampled according to the standard sample rate of the used speech to text model. This step will be further explained in section 5.2. The length of the audio files is crucial as audio files which exceed a certain limit are likely to cause memory issues during the training period. The histogram of the audio length files is depicted below (figure 4), showing that most of the file's audio duration ranges between 3.25 and 5.5 seconds, with outliers reaching up to 18.5 seconds.

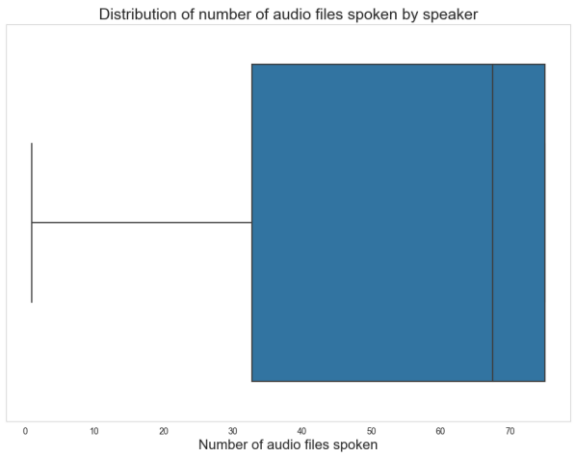

Figure 3: Boxplot of how audio files are spoken per speaker. Most speakers speak 65 times, although outliers exist, with less than five audio recordings. (Source: Figure created by authors).

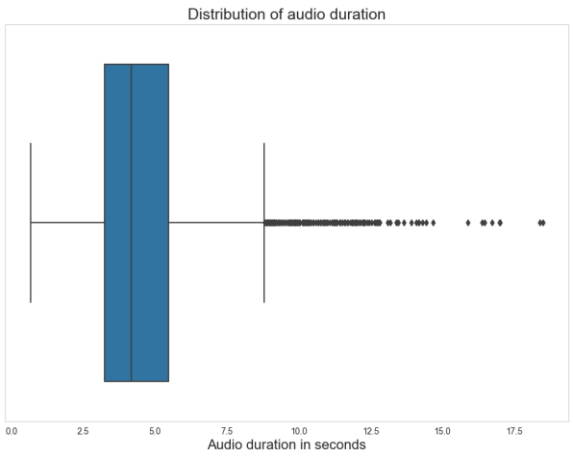

Figure 4: Boxplot of audio duration in seconds. Most of the audio files are less than 5 seconds long, while outliers have a duration of over 17 seconds. (Source: Figure created by authors).

A detailed exploration of a sample audio file is depicted in figures 17, 18, and 19 in the appendix, showing various audio features.

3.2.2 MT Samples

The MT Samples dataset is a smaller-scaled dataset that provides labeled medical transcription sample reports and examples of initially 4,999 samples. After removing missing and duplicate values, a sample size of 3,817 remains. The predictor variable is the "transcription" column which refers to the transcription of a physician's dictation of the patient's symptoms and medical condition in textual format. The text column has an average length of 3,054 and a maximum of 18,425 characters. For cleaning this column, nan values, digits, spaces, and stop words were removed.

Furthermore, the top 0.1% (110 words) of the most frequent words across all categories were removed, including “patient”, “stay”, or “hospital”. Those words are not specific to a particular class and do not contribute to prediction. To ensure that no critical words are removed, one must manually check that only non-informative words are removed. The final text was shortened to an average of 498 characters. The text length of the individual medical specialties varies between 400 to 550 characters. Since not all reports are well formatted, 33 lines had to be removed.

The categorical target variable is the “medical_specialty” column. There are initially 39 medical specialties. It becomes clear that this dataset is very unbalanced, and by far, the most frequently occurring category is “surgery”, with 1021 occurrences in contrast to other categories that appear less than ten times. Even the top ten most frequent classes have a great variability in number of occurrences as can be seen in figure 20 in the appendix. After evaluation, all categories that appear less than 100 times are removed, and eleven distinct categories that appear often enough to be used for training remain. Finally, 3100 samples are the final data size to work with.

When deep diving into the quality of the cleaned text for each medical specialty, we can detect distinctive words and symptoms appearing in a different frequency per specialty. The exemplary figures 5 and 6 for class Obstetrics/Gynecology and Neurology also show that the approach of using all the most frequent words will most likely not lead to great outcomes since most common words like “find”, “leave” or “normal” do not have a high medical meaning.

Figure 5: Wordcloud for the medical specialty Gynecology. The bigger the words, the more often they appear. (Source: Figure created by authors).

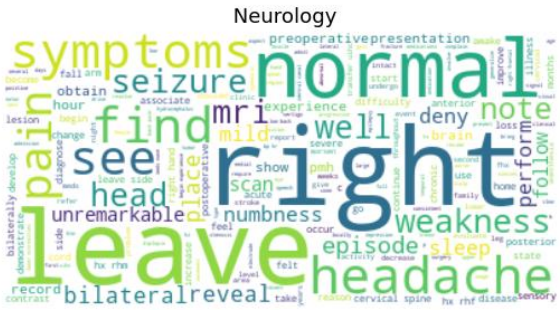

Figure 6: Wordcloud for the medical specialty Neurology. The words which appear most often are "leave", "right", and "normal". (Source: Figure created by authors).

3.2.3 MIMIC-III

The MIMIC-III dataset consists of multiple individual tables. The tables which are important for this project are the "diagnosis_icd" table and the "noteevents" table. The "diagnosis_icd" table includes the columns "ICD9_Code", "sequence_Id", and "subject_Id". Each patient has a unique "subject_Id". The "noteevents" table includes the "hadm_Id", the "subject_Id", the "category" column, as well as the "text" column. The predictor variable is the text column which is the doctor's report about a patient's stay. There are many unique categories which describe the type of report. After evaluating the quality of the different report types, it was decided to use only the discharge report category, analogous to the approach used by most other papers using MIMIC-III (Biseda et al. 2020).

Each patient stay can have multiple ICD-9 codes. Since the model cannot predict multiple diseases, the dataset diagnoses are reduced to take only the most severe ICD-9 codes, which are all samples where the "sequence number" column is one. In the final step, both tables are merged on the "subject_Id" as well as the "hadm_Id". Since this project focuses on patient usability, the ICD-9 codes have been matched to medical specialties. Therefore, this approach uses only the reports which are labeled with an ICD-9 code, hence reducing the number of rows to 58,967 rows.

To map the ICD codes to the medical departments, we have reduced each specific ICD-9

diagnose code to its ICD-9 chapter code by reducing the number to its first three digits. The 19 different ICD-9 chapter codes are mapped to 16 different medical specialties, as displayed in table 5 in the appendix. The mapping process was performed based on experience and academic references (Hansen et al. 2007; Scheurwegs et al. 2016; P. Wu et al. 2019).

The text column has an average length of 10,391 characters and a maximum length of 55,000 characters. The discharge summary text is structured with different subsections. They can include family history, social history, hospital stay, and current history. Since the KeyBERT model is not able to distinguish between family history, old illnesses, and current illnesses, the text is filtered only for current conditions, so the extracted keywords are only the current symptoms of the patient. In the following cleaning steps, NaN values, digits, spaces, and stop words were removed. In addition, the top 0.1% (200 words) of the most frequent words across all categories were removed, including “patient”, “doctor”, and “home”. To ensure that no important words are removed, it is necessary to manually check that only the uninformative words are withdrawn. The final text was shortened to 402 characters on average, whereas the average text lengths vary for the different categories between 800 to 300 characters. Since not all reports are well formatted, nine rows had to be dropped from further use.

From the **data exploration**, we find that the dataset includes 46,520 different patients, which belong to 41 unique ethnicities, whereas seven ethnicities make up 90% of the data. Furthermore, most of the patients (40%) are married. The gender distribution is 44.1% for females and 55.9% for males. Overall, each patient has, on average, 13 hospital admissions, each with an average of 11 “sequence_Id”, meaning eleven different ICD codes. When looking at the average length of a patient staying at the hospital, we can see in figure 7 that most of the patients stay between four to eight days. In figure 8, we can see the age distribution of the patients. The figure shows that most of the patients are between 60 and 80 years.

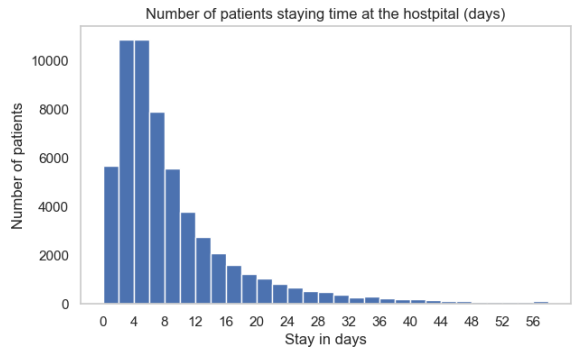

Figure 7: Number of patients staying time at the hospital in days. We can see that the data is right-skewed, meaning that some patients stay very long. Most of the patients stay around 4 to 8 days in the hospital. (Source: Figure created by authors).

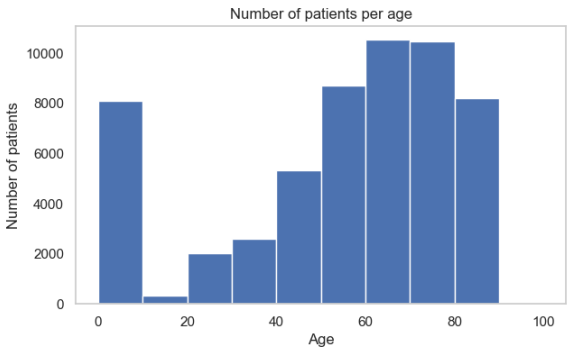

Figure 8: Number of patients per age group. We can see that most of the patients are between 60 to 80 years. Surprisingly there are very few people between 10-20 years. (Source: Figure created by authors).

When looking at the distribution of the reports in figure 9 shows the top 10 medical specialties by frequency. It shows that the medical specialties are not evenly distributed. The occurrence of each category varies from 31% for Cardiothoracic & vascular reports to Dermatology which occurs less than 0.03% in the data.

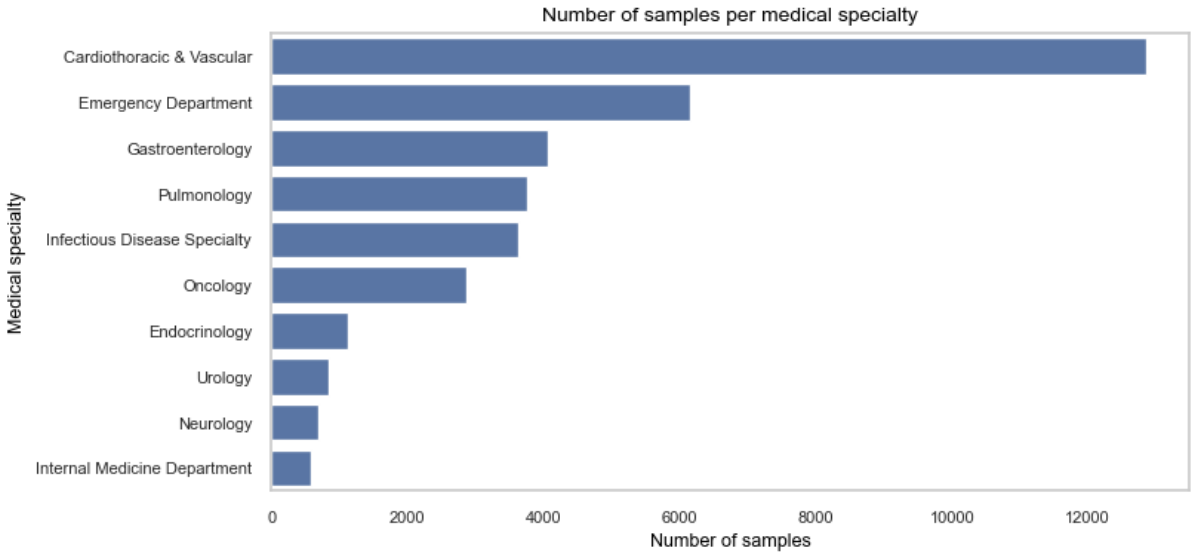

Figure 9: Number of samples per medical specialty. We can see that Cardiothoracic & Vascular is the most frequent. This chart shows only the top 10 most frequent categories. (Source: Figure created by authors).

4 Accomplishments

This section describes the tools and software solutions used to code and train the different models, and it explains how we allow users to interact with them and how we make them

accessible in a user-friendly fashion. In addition, a focus of our project was to create a structured code repository based on the latest coding principles, which provides transparency for researchers wanting to reference or reuse our methodology. Furthermore, an important aspect of the project is not only to present the feasibility of the project but also how our achievements are applicable to real-world use cases and how our work can be leveraged for future work. First, this section thoroughly explains our GitHub project structure, followed by our online web application, and ends with intended practical implications.

4.1 GitHub Structure

The structure of the repository is composed of multiple smaller individual parts, which combined provide a state-of-the-art framework and the possibility to reproduce the obtained results and to further develop the models based on the existing code repository. To ensure efficient collaboration between all group members, the distributed version control system git and the source code hosting platform GitHub were used. Git allows multiple participants to work on the same project while tracking changes to committed files ensuring the traceability of the modifications and the ownership of each code block. It is very effective and efficient in archiving the complete history of a project by storing only the changes between files (Perez-Riverol et al. 2016). In addition, git enables every member of a project to work on individual tasks by separating these tasks into so-called branches. A branch has the advantage that it does not interfere with other branches and therefore maintains an isolated structure until branches are merged into a single main branch. To merge a separate branch into the main branch, the functionality of a pull request was used, which ensures that at least one other project member has signed off on the logic and format of the code that will be merged.

GitHub, hosting over 10 million git repositories, is one of the most important software hosting platforms (Kalliamvakou et al. 2016). It provides a user-friendly interface for sharing and collaborating on coding projects. This includes providing functionalities to proof-read and verify

code from someone else, work on different work-streams simultaneously and host the code open-source. Hosting code open-source allows others to use the codebases as a reference, retrace individual steps and use it as a starting point for further similar projects (Jones 2013). For this project, GitHub was used to enforce a clean structure for the source code, have a transparent progression timeline, enable project members to oversee the work of others, and share code seamlessly. The code is open-source with an MIT License. This license allows users, among others, to copy, modify and distribute the code (Saltzer 2020).

To structure the repository in a transparent and understandable manner, multiple sub-folders were created, each containing a separate logical compartment. The most important directories are the "src", "model", "notebooks", and "data" folders. An overview of the structure is shown in figure 21 in the appendix. The "src" directory contains three main subfolders, one for speech to text, one for natural language processing, and one for classification. Within each sub-folder, the python modules follow the same principles as the structures are comparable. This results in having one constants file to store all non-changing variables, a utils file for all multiple used functions, a data cleaning file to prepare the data, training files for building the models, a predict file to test the models, and an evaluate file to compare the results of the respective models. In addition, all source code relevant to the final application is stored in this folder as well. All machine learning and artificial intelligence models are stored in the "models" folder, categorized in the subfolder of either speech-to-text, natural language processing, or classification.

To provide a reference for further users to understand the code and leverage the existing codebase, a documentation is created. The documentation is written with the python module sphinx, which generates reStructuredText as its markup language from docstrings in the python modules (Sphinx 2022). The documentation is hosted on GitHub pages, which is a documentation hosting service, making it accessible to the open-source community (https://tara-sophia.github.io/NLP_Masterthesis/). The documentation is automatically updated and

redeployed on every push to the master branch of the repository.

As an extension of our collaboration with git, we have used pre-commits, which force the author of modification to a file in the repository to maintain the same structure and conventions as the remaining files. In our project, pre-commits enforce a uniform coding format, as well as the existence of docstrings and unit tests for all functions. Tests are used to ensure that the code is working as intended. The tests are written in pytest, a testing framework for python, which asserts that a set of inputs matches the expected output. The tests are run on every commit to the repository, whereas the commit fails if one of the tests does not pass (Sottile 2017)

To ensure that the project can be run with the same packages installed as during the development phase of the project, a Conda environment was created. Conda is a free and open-source environment and package management system. Furthermore, it secures that every contributor to the project has the same version of a python package installed, streamlining the collaboration. The Conda environment file lists the required python packages that are needed for the project to function and simultaneously creates a virtual environment that is isolated from other python installations on the same machine (Anaconda 2016).

In addition to the repository being available under this link (https://github.com/Tara-Sophia/NLP_Masterthesis), it will also be uploaded under the NOVA SBE GitHub account (https://github.com/Data-Science-Knowledge-Center-Nova-SBE). This increases the visibility of the project as users and followers of the NOVA SBE repositories also become aware of the project.

In conclusion, using GitHub to host code is a great way to take advantage of the latest coding principles, incorporate third-party tools and ensure a high coding standard. GitHub provides a central repository for storing and managing code and offers a range of tools and features for reviewing and collaborating on code. This helped us to work more efficiently and effectively and will allow future users interested in our work to understand and enhance our work more

easily.

4.2 Streamlit Application

In addition to using GitHub, a user interface was set up to allow users to interact with the application easily and quickly. For this purpose, the python module Streamlit is used, which makes it easy to create and share web apps. Streamlit provides an API that allows to create a web application by writing a python script which automatically generates HTML, CSS, and JavaScript files, which are necessary to run the web application (https://streamlit.io/).

On the first page, the user of the application has the possibility to enter a voice-based input of symptoms. The speech input is processed by the speech-to-text model after the recording is completed and transcribed. Consequently, the text-based translation is passed to the Natural language processing model, which filters the most important words from the input and passes them to the classification model. To make it easier to use and clearer for the user, we decided against displaying the most important words. However, for the use of different target groups, e.g., doctors, the keywords can be displayed without any problems. On the next page, the user receives a list of the three most suitable categories, each with the percentage hit rate and a list of his input words that match this class. A "rerun" button returns the user to the initial page, accepting a new spoken input. Both pages of the web app are displayed in figures 22 and 23 in the appendix.

Since one of our goals was to allow the users to have easy access to the application at anytime, anywhere, and for free, we have hosted it under the following URL: https://huggingface.co/spaces/florentinhaugwitz/nlp_masterthesis_streamlit. Users can use this service therefore on their mobile devices without the need for any special requirements. This setup was chosen as it allows running the application on two virtual CPUs with a total of 16 GiB of RAM free of charge. Based on the size of the neural network models from the speech to text and natural language tasks, these hardware specifications are required for a smooth user

experience. Moreover, hosting the application on Hugging Face (https://huggingface.co/) permits an easy workflow for pushing updates to the application as it automatically pulls changes from a designated code repository.

To summarize, this project introduces a free and online accessible application which allows users to enter their symptoms linguistically and receive recommendations for the best-suited medical departments based on their input. This can help to reduce anxiety and provide individuals with more accurate information, providing advice and guidance to those in need. Finally, by giving consumers a tool to self-diagnose and manage common health concerns, which can assist in avoiding needless doctor visits and hospitalizations, our free application can help to lessen the strain on the healthcare system and health issues, which can help to prevent unnecessary doctor visits and hospitalizations.

4.3 Practical Implications

The realization of this project includes many use cases for different users. This includes the use of the application as well as the reuse and regeneration of the source code. Thus, the project serves a business benefit as well as a technical and academic benefit.

4.3.1 Use Cases

Regarding the practical implications of this project, there are multiple use cases with different stakeholders who benefit from the results of this project. The first includes the use of the application from a patient's perspective, the second highlights a physician's interaction with the application, and finally, the benefit for the entire healthcare system is described in more detail.

The first use case describes the use of a patient experiencing several symptoms. The main use of the application is to ascertain what type of disease may be present and what type of specialist should be consulted. In a few seconds and without answering several questions, as is often the case with existing solutions on the market, a patient can reflect on his symptom and feeling state. Here, the use of technical terms is not necessary. Within a very short time, the most appropriate

medical specialties are suggested to the patient in relation to their overall probability.

The second use case illustrates how a doctor could interact with the application. Since the application classifies the medical departments which are mapped on ICD-10 category codes, the application can just as well provide these codes to a professional. This would allow the professional to already have a suggestion of which disease or diseases to examine for. The same application and application pipeline can also be used for specifying the diseases more granular, which is emphasized in section 7.2.

The final use case tackles the problem of long waiting periods in the emergency department of hospitals. These long waiting periods cause a higher mortality rate and are caused, among other things, by an elaborate and timely referral process to other departments (Paling et al. 2020). The emergency department serves as a gateway to the hospital for many patients, even if they do not require immediate medical attention (Hogan and Bouknight 2002). An efficient solution to this problem is to incentivize the patients to inform themselves ahead of time to identify the correct specialty. Not only would the entire clinical system benefit from this, but also the individual would experience a better user experience as waiting times are shorter and the treatment is more specialized. The health system or the individual system could provide links to symptom checkers on their websites or natively include them to advertise them and thereby increase their popularity.

4.3.2 Reusability

The value of reusing open-source code is immense for other data scientists, developers, and researchers. It enables the exchange and collaboration of ideas, knowledge, and code across individuals and organizations, fostering a culture of innovation and creativity (Mergel 2015).

By allowing developers from all over the world to later collaborate on the project using open-source code on GitHub, it is possible to increase cooperation and transparency while improving the possibility of developing innovative solutions. Additionally, the open-source code offers a

platform for sharing knowledge, which can increase transparency and trust in the development process. Due to the frequent evaluation of open-source code by numerous developers, bugs and security flaws are frequently found and addressed rapidly, leading to more dependable products and higher code quality (Padhye, Mani, and Sinha 2014). Moreover, inaccurate results can be quickly identified and corrected, as the entire model consists of three transparent models each, and interim results can be examined at any stage. Additionally, visibility can be gained for this project which potentially leads to more users, contributors, and potential sponsors.

5 Speech to Text

In order to offer the user, the possibility to enter his symptoms via voice note, the first step of the project is to create a model that can convert spoken language into text. This step is included in the project as research has shown that speech input is three times faster than typing and more favorable with users (Ruan et al. 2016; Hauptmann and Rudnicky 1990). This adds to our overall goal of optimizing the user experience. Speech to text models belong to the group of automatic speech recognition (ASR) models, which are used in a variety of cutting-edge applications, such as voice search, voice assistants, and voice commands (Yu and Deng 2016). This part of the project introduces two state-of-the-art ASR models and explains their architecture, setup, and functionality. In addition, this section illustrates how both models are fine-tuned on the MSTI dataset and how they are evaluated and assessed based on their usefulness in this project. As the output of the ASR model is the input for the NLP model in the application, it is critical that the model has a high performance.

ASR models can be divided into two categories. The first category contains hybrid models, which are deep acoustic models. These models predict phonemes, the smallest unit of speech differentiating one-word element from another, from the audio input, which is transformed into a Mel Frequency Cepstral Coefficient (MFCC). The critical bandwidth of the human ear is known to vary with the frequency, which is the basis for MFCC and is prone to produce better

results in ASR (Muda, Begam, and Elamvazuthi 2010). With the use of pronunciation dictionaries, phonemes are combined with choosing the most likely letter (Këpuska, Elharati, and others 2015). The other approach is to use a deep neural network, which is able to predict the words directly from the audio input or MFCCs. These types of models require more data and resources for training (Maas et al. 2012).

For this project, Facebook's Wav2vec2.0 and Hubert models are used as base models, both belonging to the second category of ASR models. At first, the Wav2vec2.0 model is analyzed in detail, followed by the inspection of the HuBERT model. Afterward, the fine-tuning and optimization part for both models is depicted, resulting in the evaluation of the models and the implications for the rest of the project.

5.1 Wav2vec2.0 Model

The Wav2vec2.0 model is a continuation of the Wav2vec model, both developed by the Artificial department of Facebook (Baevski, Conneau, and Auli 2020). The Wav2vec model was released in 2019, being the first application of a self-supervised, convolutional, pre-trained model for speech recognition purposes. The new approach resulted in an increase in performance, surpassing the then-best model DeepSpeech2 from Baidu (Schneider et al. 2019). Due to the high resource overhead required to train the Wav2vec model and the associated optimization potential, Facebook released the improved Wav2vec2.0 model in late 2020. The Wa2vec2.0 model can be trained in less time, with less data, and achieve better results than its predecessor (Baevski et al. 2020).

As mentioned, the **Wav2vec2.0 architecture** is based on a convolutional neural network (CNN) model, which is a type of model mostly used for finding complex patterns in images (O'Shea and Nash 2015). CNNs belong to the higher-level class of neural networks, which consist of input, hidden, and output layers. Neural networks with multiple hidden layers are categorized as deep learning networks. All of these different layers may contain multiple nodes which can be

connected to nodes of the next layer (O'Shea and Nash 2015).

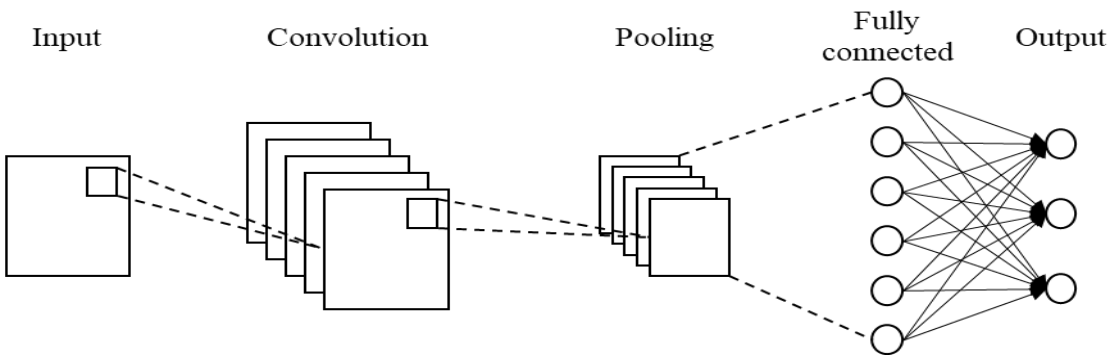

Figure 10: CNN architecture in individual steps. The workflow shows how the input is passed through a convolutional layer to extract meaningful features to a pooling layer. There the size is reduced for the fully connected layer to map the features to an output. (Source: Figure created by authors, based on O'Shea and Nash (O'Shea and Nash 2015)).

In the case of CNNs, these hidden layers mostly consist of a series of convolutional layers, followed by non-linearity layers, pooling layers, and then fully connected layers (Albawi, Mohammed, and Al-Zawi 2017). During the convolution part, a series of filters, which are generated during the training of the model, is applied to the input data to detect patterns (Krizhevsky, Sutskever, and Hinton 2017). To allow the identification of more complex patterns within the input data, the number of convolutional layers can be increased (Szegedy et al. 2015). In order to reduce the size of the output of the convolution, pooling layers are used. These pooling layers group semantically similar features, thereby improving the efficiency of the model and reducing noise within the input (LeCun, Bengio, and Hinton 2015). Succeeding a number of convolutional and pooling layers are a series of fully connected layers. Fully connected layers map the features from the convolutional layers to the desired output by ensuring that each input from the previous layer influences each node of the next layer (Krizhevsky, Sutskever, and Hinton 2017). A non-linearity layer is appended to the output of the fully connected and convolutional layers, increasing the performance and speed of the model (Krizhevsky, Sutskever, and Hinton 2017; LeCun, Bengio, and Hinton 2015).

The Wav2vec2.0 model builds on the CNN structure, with additional components added to

specialize it in the downstream task of converting speech to text. In detail, the model consists of three subcomponents, a feature encoder, a transformer module, and a quantization module, each containing multiple kinds of layers (Baevski et al. 2020). The feature encoder is responsible for extracting feature vectors from the raw audio by applying a CNN-based feature extractor to it. The transformer module is composed of multiple layers of self-attention, where each layer learns to capture different aspects of the audio signal. This is done by randomly masking input from the feature encoder and training the transformer to predict the correct output in regard to the quantization module. The quantization module is responsible for predicting the correct phonemes from an audio input. This module uses a Gumbel-softmax, a sampling technique for approximating categorical distributions, to predict which sound input relates to which letter (Jang, Gu, and Poole 2016). This task is performed by learning discrete speech units. These units are codewords sampled from codebooks, which are structures that contain all possible pairs of phenomes. Codewords are then concatenated to form the final speech unit (Jang, Gu, and Poole 2016; Baevski et al. 2020).

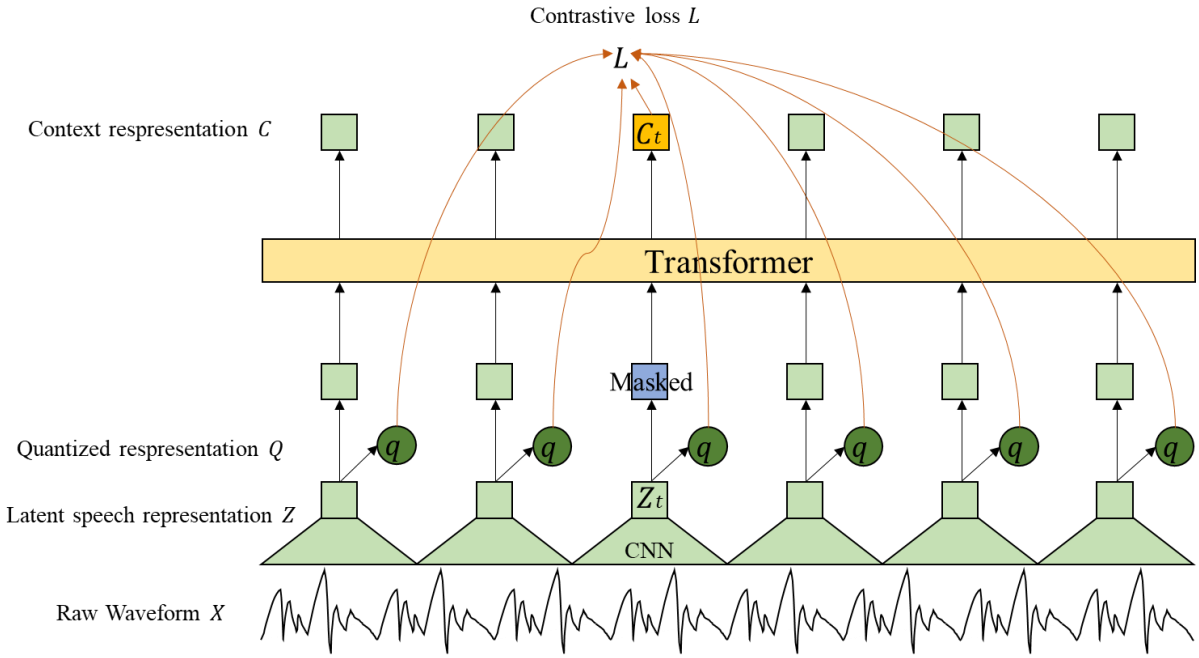

Figure 11: Wav2vec2.0 architecture setup. The latent speech representation layer extracts feature from the raw audio waveform. With the help of the quantization module and the transformer layer, the masked input parts are predicted. Based on the performance of the contrastive loss function, the weights within the model are adapted.

(Source: Figure created by authors, based on Baevski et al. (Baevski et al. 2020))

The model architecture for fine-tuning the model is based on Connectionist Temporal Classification (CTC). The CTC algorithm uses an input sequence to produce the most probable label sequence, which are words or phonemes for ASR. In order to maximize the likelihood of the predicted label sequence, the model's parameters are optimized in CTC to make a probabilistic prediction of the output (Hannun 2017).

The **pre-training** of the **Wav2vec2.0** model can be split into two different phases. The first phase of training the model consists of a self-supervised learning approach (Baevski et al. 2020). More particularly, the self-supervised approach is a contrastive task. Contrastive learning allows the model to detect patterns between input features. Although the data is not labeled, the model can find similarities and differences between the audio input (Tian et al. 2020). This enables training models without requiring only labeled data, which is in the area of automatic speech recognition highly infeasible and unpractical as this means that hours of speech data have to be transcribed manually. The data to pre-train the model was 960 hours of unlabeled recording data from the LibriSpeech corpus (Panayotov et al. 2015) and 53.2k hours of data from LibriVox (Kearns 2014). During this part of the training, the model detects and understands high-level patterns, such as extracting phenomes correctly. In addition, a diversity loss is added on top of the contrastive learning to the model during this phase. The diversity loss function is responsible for incentivizing the use of all codewords equally often and preventing the model to only choose from a fraction of codeword and codebook entries (Baevski et al. 2020).

The second training phase consists of fine-tuning the model. During this phase, the model is trained with supervised learning. The training data is from the LibriSpeech corpus but with a label. In this phase, the model trains to connect words or phonemes from the input with the associated label (Baevski, Conneau, and Auli 2020). Due to the unique setup of the training architecture, the model achieves a high performance on a cleaned dataset with only 10 minutes

of training (Baevski et al. 2020). Since the fine-tuning phase decides which specialization the model has, this phase is modified for the project. The pre-trained model is fine-tuned on the medical data. Thus, the model can transcribe medical data well.

5.2 HuBERT Model

In 2021 Facebook released another model, designed for automatic speech recognition tasks, named HuBERT (Hidden unit BERT). This model was designed to tackle three challenges of self-supervised models, namely that each speech input contains multiple sound units, there is no pre-training lexicon of input sound units, and sound units have varied lengths without explicit segmentation (Hsu, Bolte, et al. 2021). The HuBERT model addresses these issues by making use of a K-means clustering phase to produce aligned target labels for a BERT-like prediction loss function (Hsu, Tsai, et al. 2021). The BERT model's prediction loss function is a combination of the cross-entropy loss and the next sentence prediction loss. Cross-entropy is used to measure the difference between the predicted output and the true labels. The next sentence prediction loss is an additional component of the BERT model, which is used to measure the relationship between the two words or sentences (De Boer et al. 2005). The HuBERT model applies the prediction loss only over the masked regions, training the model combined on the acoustic and language input (Hsu, Bolte, et al. 2021).

The pre-training **HuBERT architecture** resembles the architecture of Wav2vec2.0. Similarly, HuBERT encodes the waveform audio inputs with a CNN. In addition, a BERT encoder, a projection layer, and a code embedding layer are used (Hsu, Bolte, et al. 2021). The CNN encoder is responsible for generating continuous speech features from the raw audio input. The speech features are afterward randomly masked and inserted in the BERT encoder to predict pre-determined cluster assignments. This allows the detection of long-range temporal relations in the input data. The HuBERT model can learn both auditory and language models from the continuous inputs because the prediction loss is calculated based solely on the masked regions

(Hsu, Bolte, et al. 2021; Anuchitanukul and Specia 2022).

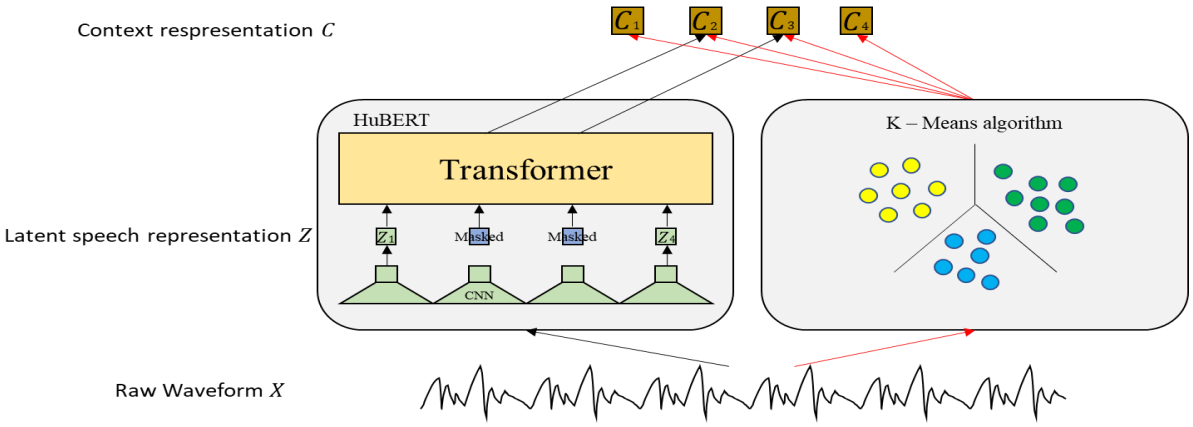

Figure 12: HuBERT architecture setup. The latent speech representation layer extracts feature from the raw audio waveform. At the same time, a K-Means algorithm groups similar audio features. Both models combined create the context representation. (Source: Figure created by authors, based on Hsu, Bolte, et al. (Hsu, Bolte, et al. 2021))

Although the pre-training setup of both models shares a similar architecture, both models can be differentiated from one another by the fine-tuning process. The three main differences are that HuBERT uses cross-entropy loss, builds targets via a separate clustering process, and reuses embeddings from the BERT encoder to improve predictions. While Wav2vec2.0 uses the combination of contrastive learning and diversity loss, HuBERT utilizes cross-entropy loss. Since a cross-entropy loss function lacks performance stability for a limited labeled dataset and has a poor generalization ability, these issues are addressed by other features of the HuBERT model and by pre-training the model with unlabeled data (Gunel et al. 2020).

As described in the section above, Wav2vec2.0 uses the Gumbel softmax function to learn how to predict its targets during training. HuBERT utilizes the k-means algorithm to create separate targets. This method increases generalization by increasing the number of targets and varying more often between them, decreases the speed to analyze similar input data, and effectively learns the semantic relationship between input sounds in an unsupervised manner (Wang, Boumadane, and Heba 2021; Baevski et al. 2020; Hsu, Bolte, et al. 2021).

The third differentiation between both models is the ability of the HuBERT model to reuse the

embeddings from the BERT encoder, while the Wav2vec2.0 model only uses the outputs of the CNN for the quantization module. Since the BERT encoder is able to capture contextual information and capture long-term dependencies in the text, it helps the model to better understand the semantic meaning of the text and to better distinguish between semantically similar words. Subsequently, this intermediate step leads to a generation of more accurate predictions (Wang, Boumadane, and Heba 2021; Baevski et al. 2020; Hsu, Tsai, et al. 2021).

The process of **pre-training** the **HuBERT** model is divided into two parts. The first step consists of extracting the hidden units from the raw audio input. This is done by fragmenting the audio into 25 milliseconds long segments, which are used as inputs for the K-means algorithm. The K-means algorithm divides the fragments into K clusters by analyzing two input features. The first input feature is the MFCC of each audio fragment. The coefficients provide an efficient indication of the rough cluster the audio fragment belongs to. Afterward, the model reuses the representations of the BERT encoder from the previous iteration to categorize the audio fragments more granular and precise. All audio frames associated with each recognized cluster will subsequently be given the unit label for the hidden unit that was created for it. After mapping each hidden unit to its appropriate embedding vector, predictions can be made in the second phase (Hsu, Bolte, et al. 2021).

The second step involves training the model with the aim of a masked language model. The HuBERT model masks half of the transformer encoder input features, training the model to predict the targets for the masked features. To receive the prediction logs, which track changes in the model over time to assess the performance, the cosine similarity is computed for the predicted masked feature and every hidden unit embedding from all feasible hidden units. The cross-entropy loss function penalizes the wrong predictions, thereby providing feedback to the model to predict more accurately for further predictions. To increase the performance when using noisy labels, the cross-entropy loss function is only applied to the masked features, as this

returns the best results (Hsu, Bolte, et al. 2021).

5.3 Model Fine-tuning

The following paragraphs describe how the data had to be adjusted for the training and how the model was set up to achieve optimal results. The fine-tuning process is a complex process, as many parameters must be optimized, often in combination with each other.

To train the model, the MSTI dataset is split into a train, evaluation, and test set with a size of 52.5%, 22.5%, and 25%. In addition, files that are shorter than 1 second and longer than 10 seconds are disregarded from the train set, as too-short audio files do not convey the same sentence structure, and too-long sentences result in memory issues during training. Subsequently, all audio files are resampled to 16kHz to match the preconditions for the Wav2vec2.0 and HuBERT models. Simultaneously, all letters are converted to lowercase, and the following special characters are removed from the various datasets “,”, “?”, “.”, “!”, “-”, “;” “:”, “[” and “]”. Since all these characters are silent in pronunciation, their removal ensures that the model is not evaluated on training a character which cannot be extracted from the acoustic input. The apostrophe is not removed, although it is silent, since it can alter the meaning of the word, e.g., the words “its” and “it’s” have different definitions. The unique remaining letters denote the categories to be predicted by the model for each letter. The 38 final categories include the letters of the English alphabet, in addition to the numbers 0,1,2,3,5 and 7, the space character, the characters “‘”, “\n”, and “`” and the additionally added special tokens “PAD” and “UNK”. In addition, each category gets a number assigned from 0 to 37. The “PAD” token is included as some input files require padding to compensate for their shortness, while the “UNK” token represents the unknown category, meaning the model cannot predict a known category. The table below shows the mapping of the keys to their numerical value.

Key	**\n**	**|**	**'**	**0**	**1**	**2**	**3**	**5**	**7**	**`**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Key	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**	**[UNK]**	**[PAD]**

Value	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37

Table 1: Speech to text vocabulary mapping file. The "key" row represents the alphanumeric values found in the transcription column the "value" row shows the value they are assigned to for the processor. (Source: Table created by authors).

For fine-tuning both models, the Hugging Face API was used. Hugging Face consists of multiple repositories and python packages which are designed for simplifying deep learning down-stream tasks. Their transformers library is used to load the models and perform the necessary steps to train and evaluate the model. Both models use the same processor to encode and decode the audio data and references. The processor for this task is downloaded from the Hugging Face library and applied to the raw audio files and their corresponding labels (Wolf et al. 2020). Every character in the label is transformed into its corresponding number. An example of this can be seen in table 2 below for the sentence, “my joints feel swollen”.

Original word	**m**	**y**	**|**	**j**	**o**	**i**	**n**	**t**	**s**	**|**	**f**	**e**	**e**	**l**	**|**	**s**	**w**	**o**	**l**	**l**	**e**	**n**
Processed word	22	34	1	19	24	18	23	29	28	1	15	14	14	21	1	28	32	24	21	21	14	23

Table 2: Example of a processed word. Each letter in the sentence “my joints feel swollen” is represented by its number, as seen in the mapping table (Table 1). (Source: Table created by authors).

Since the length of the input samples for the model can vary, it is necessary to pad all samples to the longest one of the respective batches. This is achieved by using a custom padding class, which finds the longest input sample regarding audio and label length. The remaining samples of the batch are padded from the right side until the length matches both features. The “UNK” token is used for this case.

The pre-trained Wav2vec2.0 base model is downloaded from Facebook’s Hugging Face page (https://huggingface.co/facebook/wav2vec2-base). The base model is used for this study as the large model produced memory errors during training. The pre-trained HuBERT model was also downloaded from Facebook’s Hugging Face page (https://huggingface.co/facebook/hubert-large-ll60k). In this case, the large model did not produce any errors during training. Hugging Face provides an API to use the open-source model to fine-tune the pre-trained model further.

This API includes a trainer class, which accepts the necessary input parameter to fine-tune the model.

The trainer trains the model on the train dataset and evaluates the evaluation dataset with the provided evaluation function. To monitor the performance of the train dataset, the evaluation metrics (see section 5.4 for more detail) are computed after each epoch on the validation dataset. The metrics are calculated on the validation dataset to provide unbiased results in case the model is overfitted by the train data.

For the fine-tuning of both models, the training strategy, floating point 16, and gradient checkpointing hyperparameters were set as constants. The training strategy is set to "epoch". The floating point 16 boolean is set to true to reduce the memory size. This means that the precision during training is reduced from 32-bit floating points to 16-bit floating points. Setting the gradient checkpointing boolean to true also decreases the memory usage but increases the computation time. In addition, the feature encoder, which contains the CNNs of the model, is frozen. This is necessary as the pre-trained models already have the capability to extract meaningful features from the raw audio, and this step does not have to be fine-tuned (Baevski et al. 2020).

Since both models save checkpoints of the current state of the model after each epoch, the last checkpoint was utilized for restarting the training. This allows restarting the training process where it ended and not having to compute all prior epochs as well. This is especially useful when training cannot finish due to memory errors.

In addition to the hyperparameter values, a custom earlystopping callback is included in the training process. This is useful when a model improves its performance based on the train dataset, but the performance on the validation dataset stagnates or decreases. This is a typical indication of overfitting, which can be prevented with earlystopping. The earlystopping callback recognizes that the validation metrics do not improve, thus stopping the training process. For the

speech-to-text models, the WER and CER are constant at 0 for the first epochs, as nothing is predicted correctly. To only start the earlystopping callback when the WER and CER have improved, the earlystopping callback was modified to only be called after 15 epochs (Prechelt 1998).

Both models were trained on an A4000 machine, which provides 16 GB of GPU, and 8 vCPUs for training. The average GPU occupancy rate was over 95%.

To **optimize** the models, the **hyperparameters** of both models are individually adjusted. For both models, the following hyperparameters were tuned and based on academic references (Mohamed and Aly 2021; Peng and Harwath 2022; Wang, Boumadane, and Heba 2021; Z.-C. Chen et al. 2022; Baevski et al. 2020; Hsu, Bolte, et al. 2021) and a trial-and-error approach. The following values are set for each model respectively.

Hyperparameter	Wav2vec2.0 value	HuBERT value	Explanation
Batch size	32	16	This number expresses how many samples are propagated through the network at a time. A small batch size requires less memory, and the training is faster. A large batch size increases the accuracy of the gradient (Smith et al. 2017).
Gradient accumulation steps	2	2	This number defines how often the gradients and loss are calculated without the model parameters being updated. Afterward, the parameters are updated on the cumulated gradient of consecutive batches. The higher the number, the less likely it is to run out of memory, but also that the performance decreases (Hermans, Spanakis, and Möckel 2017).
Number of training epochs	30	25	By defining the number of training epochs, the length of the training is subsequently set (Osawa et al. 2018).
Learning rate	1e-4	5e-4	This value controls how large of a step in the direction of the negative gradient the model's parameters must be updated in response to the outcome of the loss function each time. A small learning rate takes a long time to converge, while a too-high learning rate overjumps minima (Zeiler 2012).
Weight decay	0.005	0.005	By choosing a higher value, it can suppress some of the impacts of static noise on the targets and any irrelevant components of the weight vector, thereby

Warmup steps	1000	500	generalizing the model (Krogh and Hertz 1991). The number of steps at the beginning of the training at which the learning rate is much than the one which is set (Izsak, Berchansky, and Levy 2021).

Table 3: Hyperparameters for speech to text models. Each hyperparameter is assigned the optimal value for fine-tuning the respective model and a brief explanation. (Source: Table created by authors).

5.4 Results and Evaluation

This section covers the evaluation of both models on a set of different metrics. The evaluation is run on the test dataset to prevent the results from having a bias. The evaluation of the metrics leads to the selection of the model that will be used in the application.

A total of four **metrics** are used **to evaluate** the two models: Word Error Rate (WER), Character Error Rate (CER), prediction speed, and model size. The WER and CER are very common evaluation metrics in the speech to text area as they determine how many of the words or characters are correctly transcribed in comparison to the reference value. Since both metrics are error rates, a lower value is more favorable as this means that fewer words or characters are predicted incorrectly, with 0 being the perfect score. The formula for the calculation of the WER is depicted below.

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

Equation 1: Word Error Rate (WER) formula

The S refers to the number of substitutions, the D stands for the number of deletions, the I represents the number of insertions, and the N denotes the number of words in the reference. The formula for the CER is, on a high level, identical to the WER formula, although they carry different meanings.

$$CER = \frac{S + D + I}{N}$$

Equation 2: Character Error Rate (CER) formula

For the CER, the letters are counted on a character basis, and the N stands for the number of

characters in the reference. For both the WER and CER, the values can be above the range of 0 to 1 if the prediction contains more words or letters respectively than the reference. For speech-to-text use cases, the CER value is often lower than the WER due to the facts of homophones. This implies that the model predicts words that sound similar to the reference but are spelled differently, although often in just a few letters. Since the WER would mark a word as an S which only has one incorrect letter, but the CER would still account that the rest of the word is predicted correctly, the CER value has a better score (Morris, Maier, and Green 2004). In table 6 in the appendix, an example of predictions and references with their CER and WER is provided to illustrate the different computations.

The prediction speed is an important metric for this project as this implies how long a user of the application must wait for their input to be analyzed and an output generated. This is important since users are accustomed to receiving quick responses to their queries online and otherwise lose interest in the application. The time required for each model to compute the same 1,665 predictions can be seen in table 4.

The size of the model is important for this project because the application is hosted on a free platform, which only provides a certain amount of memory and has limited computing power (see section 7.1.). A larger model ties up more resources which can cause a worse user experience as the application starts to lag and the application has a higher failure rate.

All four evaluation metrics are important for assessing the models, although the results of each metric are not weighted the same. The WER is the most important value, followed by the CER and the speed of the model. The accuracy of the predictions has a crucial impact on the performance of the natural language processing and classification model, which is why the outcome of the transcription is of utter importance. A better WER score can relativize a slightly longer waiting time for the user.

The **assessment of both models** is performed for each evaluation metric and produces the

following results.

	WER	CER	Speed (1665 samples)	Size
Wav2vec2.0	11.0%	4.7%	490 seconds	0.38 GB
HuBERT	10.6%	5.3%	630 seconds	1.26 GB

Table 4: Performance evaluation of both ASR models. Both models perform similarly well for the error rates, but the Wav2vec2.0 model outperforms the HuBERT model in terms of speed and size. (Source: Table created by authors).

The above table shows that the HuBERT model is performing better regarding the WER but is outperformed in all other categories by the Wav2vec2.0 model. Especially interesting is the fact that the Wav2vec2.0 model has a lower CER value but a higher WER value than the HuBERT model. This means that the Wav2vec2.0 model misspells words more frequently than the other model. In addition, the fact that the Wav2vec2.0 is the faster and smaller of the two are two advantages for the implementation based on the resource limitations of the project.

As both models are performing similarly in terms of WER, both models were tested in the Streamlit application. This resulted in favoring the Wav2vec2.0 model as this one provided a smoother interaction. The predictions were received noticeably faster, and the application did not encounter any disturbances. The HuBERT model, apart from occasionally running very well, often had the issue of running out of memory. This resulted in having to restart the application server in the backend. Since this does not provide any benefit to a user, even as the WER score is better, the Wav2vec2.0 model was chosen for implementation in the final application pipeline.

In conclusion, this section introduced an approach for fine-tuning state-of-the-art automatic speech recognition models on medical speech data. The models chosen, Wav2vec2.0 and HuBERT, both performed very well on the little amount of speech data which was available for this study and scored Word Error Rates of 11% and 10.6%. This underlines the versatility of both models and proves the high performance of the approach used in this study. Finally, the Wav2vec2.0 model was chosen for the implementation in the web application as it has a smaller size, predicts faster, and has a similar Word Error Rate and Character Error Rate as the HuBERT

model.

6 Summary

The most important contributions of the speech to text model are highlighted below. This section described a method for optimizing cutting-edge automatic speech recognition models on data from medical speech. The two models, Wav2vec2.0 and HuBERT, scored impressive Word Error Rates of 11 and 10.6%, respectively, on the scant quantity of voice data that was available for this study. Ultimately, the Wav2vec2.0 model was selected for the implementation in Streamlit since it performs more smoothly, predicts more quickly, and the error rates for both models are comparably good.

7 Discussion

The outcomes of this paper have provided insight into the development and implementation of a symptom checker. Strong results are achieved, which leads to the conclusion that this prototype can be turned into a viable product. Nevertheless, the results should be interpreted with caution due to the limitations of the present study that will be presented in this chapter. A reflection on the research process is provided, and several limitations and potential consequences of the study's design are discussed. The section ends with several recommendations and an outlook for future research.

7.1 Limitations

During this study, we faced several limitations. The first is that the performance of the model is highly dependent on the input data. Since the NLP model has been trained using different medical records, it has learned specific complex words that are common to doctors' speech features very well but are not found in everyday speech features. Consequently, the model works well when given specific medical nuances, while it is not so familiar with the more generalized client speech, which has different linguistic features. Hence, the model is not always able to understand the client's language. A major improvement could be to have the model trained on

patient data as well.

Second, the MIMIC-III dataset contains only intensive care records from a single healthcare facility, but since there are significant differences in care and practices, additional records from different facilities could improve the generalizability of the model. In addition, the model has only a limited number of diseases it has seen and can classify, which may result in poor performance for unseen data and diseases. Furthermore, clinical guidelines and practice constantly change. When new guidelines with new procedures and diagnoses are implemented, the algorithm's performance could be affected, and thus, the algorithm should be recalibrated.

Regarding the technical limitations, we faced a lack of computing power for the NLP model. Since the models were running on our laptop or on a virtual server called Paperspace (Paperspace 2022), which was limited to six hours, the runtime was an additional technical constraint. Some models required too many hours to tune the hyperparameters, which could not be done because the required runtime and storage capacity was not available. Since natural language processing models typically require a significant amount of data and computational power to be trained, a laptop may not be able to handle larger-scale models. In addition, laptops typically do not have powerful GPUs or TPUs, which can significantly affect the accuracy and speed of a natural language processing model.

Another limitation was our domain knowledge. Without medical domain knowledge, it is challenging to understand medical concepts and theories hence making it harder to analyze the extracted keywords and the predicted classifications.

7.2 Outlook

Our models can be used as a solid baseline for future work. As stated in the limitations section, there is still a need for improved data quality for further machine learning processes. Extensive data collection, including high-quality electronic health records from multiple hospitals, is necessary to increase the performance and generalizability of the models. Additionally, with the

increased use of our application, the data entered will be returned to the model for training and can enhance the performance even more.

Moreover, if a person with medical domain knowledge was on the team, then the insights of the post-hoc analysis of the classification could also be used to remove other non-relevant features during the cleaning process and improve the model results even more to increase its meaningfulness.

Currently, predictions are made on the highest level of medical specialties. In the future, to provide more use cases, diagnoses can be predicted using a more granular classification of ICD codes. Moreover, with the integration of specific user info like age, gender, or historical medical records into the symptom checker interface, a better contextual understanding of individual symptoms can be provided. Furthermore, other areas in the field of disease detection could be explored, such as the potential to combine the information from clinical notes with other data sources, including image data and laboratory test data, to predict diagnoses.

8 Conclusion

Our study provides valuable insights into the development of a symptom checker. The prototype of a symptom checker that was built has shown to be effective in predicting medical departments based on a user's oral description of their symptoms. Our results exceeded expectations, indicating a remarkable level of achievement in comparison to existing work. We believe that the proposed use cases of this symptom checker can help to ease the burden on healthcare professionals and the healthcare system while also providing patients with more information to better make informed decisions on which medical department to consult. In conclusion, this symptom checker is an important and invaluable tool in the pursuit of understanding and managing one's health.

Bibliography

"5 Best Medical Symptom Checkers." 2022. OpenMD.Com. November 22, 2022. https://openmd.com/directory/symptoms.

Akrout, Mohamed, Amir-massoud Farahmand, Tory Jarmain, and Latif Abid. 2019. "Improving Skin Condition Classification with a Visual Symptom Checker Trained Using Reinforcement Learning." In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, edited by Dinggang Shen, Tianming Liu, Terry M. Peters, Lawrence H. Staib, Caroline Essert, Sean Zhou, Pew-Thian Yap, and Ali Khan, 549–57. Lecture Notes in Computer Science. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-32251-9_60.

Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. 2017. "Understanding of a Convolutional Neural Network." In *2017 International Conference on Engineering and Technology (ICET)*, 1–6. https://doi.org/10.1109/ICEngTechnol.2017.8308186.

Anaconda. 2016. "Anaconda Software Distribution." Anaconda Documentation. 2016. https://anaconda.com/.

Anuchitanukul, Atijit, and Lucia Specia. 2022. "Burst2Vec: An Adversarial Multi-Task Approach for Predicting Emotion, Age, and Origin from Vocal Bursts." *ArXiv Preprint ArXiv:2206.12469*.

Baevski, Alexei, Alexis Conneau, and Michael Auli. 2020. "Wav2vec 2.0: Learning the structure of speech from raw audio." September 24, 2020. https://ai.facebook.com/blog/wav2vec-20-learning-the-structure-of-speech-from-raw-audio/.

Baevski, Alexei, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. "Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations." arXiv. https://doi.org/10.48550/arXiv.2006.11477.

Biseda, Brent, Gaurav Desai, Haifeng Lin, and Anish Philip. 2020. "Prediction of ICD Codes with Clinical BERT Embeddings and Text Augmentation with Label Balancing Using MIMIC-III." arXiv. http://arxiv.org/abs/2008.10492.

Costa, Luana Bonome Message, and Moacir Godinho. 2016. "Lean Healthcare: Review, Classification and Analysis of Literature." *Production Planning & Control* 27 (10): 823–36.

Davenport, Thomas, and Ravi Kalakota. 2019. "The Potential for Artificial Intelligence in Healthcare." *Future Healthcare Journal* 6 (2): 94–98. https://doi.org/10.7861/futurehosp.6-2-94.

De Boer, Pieter-Tjerk, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. 2005. "A Tutorial on the Cross-Entropy Method." *Annals of Operations Research* 134 (1): 19–67.

Gilbert, Stephen, Alicia Mehl, Adel Baluch, Caoimhe Cawley, Jean Challiner, Hamish Fraser, Elizabeth Millen, et al. 2020. "How Accurate Are Digital Symptom Assessment Apps for Suggesting Conditions and Urgency Advice? A Clinical Vignettes Comparison to GPs." *BMJ Open* 10 (12): e040269. https://doi.org/10.1136/bmjopen-2020-040269.

Gunel, Beliz, Jingfei Du, Alexis Conneau, and Ves Stoyanov. 2020. "Supervised Contrastive Learning for Pre-Trained Language Model Fine-Tuning." *ArXiv Preprint ArXiv:2011.01403*.

Hauptmann, Alexander G, and Alexander Rudnicky. 1990. "A Comparison of Speech and Typed Input." In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*.

"Health. Powered by Ada." 2022. Ada. November 22, 2022. https://ada.com/.

Hochberg, Irit, Raviv Allon, and Elad Yom-Tov. 2020. "Assessment of the Frequency of Online Searches for Symptoms Before Diagnosis: Analysis of Archival Data." *Journal of Medical Internet Research* 22 (3): e15065. https://doi.org/10.2196/15065.

Hogan, Andrew J, and Reynard R Bouknight. 2002. "The Emergency Department as a Gateway for Hospitalization: Evidence from the National Medical Expenditure Survey." *Journal of Health Care for the Poor and Underserved* 13 (3): 288–97.

Honigman, Leah S, Jennifer L Wiler, Sean Rooks, and Adit A Ginde. 2013. "National Study of Non-Urgent Emergency Department Visits and Associated Resource Utilization." *Western Journal of Emergency Medicine* 14 (6): 609.

Hsu, Wei-Ning, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. "Hubert: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29: 3451–60.

Hsu, Wei-Ning, Yao-Hung Hubert Tsai, Benjamin Bolte, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. "HuBERT: How Much Can a Bad Teacher Benefit ASR Pre-Training?" In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6533–37. IEEE.

Huang, Jinmiao, Cesar Osorio, and Luke Wicent Sy. 2019. "An Empirical Evaluation of Deep Learning for ICD-9 Code Assignment Using MIMIC-III Clinical Notes." *Computer Methods and Programs in Biomedicine* 177 (August): 141–53. https://doi.org/10.1016/j.cmpb.2019.05.024.

Hügle, Maria, Patrick Omoumi, Jacob M van Laar, Joschka Boedecker, and Thomas Hügle. 2020. "Applied Machine Learning and Artificial Intelligence in Rheumatology." *Rheumatology Advances in Practice* 4 (1): rkaa005. https://doi.org/10.1093/rap/rkaa005.

Humbert-Droz, Marie, Pritam Mukherjee, and Olivier Gevaert. 2022. "Strategies to Address the Lack of Labeled Data for Supervised Machine Learning Training With Electronic Health Records: Case Study for the Extraction of Symptoms From Clinical Notes." *JMIR Medical Informatics* 10 (3): e32903. https://doi.org/10.2196/32903.

Jang, Eric, Shixiang Gu, and Ben Poole. 2016. "Categorical Reparameterization with Gumbel-Softmax." *ArXiv Preprint ArXiv:1611.01144*.

Johnson, Alistair, Tom Pollard, and Roger Mark. 2015. "MIMIC-III Clinical Database." PhysioNet. https://doi.org/10.13026/C2XW26.

Jones, Zachary M. 2013. "Git/GitHub, Transparency, and Legitimacy in Quantitative Research," 2.

Kalliamvakou, Eirini, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2016. "An In-Depth Study of the Promises and Perils of Mining GitHub." *Empirical Software Engineering* 21 (5): 2035–71. https://doi.org/10.1007/s10664-015-9393-5.

Kearns, Jodi. 2014. "Librivox: Free Public Domain Audiobooks." *Reference Reviews*.

Këpuska, Veton Z, Hussien A Elharati, and others. 2015. "Robust Speech Recognition System Using Conventional and Hybrid Features of MFCC, LPCC, PLP, RASTA-PLP and Hidden Markov Model Classifier in Noisy Conditions." *Journal of Computer and Communications* 3 (06): 1.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2017. "ImageNet Classification with Deep Convolutional Neural Networks." *Commun. ACM* 60 (6): 84–90. https://doi.org/10.1145/3065386.

Kujala, Sari, Tarja Heponiemi, Iliris Hörhammer, and Riitta Hänninen-Ervasti. 2020. "Health Professionals' Experiences of the Benefits and Challenges of Online Symptom Checkers." *Digital Personalized Health and Medicine*, 966–70. https://doi.org/10.3233/SHTI200305.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep Learning." *Nature* 521 (7553): 436–44.

Li, David, Kulamakan Kulasegaram, and Brian D. Hodges. 2019. "Why We Needn't Fear the Machines: Opportunities for Medicine in a Machine Learning World." *Academic Medicine* 94 (5): 623–25. https://doi.org/10.1097/ACM.0000000000002661.

Maas, Andrew, Quoc V Le, Tyler M O'neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y Ng. 2012. "Recurrent Neural Networks for Noise Reduction in Robust ASR."

Maier, Andreas. 2020. "Will We Ever Solve the Shortage of Data in Medical Applications?" Medium. June 11, 2020. https://towardsdatascience.com/will-we-ever-solve-the-shortage-of-data-in-medical-applications-70da163e2c2d.

McFee, Brian, Colin Raffel, Dawen Liang, Daniel Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. "Librosa: Audio and Music Signal Analysis in Python." In , 18–24. Austin, Texas. https://doi.org/10.25080/Majora-7b98e3ed-003.

Mergel, Ines. 2015. "Open Collaboration in the Public Sector: The Case of Social Coding on GitHub." *Government Information Quarterly* 32 (4): 464–72. https://doi.org/10.1016/j.giq.2015.09.004.

Mooney, Paul. 2018. “Medical Speech, Transcription, and Intent.”
Morris, Andrew, Viktoria Maier, and Phil Green. 2004. “From WER and RIL to MER and WIL: Improved Evaluation Measures for Connected Speech Recognition.” In .
MTHelpLine. 2022. “Transcribed Medical Transcription Sample Reports and Examples - MTSamples.” November 16, 2022. https://mtsamples.com/.
Muda, Lindasalwa, Mumtaj Begam, and Irraivan Elamvazuthi. 2010. “Voice Recognition Algorithms Using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques.” *ArXiv Preprint ArXiv:1003.4083*.
Müller, Regina, Malte Klemmt, Hans-Jörg Ehni, Tanja Henking, Angelina Kuhnmünch, Christine Preiser, Roland Koch, and Robert Ranisch. 2022. “Ethical, Legal, and Social Aspects of Symptom Checker Applications: A Scoping Review.” *Medicine, Health Care and Philosophy* 25 (4): 737–55. https://doi.org/10.1007/s11019-022-10114-y.
Nuthakki, Siddhartha, Sunil Neela, Judy W. Gichoya, and Saptarshi Purkayastha. 2019. “Natural Language Processing of MIMIC-III Clinical Notes for Identifying Diagnosis and Procedures with Neural Networks.” arXiv. http://arxiv.org/abs/1912.12397.
O’Shea, Keiron, and Ryan Nash. 2015. “An Introduction to Convolutional Neural Networks.” *ArXiv Preprint ArXiv:1511.08458*.
Padhye, Rohan, Senthil Mani, and Vibha Singhal Sinha. 2014. “A Study of External Community Contribution to Open-Source Projects on GitHub.” In *Proceedings of the 11th Working Conference on Mining Software Repositories*, 332–35. MSR 2014. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2597073.2597113.
Paling, Steven, Jennifer Lambert, Jasper Clouting, Júlia González-Esquerré, and Toby Auterson. 2020. “Waiting Times in Emergency Departments: Exploring the Factors Associated with Longer Patient Waits for Emergency Care in England Using Routinely Collected Daily Data.” *Emergency Medicine Journal* 37 (12): 781–86.
Panayotov, Vassil, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. “Librispeech: An Asr Corpus Based on Public Domain Audio Books.” In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5206–10. IEEE.
Panhuis, Willem G. van, Proma Paul, Claudia Emerson, John Grefenstette, Richard Wilder, Abraham J. Herbst, David Heymann, and Donald S. Burke. 2014. “A Systematic Review of Barriers to Data Sharing in Public Health.” *BMC Public Health* 14 (1): 1144. https://doi.org/10.1186/1471-2458-14-1144.
Paperspace. 2022. “Notebooks Tutorial | Paperspace.” 2022. https://docs.paperspace.com/gradient/tutorials/notebooks-tutorial/.
Park, Andrea. 2021. “Symptom Checking Platform Developer Ada Health Inks Partnerships with Sanofi, Takeda, Alnylam.” Fierce Biotech. June 14, 2021. https://www.fiercebiotech.com/medtech/symptom-checking-platform-ada-health-inks-partnerships-takeda-sanofi-alnylam.
Perez-Riverol, Yasset, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, et al. 2016. “Ten Simple Rules for Taking Advantage of Git and GitHub.” *PLoS Computational Biology* 12 (7): e1004947. https://doi.org/10.1371/journal.pcbi.1004947.
Putnam, Cheryl. 2010. “Mayo Clinic.” *Journal of Consumer Health on the Internet* 14 (4): 392–400. https://doi.org/10.1080/15398285.2010.524122.
Ruan, Sherry, Jacob O Wobbrock, Kenny Liou, Andrew Ng, and James Landay. 2016. “Speech Is 3x Faster than Typing for English and Mandarin Text Entry on Mobile Devices.” *ArXiv Preprint ArXiv:1608.07323*.
Saltzer, Jerome H. 2020. “The Origin of the ‘Mit License.’” *IEEE Annals of the History of Computing* 42 (4): 94–98.
Singh, A. K. Bhavani, Mounika Guntu, Ananth Reddy Bhimireddy, Judy W. Gichoya, and Saptarshi Purkayastha. 2020. “Multi-Label Natural Language Processing to Identify Diagnosis and Procedure Codes from MIMIC-III Inpatient Notes.” arXiv. http://arxiv.org/abs/2003.07507.

Son, Jaemin, Joo Young Shin, Hoon Dong Kim, Kyu-Hwan Jung, Kyu Hyung Park, and Sang Jun Park. 2020. “Development and Validation of Deep Learning Models for Screening Multiple Abnormal Findings in Retinal Fundus Images.” *Ophthalmology* 127 (1): 85–94. https://doi.org/10.1016/j.ophtha.2019.05.029.

Sottile, Anthony. 2017. “Pre-Commit.” *A Framework for Managing and Maintaining Multi-Language Pre-Commit Hooks.* https://pre-commit.com/.

Sphinx. 2022. “Sphinx Documentation,” 641.

“Symptom Checker - Mayo Clinic.” 2022. November 22, 2022. https://www.mayoclinic.org/symptom-checker/select-symptom/itt-20009075.

“Symptom Checker with Body from WebMD - Check Your Medical Symptoms.” 2022. WebMD. November 21, 2022. https://symptoms.webmd.com/.

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. “Going Deeper with Convolutions.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9.

Tian, Yonglong, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. 2020. “What Makes for Good Views for Contrastive Learning?” *Advances in Neural Information Processing Systems* 33: 6827–39.

Wang, Yingzhi, Abdelmoumene Boumadane, and Abdelwahab Heba. 2021. “A Fine-Tuned Wav2vec 2.0/Hubert Benchmark for Speech Emotion Recognition, Speaker Verification and Spoken Language Understanding.” *ArXiv Preprint ArXiv:2111.02735*.

Weik, Martin. 2012. *Communications Standard Dictionary*. Springer Science & Business Media.

Wu, Patrick, Aliya Gifford, Xiangrui Meng, Xue Li, Harry Campbell, Tim Varley, Juan Zhao, et al. 2019. “Mapping ICD-10 and ICD-10-CM Codes to Phecodes: Workflow Development and Initial Evaluation.” *JMIR Medical Informatics* 7 (4): e14325. https://doi.org/10.2196/14325.

Yu, Dong, and Li Deng. 2016. *Automatic Speech Recognition*. Vol. 1. Springer.

Zhan, Xianghao, Marie Humbert-Droz, Pritam Mukherjee, and Olivier Gevaert. 2021. “Structuring Clinical Text with AI: Old vs. New Natural Language Processing Techniques Evaluated on Eight Common Cardiovascular Diseases.” medRxiv. https://doi.org/10.1101/2021.01.27.21250477.

Appendix

3.2.1 Speech to Text

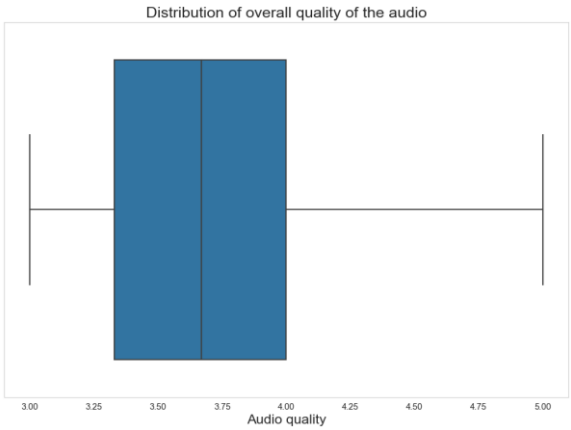

Figure 13: Boxplot of speech to text audio quality. The audio quality ranges from 3 to 5 points, with an average of 3.68 points. (Source: Figure created by authors).

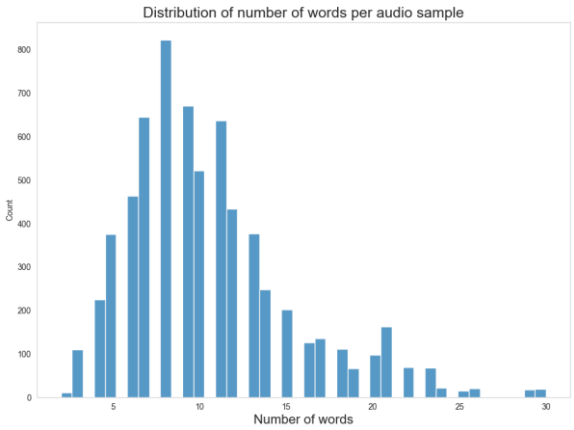

Figure 14: Histogram of speech to text number of words in the label column. Most of the phrases are 8 words long, although the longest phrases contain up to 29 words. (Source: Figure created by authors).

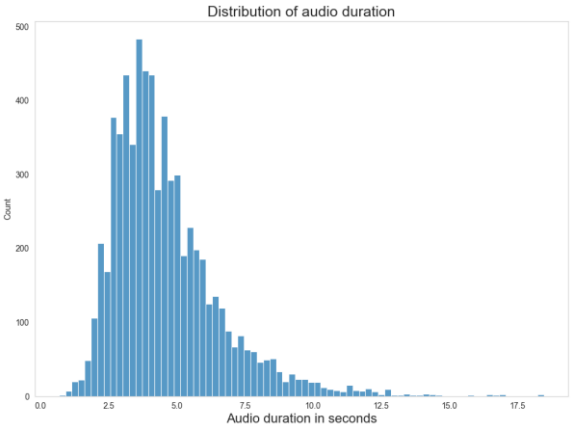

Figure 15: Histogram of the length of the speech to text audio files. The majority of the audio files are between 0 and 100 seconds long. The distribution is skewed to the right, which means that there are a few audio files that are very long, and most of them are between 2.5 and 5.0 seconds. (Source: Figure created by authors).

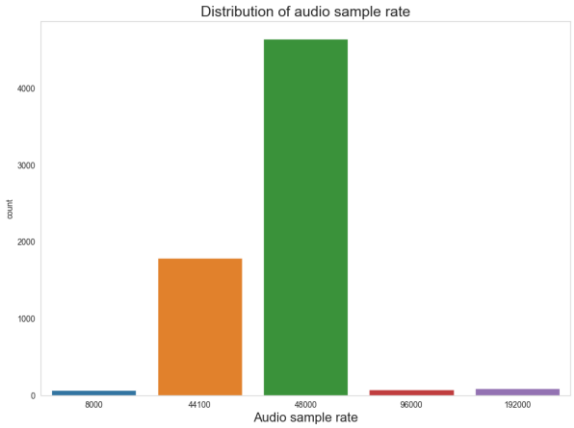

Figure 16: Bar chart showing how often each sample rate occurs. The sample rate is the number of samples of audio carried per second, measured in Hertz (Hz). The sample rate is the number of times a sound wave is measured per second. The most common audio sample rate is 48000, followed by 44100. (Source: Figure created by authors).

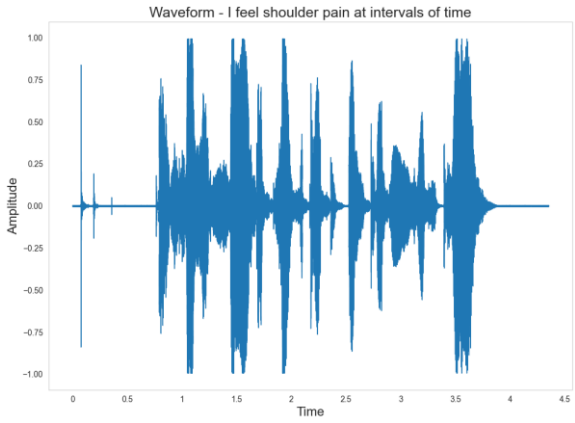

Figure 17: Waveform of sample audio file. The vertical axis of the waveform shows the amplitude of the sound, while the horizontal axis shows the time. The peaks and valleys indicate a dynamic sound as each peak represents a word, while a valley represents silence. (Source: Figure created by authors).

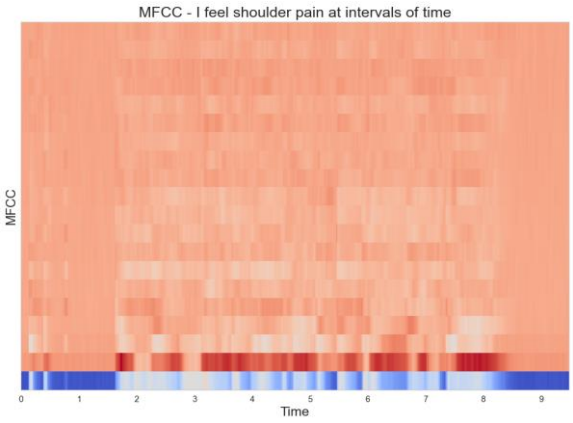

Figure 18: Mel-frequency cepstral coefficients of sample audio file. The MFCC figure displays the strength of the audio signal's various frequency components as measured at various points in time. (Source: Figure created by authors).

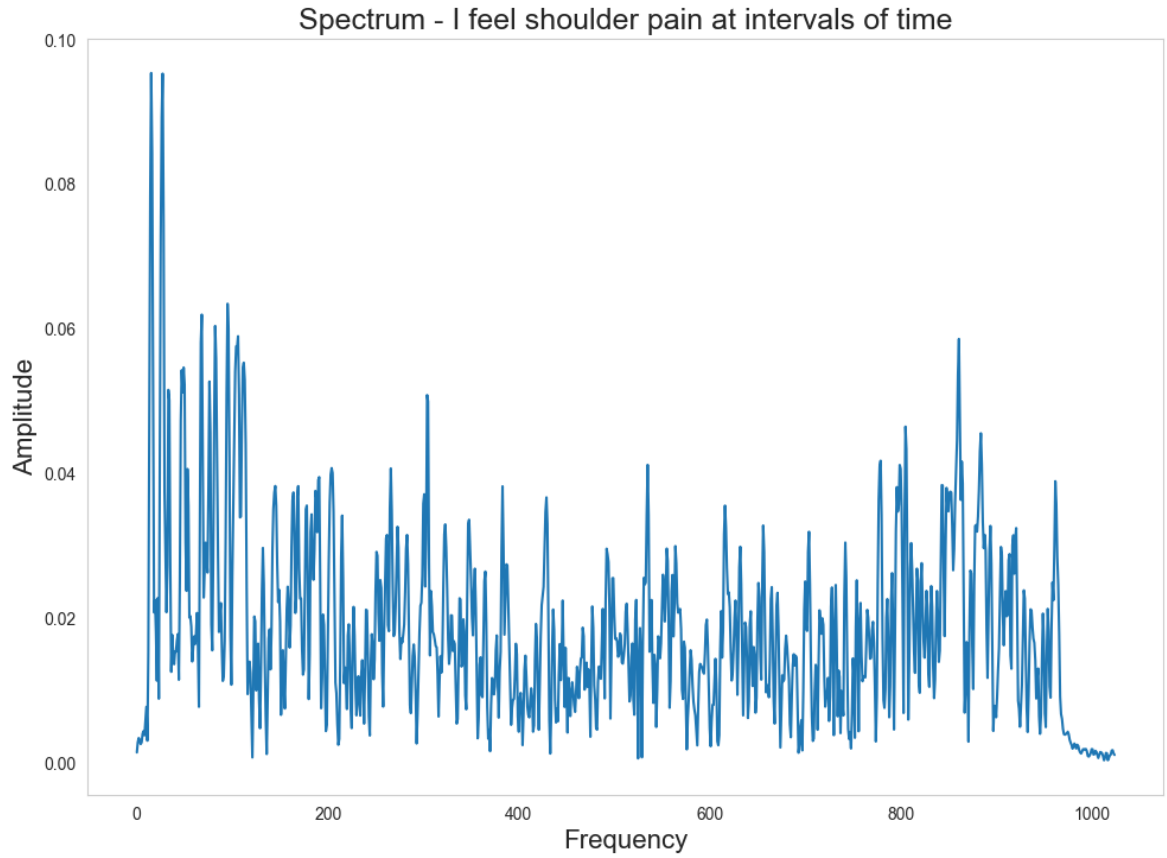

Figure 19: Spectrum of sample audio file. An audio file's spectrum is a graphic depiction of the sound's many frequencies. It displays how the sound's various frequency components stack up in terms of relative strength. The spectrum can help you get a feel of the sound's general character and balance, as well as any standout elements or patterns that might be present. (Source: Figure created by authors).

3.2.2 MT Samples

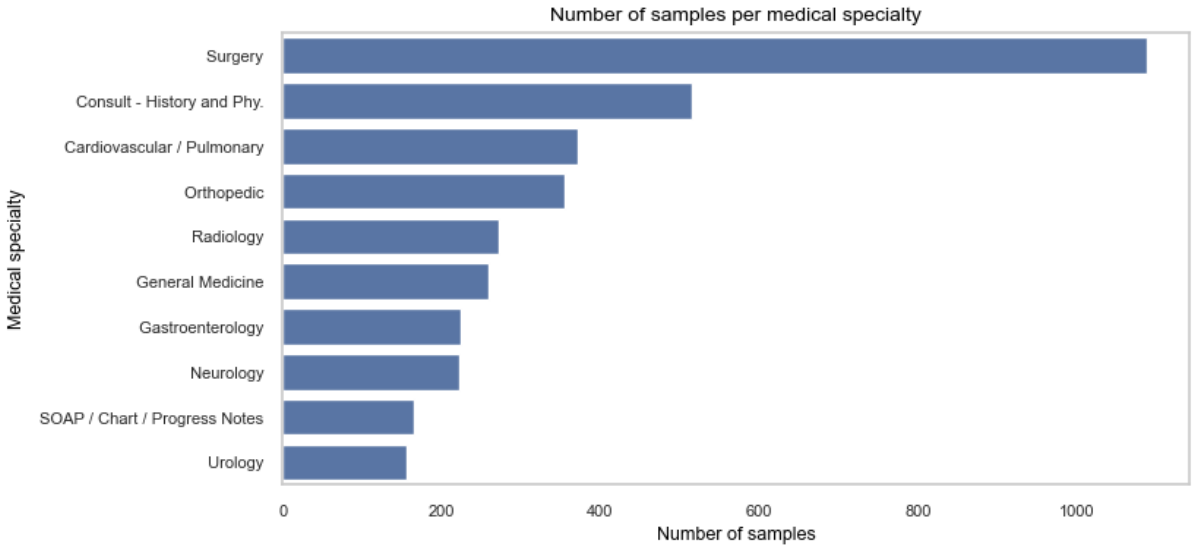

Figure 20: Number of Samples per Medical Specialty in the MT Samples dataset. The figure shows only the top 10 medical specialties by the number of samples. Overall, there are 16 different medical specialties. (Source: Figure created by authors).

3.2.3 MIMIC-III

ICD Chapter code	Medical Specialty
Certain Conditions Originating In The Perinatal Period	Obstetrics & Gynaecology
Complications Of Pregnancy, Childbirth, And The Puerperium	Obstetrics & Gynaecology
Congenital Anomalies	Primary Care
Diseases Of The Blood And Blood-Forming Organs	Hematology
Diseases Of The Circulatory System	Cardiothoracic & Vascular
Diseases Of The Digestive System	Gastroenterology
Diseases Of The Genitourinary System	Urology
Diseases Of The Musculoskeletal System And Connective Tissue	Orthopedic surgery
Diseases Of The Nervous System And Sense Organs	Neurology
Diseases Of The Respiratory System	Pulmonology
Diseases Of The Skin And Subcutaneous Tissue"	Dermatology
Endocrine, Nutritional And Metabolic Diseases, And Immunity Disorders	Endocrinology
Infectious And Parasitic Diseases	Infectious Disease Specialty
Injury And Poisoning	Emergency Department
Mental Disorders	Psychiatry
Neoplasms	Oncology
Supplementary Classification Of External Causes Of Injury And Poisoning	Emergency Department
Supplementary Classification Of Factors Influencing Health Status And Contact With Health Services	Internal Medicine Department
Symptoms, Signs, And Ill-Defined Conditions	Internal Medicine Department

Table 5: Mappings of ICD-9 chapter codes and medical specialties. 19 different chapter codes are mapped to 16 different medical specialties. The mapping process is described in section 3.2.3. (Source: Table created by authors).

4.1 GitHub Structure

Figure 21: Structure of GitHub repository. Depicted are the most important folders of the repository and the structure for the "src" folder. This figure illustrates the transparent structure and the possibility of recycling code based on the modular setup. (Source: Figure created by authors).

4.2 Streamlit Application

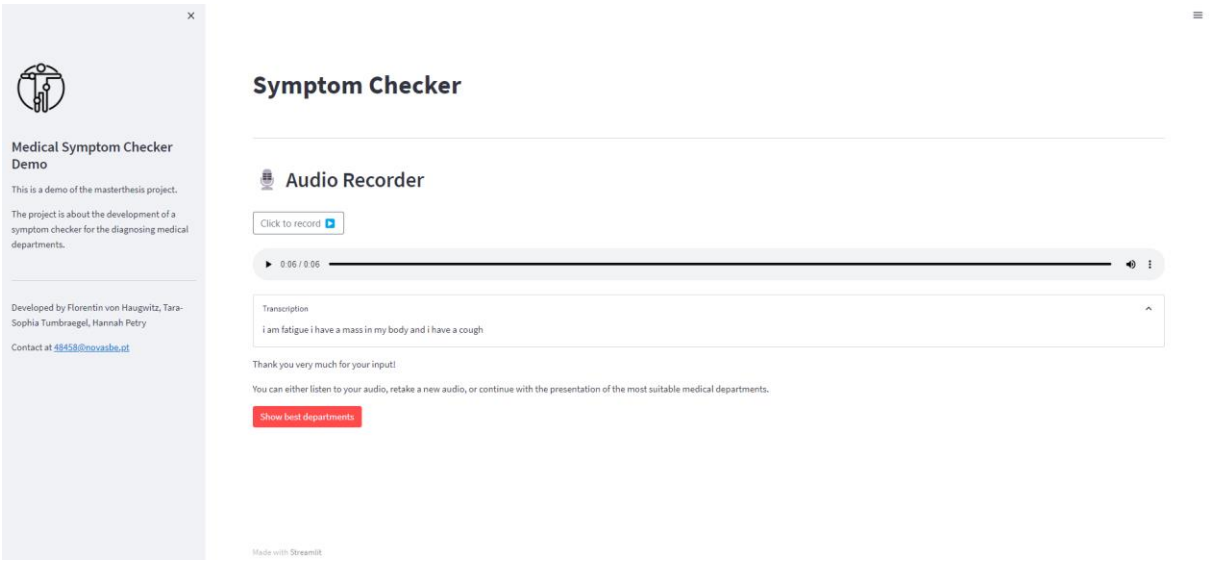

Figure 22: Layout of starting page of the application. The user can record their audio input, listen to it, and see the performed transcription. If they are satisfied with their input, they can click on the red button to see the most suitable medical departments. (Source: Figure created by authors).

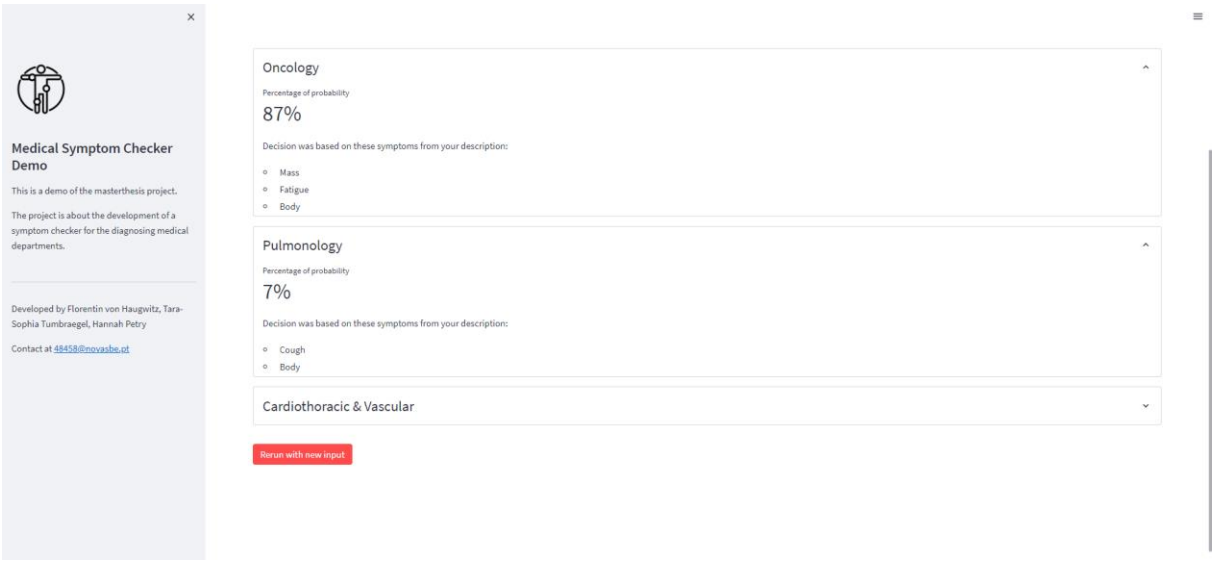

Figure 23: Layout of the classification page of the application. The user sees the three most suitable medical departments based on their input. Each department recommendation can be expanded to show the probability percentage and the keywords which led to the respective decision. (Source: Figure created by authors).

5.4 Results and Evaluation

Reference Phrase	Predicted Phrase	WER Score	CER Score
my stomach hurts	my stomag hurts	0.333	0.125
i don't have pain	i do not have pain	0.5	0.118
my knee pains me a lot	my nee paints me a lot	0.333	0.091

Table 6: Evaluation metrics for three samples. This shows that small mistakes mostly have a large influence on the WER but a rather smaller one on the CER. The first example has only one spelling mistake, the second example predicts an entire word wrong, and the third one has two spelling mistakes. (Source: Table created by authors).

Turnitin Originality Report

Processed on: 14-Dec-2022 19:03 WET
ID: 1981324894
Word Count: 30687
Submitted: 1

2022-23_Fall_48458_Hannah Petry_48333_Tara-Sophia Tumbraegel_48174_Florentin von Haugwitz
By Hannah Petry

Similarity Index	Similarity by Source
2%	Internet Sources: 1% Publications: 1% Student Papers: 1%

< 1% match ("Collaboration Across Boundaries for Social-Ecological Systems Science", Springer Science and Business Media LLC, 2019)
"Collaboration Across Boundaries for Social-Ecological Systems Science", Springer Science and Business Media LLC, 2019

< 1% match (student papers from 20-May-2022)
Submitted to Universidade Nova De Lisboa on 2022-05-20

< 1% match (student papers from 12-Dec-2022)
Submitted to Universidade Nova De Lisboa on 2022-12-12

< 1% match (Medicolegal Library, 1986.)
Medicolegal Library, 1986.

< 1% match (student papers from 12-Aug-2021)
Submitted to University of Leeds on 2021-08-12

< 1% match (Soraya Sedkaoui. "chapter 2 Statistical and Computational Needs for Big Data Challenges", IGI Global, 2018)
Soraya Sedkaoui. "chapter 2 Statistical and Computational Needs for Big Data Challenges", IGI Global, 2018

< 1% match (Internet from 04-Dec-2021)
https://www.fiercebiotech.com/medtech/symptom-checking-platform-ada-health-inks-partnerships-takeda-sanofi-alnylam

< 1% match (Jinmiao Huang, Cesar Osorio, Luke Wicent Sy. "An empirical evaluation of deep learning for ICD-9 code assignment using MIMIC-III clinical notes", Computer Methods and Programs in Biomedicine, 2019)
Jinmiao Huang, Cesar Osorio, Luke Wicent Sy. "An empirical evaluation of deep learning for ICD-9 code assignment using MIMIC-III clinical notes", Computer Methods and Programs in Biomedicine, 2019

< 1% match (Internet from 29-Oct-2022)
https://research.aalto.fi/en/publications/health-professionals-experiences-of-the-benefits-and-challenges-o

< 1% match (Internet from 24-Sep-2022)
https://aaltodoc.aalto.fi/bitstream/handle/123456789/115108/master_Heino_Joel_2022.pdf?isAllowed=y&sequence=1

< 1% match (Internet from 24-Apr-2020)
http://docplayer.net/43871854-Population-health-management-report.html

< 1% match (Internet from 25-Oct-2022)
https://www.medrxiv.org/content/10.1101/2021.01.27.21250477v1.full.pdf

< 1% match (student papers from 29-Sep-2020)
Submitted to University College London on 2020-09-29

< 1% match (Cheryl Putnam. "Mayo Clinic ", Journal of Consumer Health on the Internet, 10/2010)
Cheryl Putnam. "Mayo Clinic ", Journal of Consumer Health on the Internet, 10/2010

< 1% match ("International Conference on Innovative Computing and Communications", Springer Science and Business Media LLC, 2022)
"International Conference on Innovative Computing and Communications", Springer Science and Business Media LLC, 2022

< 1% match (Manuella Adrian, Minh Van Truong, Tim Osazuwa. "Measuring Levels of Comorbidity in Drug User* Emergency Patients Treated in Ontario Hospitals", Substance Use & Misuse, 2009)
Manuella Adrian, Minh Van Truong, Tim Osazuwa. "Measuring Levels of Comorbidity in Drug User* Emergency Patients Treated in Ontario Hospitals", Substance Use & Misuse, 2009

< 1% match (Internet from 21-Jun-2010)
http://dspace.fsktm.um.edu.my/xmlui/bitstream/handle/1812/363/SDD.pdf?sequence=1

< 1% match (Internet from 24-Oct-2022)
https://www.arxiv-vanity.com/papers/1704.06300/

< 1% match (Internet from 14-Oct-2022)
https://www.coursehero.com/file/100524823/research-reportdocx/

< 1% match (Kaouter Karboub, Mohamed Tabaa. "A Machine Learning Based Discharge Prediction of Cardiovascular Diseases Patients in Intensive Care Units", Healthcare, 2022)
Kaouter Karboub, Mohamed Tabaa. "A Machine Learning Based Discharge Prediction of Cardiovascular Diseases Patients in Intensive Care Units", Healthcare, 2022

< 1% match (student papers from 16-Oct-2021)

Submitted to Harrisburg University of Science and Technology on 2021-10-16

< 1% match (Internet from 24-Sep-2022)
https://aclanthology.org/2021.iwslt-1.pdf

< 1% match (Internet from 13-Dec-2022)
https://ebin.pub/progress-in-artificial-intelligence-20th-epia-conference-on-artificial-intelligence-epia-2021-virtual-event-september-79-2021-proceedings-lecture-notes-in-computer-science-12981-1st-ed-2021-3030862291-9783030862299.html

< 1% match (Internet from 01-Oct-2022)
https://web.archive.org/web/20210711230853if_/https:/arxiv.org/pdf/2107.02975v1.pdf

< 1% match ("Intelligent Systems", Springer Science and Business Media LLC, 2020)
"Intelligent Systems", Springer Science and Business Media LLC, 2020

< 1% match (Antonius Rachmat Chrismanto, Anny Kartika Sari, Yohanes Suyanto. "SPAMID-PAIR: A Novel Indonesian Post–Comment Pairs Dataset Containing Emoji", International Journal of Advanced Computer Science and Applications, 2022)
Antonius Rachmat Chrismanto, Anny Kartika Sari, Yohanes Suyanto. "SPAMID-PAIR: A Novel Indonesian Post–Comment Pairs Dataset Containing Emoji", International Journal of Advanced Computer Science and Applications, 2022

< 1% match (Lecture Notes in Computer Science, 2003.)
Lecture Notes in Computer Science, 2003.

< 1% match (Rodrigo Sandoval-Almazan, J. Ramon Gil-Garcia, David Valle-Cruz. "Chapter 9 Going Beyond Bureaucracy Through Gamification: Innovation Labs and Citizen Engagement in the Case of "Mapaton" in Mexico City", Springer Science and Business Media LLC, 2017)
Rodrigo Sandoval-Almazan, J. Ramon Gil-Garcia, David Valle-Cruz. "Chapter 9 Going Beyond Bureaucracy Through Gamification: Innovation Labs and Citizen Engagement in the Case of "Mapaton" in Mexico City", Springer Science and Business Media LLC, 2017

< 1% match (Internet from 02-Jan-2022)
https://amtaweb.org/wp-content/uploads/2021/10/2021.mtsummit-research.pdf

< 1% match (Internet from 23-Oct-2022)
https://bmccancer.biomedcentral.com/articles/10.1186/s12885-021-08184-x

< 1% match (Internet from 05-Nov-2021)
https://christophm.github.io/interpretable-ml-book/interpretable-ml.pdf

< 1% match (Internet from 04-Nov-2022)
https://iacis.org/iis/2022/3_iis_2022_53-63.pdf

< 1% match (Internet from 26-Sep-2022)
https://open.uct.ac.za/bitstream/handle/11427/36534/thesis_sci_2022_pillay%20nakkita.pdf;jsessionid=20D7F8D6963B1965EB4CDE23761D0(
sequence=1

< 1% match (Internet from 08-Jun-2020)
https://ro.ecu.edu.au/cgi/viewcontent.cgi?amp=&article=3313&context=theses

< 1% match (Internet from 12-Oct-2022)
https://scholarworks.iupui.edu/bitstream/handle/1805/30240/Nuthakki2019Natural-preprint.pdf?isAllowed=y&sequence=1

< 1% match (Internet from 13-Nov-2021)
https://www.journaltocs.ac.uk/index.php?action=browse&journalID=28434&pageb=1&publisherID=450&subAction=pub

< 1% match (Felipe de Souza Salvatore. "Analyzing natural language inference from a rigorous point of view", Universidade de Sao Paulo, Agencia USP de Gestao da Informacao Academica (AGUIA), 2020)
Felipe de Souza Salvatore. "Analyzing natural language inference from a rigorous point of view", Universidade de Sao Paulo, Agencia USP de Gestao da Informacao Academica (AGUIA), 2020

< 1% match (Jaroslav Michalek, Jakub Odehnal, Marek Sedlaik. "A Competitiveness Evaluation of the Ukrainian Regions", Engineering Economics, 2012)
Jaroslav Michalek, Jakub Odehnal, Marek Sedlaik. "A Competitiveness Evaluation of the Ukrainian Regions", Engineering Economics, 2012

ID Cover Page Summary of WP Student Team Development of an Innovative and Transparent Symptom Checker Group constitution: Student Name Program Individual Title Florentin von Haugwitz Business Analytics Development of an Innovative and Transparent Symptom Checker with a Focus on Automatic Speech Recognition Tara-Sophia Tumbraegel Business Analytics Development of an Innovative and Transparent Symptom Checker with a Focus on Natural Language Processing Hannah Petry Business Analytics Development of an Innovative and Transparent Symptom Checker with a Focus on Multi-Label Classification Work project carried out under the supervision of: Advisor: Rongjiao Ji Co-advisor (if applicable): Insert co-advisor name Reading observations for jury: If applicable Note: this template is a support document for the jury members before the defense, only. A Work Project presented as part of the requirements for the Award of a Master's degree in Business Analytics from the Nova School of Business and Economics. DEVELOPMENT OF AN INNOVATIVE AND TRANSPARENT SYMPTOM CHECKER Development of an Innovative and Transparent Symptom Checker with a Focus on Automatic Speech Recognition - Florentin von Haugwitz (48174) Development of an Innovative and Transparent Symptom Checker with a Focus on Natural Language Processing - Tara-Sophia Tumbraegel (48333) Development of an Innovative and Transparent Symptom Checker with a Focus on Multi-Label Classification - Hannah Petry (48458) Work project carried out under the supervision of: RONGJIAO JI 14/12/2022 Abstract This paper presents the creation of a medical symptom checker with state-of-the-art machine and deep learning technologies. It examines the use and development of speech-to-text, natural language processing, and classification models, which are trained on medical datasets. All models are discussed, highlighting their advantages and disadvantages for the study. Moreover, the paper introduces a web application which provides a user-friendly interface, allowing users to interact with the models and showcase the results. Finally, the paper offers an outlook on the

future use cases of the application and how it may improve healthcare outcomes. Keywords: Deep learning, Machine Learning, Classification, Speech to text, Natural Language Processing, Medical symptom checker, Streamlit This work used infrastructure and resources funded by CMU Portugal program explorative research project MD2TRUST (CMU/TIC/0016/2021). Table of Contents Abstract ..... 1 1 Introduction..... 3 2 Literature Review and Existing Applications ..... 6 2.1 Existing Symptom Checker ..... 6 2.2 Existing Work on the Datasets..... 8 3 Data Sources ..... 9 3.1 Data Finding ..... 10 3.2 Data Exploration..... 11 3.2.1 Speech to Text ..... 11 3.2.2 MT Samples ..... 15 3.2.3 MIMIC-III ..... 16 4 Accomplishments ..... 18 4.1 GitHub Structure ..... 19 4.2 Streamlit Application ..... 22 4.3 Practical Implications..... 23 4.3.1 Use Cases..... 23 4.3.2 Reusability ..... 24 5 Individual Parts ..... 25 5.1 Speech to Text (Florentin von Haugwitz, 48174) ..... 25 5.1.1 Wav2vec2.0 Model..... 26 5.1.2 HuBERT Model ..... 30 5.1.3 Model Fine-tuning ..... 33 5.1.4 Results and Evaluation ..... 37 5.2 Natural Language Processing (Tara-Sophia Tumbraegel, 48333) ..... 40 5.2.1 Current State and Innovations ..... 41 5.2.2 BERT Model ..... 43 5.2.3 Medical Corpus BERT ..... 46 5.2.4 Bio_ClinicalBERT Fine-tuning..... 47 5.2.5 Results and Evaluation with KeyBERT ..... 51 5.3 Classification (Hannah Petry, 48458) ..... 56 5.3.1 Performance and Explainability Trade-off ..... 57 5.3.2 Preprocessing ..... 58 5.3.3 Classification Models..... 60 5.3.4 Model Training ..... 66 5.3.5 Results and Evaluation ..... 67 6 Summary..... 71 7 Discussion..... 73 7.1 Limitations ..... 73 7.2 Outlook ..... 74 8 Conclusion..... 75 1 Introduction

Unnecessary emergency department visits, causing longer waiting times, can have life- threatening consequences for other patients (Costa and Godinho 2016). Most patients visit the emergency department due to the fear of serious health problems or lack of knowledge about which department is most suitable (Honigman et al. 2013). The thesis aims to use Artificial Intelligence (AI) to develop a prototype symptom checker application that patients and doctors can use for medical department and disease detection based on the patient's oral description of their symptoms. The benefit of this application is the reduction of the number of emergency department visits as patients are a more informed which department to consult. It is to be proven whether such a symptom checker can be developed and whether this is a value-adding process for a user. The problem patients are facing is the lack of knowledge of where to seek help for their health problems and the symptoms they are experiencing. Often, they are stuck in the phone queue for a consultation or sent from medical department to department. This not only costs the health system money but is a waste of valuable time for the patient. Our study presents a customer- friendly solution to facilitate and speed up the initial process of diagnosing a patient to solve this issue which could be later expanded to also work as an assisting tool for doctors. We built an end-to-end solution with an easy-to-use interface, as illustrated in figure 1, starting with the audio description of a patient's symptoms, resulting in prediction probabilities for the three most relevant medical departments for the experienced symptoms. A distinctive feature of the product is that the patient has the comfort of expressing his symptoms in spoken language instead of choosing them from a given list of symptoms which adds to a good user experience. Furthermore, the entire process is very transparent, from the initial input to the final output, compared to other existing solutions. The patient will also be able to understand why the model made a prediction which increases its trustworthiness. Figure 1: An illustration of the user journey and the architecture behind the web interface. The architecture is divided into three parts for which a final model was built, namely speech-to-text recognition, transcribing the oral patient's description into text, natural language processing which filters the transcription for keywords, and classification, which predicts medical departments based on the keywords. (Source: Figure created by authors). Symptom checkers are ready to be accepted by users. A recent study examines the benefits and challenges of online symptom checkers and reveals that most healthcare professionals advocate the use of them. They allow patients to quickly connect with a healthcare professional and refer to self-treatment instructions, regardless of time and place. Moreover, medical professionals are pleased that the symptom checker provides more helpful information before meeting with the patient (Kujala et al. 2020). Hence, there is a market ready to be entered, also because patients increasingly turn to online resources to self-diagnose their symptoms before consulting a medical professional (Hochberg, Allon, and Yom-Tov 2020). If a symptom checker empowers patients through complementary information to make informed decisions, this contributes to increasing the efficiency of the healthcare system and advancing disease prediction. Furthermore, through our work, we aim to draw attention to the use of AI and machine learning in health care since it has the power to revolutionize the healthcare system. Moreover, deep 4 learning, which uses artificial neural networks to identify patterns and learn from large amounts of data, is used in this study for the tasks of speech recognition and natural language processing. Machine and deep learning represent a tremendous opportunity for healthcare since it has the potential to improve the accuracy of predictions of both health and disease. Algorithms can be trained to make predictions based on data from large populations and can learn to identify too difficult or more time-consuming patterns for humans to discern. Additionally, they are used for personalized predictions based on an individual's characteristics, like our symptom checker. Machine learning is already used in healthcare to improve disease risk predictions and develop personalized treatment plans (Son et al. 2020). Other categories of machine learning in health care involve patient engagement and adherence along the care continuum or administrative activities such as medical records management (Davenport and Kalakota 2019). It is important to emphasize that machine learning in health care must not be seen as a threat to the profession of physicians but rather as an opportunity to move routine tasks to automated technology, which will lead to a shift in the scope of physicians' tasks. It will allow them

to dedicate more time to high-quality tasks like building meaningful relationships with their patients, making their human capabilities far more valuable than their diagnostic skills (Li, Kulasegaram, and Hodges 2019). This paper is organized into four main sections. First, a literature review is conducted to highlight existing applications providing a background to this study and illustrating how our approach is innovative. Second, the accomplishments regarding our programming and web application are explained. Third, the methodology and results are described in detail. This section covers the different models, our academic approach, and our evaluation. Finally, the findings and implications are examined in the discussion part. To conclude, having described the issues in a patient's diagnosis process and the current lack of autonomy and tools for a patient to make informed decisions, clearly, our symptom checker, developed with the use of AI, helps to solve those problems, and assists in the process of 5 consulting the correct medical department. 2 Literature Review and Existing Applications This section first lists and describes several existing symptom checkers which can be used as a comparison for this study, highlighting the special features of this prototype. Secondly, it looks at the existing literature on the datasets to take useful derivations for this study. 2.1 Existing Symptom Checker Comparing this thesis' symptom checker to existing symptom checkers provides an understanding of the competitive landscape and what makes this product unique. A comparison can help to refine the product. Additionally, a comparison to existing products can aid in identifying market gaps and innovation opportunities and understanding what customers are looking for to develop a successful product. It is especially useful since it is a new market area, and there are not too many ready-made products on the market yet. People can quickly find potential causes for their medical problems using symptom checkers. Users typically describe their symptoms or choose similar symptoms from a list and sometimes answer specific follow-up questions that help to focus on the most likely causes (Müller et al. 2022). When searching online for publicly available symptom checkers, WebMD, one of the most popular online symptom checkers in the United States, appears at the top of the search page. According to a study published in the British Medical Journal, the WebMD symptom checker has a diagnostic accuracy of 35.5%, where the top three condition suggestions contain the primary diagnosis (Gilbert et al. 2020). The WebMD symptom checker only allows one to select from a given list of symptoms instead of describing them. It is, however, possible to add biodata and select symptoms by body location. As an intermediate step, one must rank the selected symptoms and can answer some optional questions to exclude certain conditions like pregnancy, for example. This symptom checker outputs a list of conditions with the tag of how likely the 6 match is from strong to low that match the selected symptoms. One receives additional information about condition details and treatment options. The tool lacks information about the most important symptoms to predict a diagnosis ("Symptom Checker with Body from WebMD - Check Your Medical Symptoms" 2022). The online symptom checker of the Mayo Clinic also appears frequently in web searches. ", Journal of Consumer Health on the Internet, 10/2010">The Mayo Clinic is a not-for-profit group medical practice based ", Journal of Consumer Health on the Internet, 10/2010">in the United States ", Journal of Consumer Health on the Internet, 10/2010">and offers health information on its website (Putnam 2010). One can choose from a list of 45 symptoms and select related factors for the chosen symptom. Afterward, a list of diseases and conditions appears that match at least one of the factors selected. The diseases with the most matches are listed first. The user is provided with information on which factors relate to which disease and other factors related to the disease which have not been mentioned. If a disease is clicked on, a more detailed description is displayed, such as symptoms, advice on when to see a doctor, or risk factors. The Mayo Clinic also states that its symptom checker is not a diagnostic tool but a tool helping to narrow down the search ("Symptom Checker - Mayo Clinic" 2022). According to studies from several medical journals referenced by the health search engine openmd.com the best free symptom checker regarding condition coverage, diagnostic accuracy, and appropriateness of urgent care advice is the smartphone app Ada which requires registration for usage ("5 Best Medical Symptom Checkers" 2022). Founded in 2011, the German start-up now has over 12 million users, and its medical intelligence achieves a diagnostic accuracy of 70.5% (Gilbert et al. 2020). A chatbot interface guides the user through a questionnaire about the symptoms they are experiencing. For every symptom, it is asked about its length, the exact location, the intensity, and potentially related symptoms. After gathering enough information, a report with potential conditions is created. There is a more detailed description for every condition, and the probability of a condition is given by indicating how many people out of 10 with the same symptoms also experience this condition ("Health. Powered by Ada." 2022). 7 According to an article by Andrea Park, Ada is a much more powerful version of the standard WebMD symptom checker. It is continuously trained on a vast database of millions of clinical observations (Park 2021). However, none of the symptom checkers mentioned disclosing their models or the data they use for their algorithms as we did. Other research dealing with symptom checkers often only focuses on a specific medical field and does not build a viable end-to-end product (Akrout et al. 2019; Hügle et al. 2020). A viable end-to-end product consists not only of a proposed and tested algorithm for a problem as presented by various papers but also of the implementation of the solution in an interface that a user can work with, as is the case with the symptom checker of our project. 2.2 Existing Work on the Datasets Automation is becoming increasingly important in the health sector, and more and more research is being conducted in this area. The healthcare industry faces problems of cost, spending, and increasing waiting times. Therefore, there is a high motivation for hospitals and researchers to conduct research and analyze medical data to find efficiencies and cost reduction potentials (Nuthakki et al. 2019). There are different data sets in the medical field which are usable for free. This thesis focusses on three different ones. The MIMIC-III dataset and the MT Samples dataset, and the Medical Speech, Transcription, and Intent (MSTI) dataset. The datasets enjoy different scientific recognition. All datasets have already been used in significant studies or programming projects, but only the MIMIC-III dataset has relevant studies to compare with our task. Therefore, this analysis refers only to the MIMIC-III dataset. The MIMC-III is one of the biggest publicly available datasets in the medical domain. It has been a valuable resource for researchers in a variety of fields and has enabled a wide range of studies that would not have been possible without its availability. Its use has provided insights into the risk factors, costs, and effectiveness of various medical interventions, as well as the 8 performance of decision support systems. Most of the research follows a similar goal as ours, trying to predict ICD codes based on the doctor's report. Our approach, in addition, finds a connection between the ICD-9 code to the department. Recent works use NLP algorithms to process the MIMIC-III database. For instance, a study by Huang et al. compares NLP methods to traditional classification methods with manually created rules to extract ICD-9 codes from the patient's text (Huang, Osorio, and Sy 2019). A paper by Xianghao Zhan et al. compares five different NLP techniques which improve the quality of diagnostic codes by extracting structured diagnostic codes from unstructured notes concerning cardiovascular diseases (Zhan et al. 2021). Siddhartha Nuthakki et al. focus on mapping clinical notes to medical codes and predicting the final diagnosis for the top ten most occurring diagnoses. They use the deep learning method, ULMFiT, and achieve an accuracy of 80.5% for the top ten diagnoses (Nuthakki et al. 2019). The paper "multi-label natural language processing to identify diagnosis and procedure codes from MIMIC-III inpatient notes" by AK Bhavani Singh et al. tries to predict the top 10 and top 50 ICD codes with the BERT model that is also being used in our study. They achieve an accuracy score of 87% (Singh et al. 2020, 2). In conclusion, this literature review has provided a comprehensive overview of the study of medical symptom checkers and the MIMIC-III dataset. A variety of research approaches and methods have been employed to investigate this topic. However, there is still a lack of a baseline for the community to reliably assess different algorithms on benchmark datasets. Also, it highlights the need to further explore the topic as there is room for further improvement for both the work on the MIMIC-III dataset and how to best develop a user-friendly symptom checker. 3 Data Sources This section is concerned with the data finding and exploration of our project. The data finding is explained to help readers comprehend the research process and to demonstrate the validity and dependability of the data utilized in this study. Additionally, it provides readers with information 9 about the selection of the data and the decision-making process. Secondly, data exploration is an essential part of any machine learning project since it helps to determine the characteristics of the data and features that will be used for training. It helps to identify any potential issues with the data that need to be addressed before building a model. By understanding our data better, our model can be built more accurately and better suited to the problems of a symptom checker. 3.1 Data Finding In any data science project, one of the biggest challenges is

finding the right data which fits the problem that should be solved and is large enough to be statistically significant even after data cleaning and subset selection. Within the time frame of our master thesis, it was only possible to work with publicly available electronic healthcare data because negotiations with a private Portuguese hospital to support us with their electronic health records (EHR) were unsuccessful. Therefore, the data-finding process was managed by the team of this work project, and access to three different datasets was gained in the end. However, one must note that due to its nature, the availability of public healthcare data is extremely limited because of legal barriers like privacy regulations and the need to apply for access to specific databases (Humbert-Droz, Mukherjee, and Gevaert 2022). Furthermore, although healthcare data is very valuable for academic research and technology development, most of the existing datasets are relatively small (Maier 2020; van Panhuis et al. 2014). For our project, specific types of healthcare data are needed consisting of audio files, written text that includes the description of symptoms, and any assigned class labels such as medical departments or diseases. For the speech to text part of the project, labeled medical audio data is required. This means that every audio file needs to have a transcription. The data used for the project is called Medical Speech, Transcription, and Intent (MSTI) and is hosted on Kaggle.com (Mooney 2018). For the NLP and classification part, the work was started based on a scraped version from MTSamples.com, a public community platform website that contains sample medical transcription reports for different medical specialties initially created 10 to teach clinical coders and transcriptionists (MTHelpLine 2022). At the beginning of the project, we applied for access to the third version of the Multi-parameter Intelligent Monitoring in Intensive Care (MIMIC-III), an extensive clinical database including anonymized health- related data from more than 40,000 patients treated in the intensive care unit at Beth Israel Deaconess Medical Center between 2001 and 2012 (Johnson, Pollard, and Mark 2015). This process took over six weeks as the necessary credentials first had to be obtained, and a training course had to be completed before receiving access to the dataset. Although the time to work with the dataset was limited, it was decided to incorporate MIMIC-III into the project because the value-add is significant, as MIMIC-III is not only much larger than MT Samples but also contains codes from the international classification of diseases (ICD). Originally developed by the World Health Organization (WHO) to monitor mortality and morbidity, today, the ICD- codes are widely used by healthcare providers to accurately diagnose and document medical conditions, procedures, and treatment and to determine reimbursement and eligibility for health insurance coverage (P. Wu et al. 2019). 3.2 Data Exploration Data exploration is a process of inspecting, cleansing, and transforming data with the goal of discovering hidden patterns, suggesting conclusions, and supporting decision-making. For our study, it is crucial to understand the existing dataset in order to know how to clean the data best, how to map the ICD codes to medical departments, how to optimize the resources for our training, and how to detect unwanted outliers. 3.2.1 Speech to Text The MSTI dataset, which is used for fine-tuning the speech to text models, provides labeled audio snippets containing medical terminology (Mooney 2018). The exploration of the dataset is split into parts, as the first part focuses on the analysis of the metadata, while the second part focuses on the analysis of the audio data. The metadata is a CSV file which lists the paths to all 11 files, including further meta information regarding the quality of the audio, the transcribed label, the id of the speaker and transcriber, and the category of the audio. The second part depicts and generates key features of the audio files for analysis. The metadata file consists of 6661 samples and 13 columns. Since the columns "background_noise_audible, "background_noise_audible:confidence", "audio_clipping", "audio_clipping:confidence", "quiet_speaker:confidence", "background_noise_audible", "quiet_speaker", "file_download", and "prompt" do not provide any added value. Since the categories from the "prompt" column, which contains medical categories for the phrase, are not transferable to the categories from the other dataset, this column is no longer needed. For further feature exploration, the number of words each phrase contains was added to the dataset. The dataset contains zero duplicate samples and zero empty values, meaning no samples are dropped before the exploration of the dataset. The "phrase" column contains the transcription of the audio. On average, each phrase consists of 10.5 words, reaching from a minimum of two words to a maximum of thirty words (figure 23, appendix). A composition of the most frequent words, excluding general stop words, is shown in the word cloud beneath (figure 2). Figure 2: Word cloud of most frequent words based on the "phrase" column. The larger the word is written, the more often the word occurs in the mentioned column. The simple choice of words indicates that the training data rather consists of phrases by patients than professional doctors. (Source: Figure created by authors). The "overall_quality_of_the_audio" column assesses each sample and marks them between 1 and 5. In this dataset, the quality ranges between 3 and 5 points, with an average value of 3.67. After reviewing samples with a quality of 3.5 and lower, it was decided that these samples remain in the dataset as the quality is sufficient. An overview of the quality is depicted in figure 22 in the appendix. In total, the audio files were spoken by 124 people and transcribed by 45 people. The number of audio files spoken per person ranges from 1 to 75, while the number of transcriptions ranges from 15 to 445. The boxplot below underscores the distribution of the speakers (figure 3). It becomes clear that most of the speakers have spoken most of the audio files, which is not ideal for training as training the data with a broad range of voices culminates in better results when using the model in applications. Unfortunately, it is not possible to extract any more information about the speaker, such as gender, age, or if they are a native speaker. This would help to train and evaluate the quality of the predictions. In addition, it is not possible to retrieve information about the quality of the label. Exploring random samples proved that misspellings occur within the labels, which could lead to inaccurate results. 13 The data analysis is split between the analysis of the meta data and the audio files. To explore the audio files, which are stored in WAV format, the python package librosa was applied. This allows the extraction of individual characteristics from the audio files (McFee et al. 2015). Since the most important features regarding the audio data for this study are the audio length and the sample rate, both features were extracted from each audio sample. The sample rate defines how often per second a sound is sampled, thus the frequency of samples in a digital recording. The higher the sample rate is, the better quality of the sound sample (Weik 2012). The MSTI dataset provides audio samples with rates between 8000 and 192000 kHz. The exact sample rate distribution is represented in figure 25 in the appendix. As speech recognition models must be trained with audio data of the same sample rate, the audio files of the incorrect format must be resampled according to the standard sample rate of the used speech to text model. This step will be further explained in section 5.1.2. The length of the audio files is crucial as audio files which exceed a certain limit are likely to cause memory issues during the training period. The histogram of the audio length files is depicted below (figure 4), showing that most of the file's audio duration ranges between 3.25 and 5.5 seconds, with outliers reaching up to 18.5 seconds. Figure 3: Boxplot of how audio files are spoken per speaker. Most speakers speak 65 times, although outliers exist, with less than five audio recordings. (Source: Figure created by authors). Figure 4: Boxplot of audio duration in seconds. Most of the audio files are less than 5 seconds long, while outliers have a duration of over 17 seconds. (Source: Figure created by authors). A detailed exploration of a sample audio file is depicted in figures 26, 27, and 28 in the appendix, showing various audio features. 3.2.2 MT Samples The MT Samples dataset is a smaller-scaled dataset that provides labeled medical transcription sample reports and examples of initially 4,999 samples. After removing missing and duplicate values, a sample size of 3,817 remains. The predictor variable is the "transcription" column which refers to the transcription of a physician's dictation of the patient's symptoms and medical condition in textual format. The text column has an average length of 3,054 and a maximum of 18,425 characters. For cleaning this column, nan values, digits, spaces, and stop words were removed. Furthermore, the top 0.1% (110 words) of the most frequent words across all categories were removed, including "patient", "stay", or "hospital". Those words are not specific to a particular class and do not contribute to prediction. To ensure that no critical words are removed, one must manually check that only non-informative words are removed. The final text was shortened to an average of 498 characters. The text length of the individual medical specialties varies between 400 to 550 characters. Since not all reports are well formatted, 33 lines had to be removed. The categorical target variable is the "medical_specialty" column. There are initially 39 medical specialties. It becomes clear that this dataset is very unbalanced, and by far, the most frequently occurring category is "surgery", with 1021 occurrences in contrast to other categories that appear less than ten times. Even the top ten most frequent classes have a great variability in number of occurrences as can be seen in figure 29 in the appendix. After evaluation, all categories that appear less than 100 times are

removed, and eleven distinct categories that appear often enough to be used for training remain. Finally, 3100 samples are the final data size to work with. When deep diving into the quality of the cleaned text for each medical specialty, we can detect distinctive words and symptoms appearing in a different frequency per specialty. The exemplary figures 5 and 6 for class Obstetrics/Gynecology and Neurology also show that the approach of 15 using all the most frequent words will most likely not lead to great outcomes since most common words like "find", "leave" or "normal" do not have a high medical meaning. Figure 5: Wordcloud for the medical specialty Gynecology. The bigger the words, the more often they appear. (Source: Figure created by authors). Figure 6: Wordcloud for the medical specialty Neurology. The words which appear most often are "leave", "right", and "normal". (Source: Figure created by authors). 3.2.3 MIMIC-III The MIMIC-III dataset consists of multiple individual tables. The tables which are important for this project are the "diagnosis_icd" table and the "noteevents" table. The "diagnosis_icd" table includes the columns "ICD9_Code", "sequence_Id", and "subject_Id". Each patient has a unique "subject_Id". The "noteevents" table includes the "hadm_Id", the "subject_Id", the "category" column, as well as the "text" column. The predictor variable is the text column which is the doctor's report about a patient's stay. There are many unique categories which describe the type of report. After evaluating the quality of the different report types, it was decided to use only the discharge report category, analogous to the approach used by most other papers using MIMIC- III (Biseda et al. 2020). Each patient stay can have multiple ICD-9 codes. Since the model cannot predict multiple diseases, the dataset diagnoses are reduced to take only the most severe ICD-9 codes, which are all samples where the "sequence number" column is one. In the final step, both tables are merged on the "subject_Id" as well as the "hadm_Id". Since this project focuses on patient usability, the ICD-9 codes have been matched to medical specialties. Therefore, this approach uses only the reports which are labeled with an ICD-9 code, hence reducing the number of rows to 58,967 rows. To map the ICD codes to the medical departments, we have reduced each specific ICD-9 diagnose code to its ICD-9 chapter code by reducing the number to its first three digits. The 19 different ICD-9 chapter codes are mapped to 16 different medical specialties, as displayed in table 11 in the appendix. The mapping process was performed based on experience and academic references (Hansen et al. 2007; Scheurwegs et al. 2016; P. Wu et al. 2019). The text column has an average length of 10,391 characters and a maximum length of 55,000 characters. The discharge summary text is structured with different subsections. They can include family history, social history, hospital stay, and current history. Since the KeyBERT model is not able to distinguish between family history, old illnesses, and current illnesses, the text is filtered only for current conditions, so the extracted keywords are only the current symptoms of the patient. In the following cleaning steps, NaN values, digits, spaces, and stop words were removed. In addition, the top 0.1% (200 words) of the most frequent words across all categories were removed, including "patient", "doctor", and "home". To ensure that no important words are removed, it is necessary to manually check that only the uninformative words are withdrawn. The final text was shortened to 402 characters on average, whereas the average text lengths vary for the different categories between 800 to 300 characters. Since not all reports are well formatted, nine rows had to be dropped from further use. From the data exploration, we find that the dataset includes 46,520 different patients, which belong to 41 unique ethnicities, whereas seven ethnicities make up 90% of the data. Furthermore, most of the patients (40%) are married. The gender distribution is 44.1% for females and 55.9% for males. Overall, each patient has, on average, 13 hospital admissions, each with an average of 11 "sequence_Id", meaning eleven different ICD codes. When looking at the average length of a patient staying at the hospital, we can see in figure 7 that most of the patients stay between four to eight days. In figure 8, we can see the age distribution of the patients. The figure shows 17 that most of the patients are between 60 and 80 years. Figure 7: Number of patients staying time at the hospital in days. We can see that the data is right- skewed, meaning that some patients stay very long. Most of the patients stay around 4 to 8 days in the hospital. (Source: Figure created by authors). Figure 8: Number of patients per age group. We can see that most of the patients are between 60 to 80 years. Surprisingly there are very few people between 10-20 years. (Source: Figure created by authors). When looking at the distribution of the reports in figure 9 shows the top 10 medical specialties by frequency. It shows that the medical specialties are not evenly distributed. The occurrence of each category varies from 31% for Cardiothoracic & vascular reports to Dermatology which occurs less than 0.03% in the data. Figure 9: Number of samples per medical specialty. We can see that Cardiothoracic & Vascular is the most frequent. This chart shows only the top 10 most frequent categories. (Source: Figure created by authors). 4 Accomplishments This section describes the tools and software solutions used to code and train the different models, and it explains how we allow users to interact with them and how we make them accessible in a user-friendly fashion. In addition, a focus of our project was to create a structured code repository based on the latest coding principles, which provides transparency for researchers wanting to reference or reuse our methodology. Furthermore, an important aspect of the project is not only to present the feasibility of the project but also how our achievements are applicable to real-world use cases and how our work can be leveraged for future work. First, this section thoroughly explains our GitHub project structure, followed by our online web application, and ends with intended practical implications. 4.1 GitHub Structure The structure of the repository is composed of multiple smaller individual parts, which combined provide a state-of-the-art framework and the possibility to reproduce the obtained results and to further develop the models based on the existing code repository. To ensure efficient collaboration between all group members, the distributed version control system git and the source code hosting platform GitHub were used. Git allows multiple participants to work on the same project while tracking changes to committed files ensuring the traceability of the modifications and the ownership of each code block. It is very effective and efficient in archiving the complete history of a project by storing only the changes between files (Perez-Riverol et al. 2016). In addition, git enables every member of a project to work on individual tasks by separating these tasks into so-called branches. A branch has the advantage that it does not interfere with other branches and therefore maintains an isolated structure until branches are merged into a single main branch. To merge a separate branch into the main branch, the functionality of a pull request was used, which ensures that at least one other project member has signed off on the logic and format of the code that will be merged. GitHub, hosting over 10 million git repositories, is one of the most important software hosting platforms (Kalliamvakou et al. 2016). It provides a user-friendly interface for sharing and 19 collaborating on coding projects. This includes providing functionalities to proof-read and verify code from someone else, work on different work-streams simultaneously and host the code open- source. Hosting code open-source allows others to use the codebases as a reference, retrace individual steps and use it as a starting point for further similar projects (Jones 2013). For this project, GitHub was used to enforce a clean structure for the source code, have a transparent progression timeline, enable project members to oversee the work of others, and share code seamlessly. The code is open-source with an MIT License. This license allows users, among others, to copy, modify and distribute the code (Saltzer 2020). To structure the repository in a transparent and understandable manner, multiple sub-folders were created, each containing a separate logical compartment. The most important directories are the "src", "model", "notebooks", and "data" folders. An overview of the structure is shown in figure 30 in the appendix. The "src" directory contains three main subfolders, one for speech to text, one for natural language processing, and one for classification. Within each sub-folder, the python modules follow the same principles as the structures are comparable. This results in having one constants file to store all non-changing variables, a utils file for all multiple used functions, a data cleaning file to prepare the data, training files for building the models, a predict file to test the models, and an evaluate file to compare the results of the respective models. In addition, all source code relevant to the final application is stored in this folder as well. All machine learning and artificial intelligence models are stored in the "models" folder, categorized in the subfolder of either speech-to-text, natural language processing, or classification. To provide a reference for further users to understand the code and leverage the existing codebase, a documentation is created. The documentation is written with the python module sphinx, which generates reStructuredText as its markup language from docstrings in the python modules (Sphinx 2022). The documentation is hosted on GitHub pages, which is a documentation hosting service, making it accessible to the open-source community (https://tara-20 sophia.github.io/NLP_Masterthesis/). The documentation is automatically updated and redeployed on every push to the master branch of the repository. As an extension of our collaboration with git, we have used pre-commits, which force the

author of modification to a file in the repository to maintain the same structure and conventions as the remaining files. In our project, pre-commits enforce a uniform coding format, as well as the existence of docstrings and unit tests for all functions. Tests are used to ensure that the code is working as intended. The tests are written in pytest, a testing framework for python, which asserts that a set of inputs matches the expected output. The tests are run on every commit to the repository, whereas the commit fails if one of the tests does not pass (Sottile 2017) To ensure that the project can be run with the same packages installed as during the development phase of the project, a Conda environment was created. Conda is a free and open-source environment and package management system. Furthermore, it secures that every contributor to the project has the same version of a python package installed, streamlining the collaboration. The Conda environment file lists the required python packages that are needed for the project to function and simultaneously creates a virtual environment that is isolated from other python installations on the same machine (Anaconda 2016). In addition to the repository being available under this link (https://github.com/Tara- Sophia/NLP_Masterthesis), it will also be uploaded under the NOVA SBE GitHub account (https://github.com/Data-Science-Knowledge-Center-Nova-SBE). This increases the visibility of the project as users and followers of the NOVA SBE repositories also become aware of the project. In conclusion, using GitHub to host code is a great way to take advantage of the latest coding principles, incorporate third-party tools and ensure a high coding standard. GitHub provides a central repository for storing and managing code and offers a range of tools and features for reviewing and collaborating on code. This helped us to work more efficiently and effectively 21 and will allow future users interested in our work to understand and enhance our work more easily. 4.2 Streamlit Application In addition to using GitHub, a user interface was set up to allow users to interact with the application easily and quickly. For this purpose, the python module Streamlit is used, which makes it easy to create and share web apps. Streamlit provides an API that allows to create a web application by writing a python script which automatically generates HTML, CSS, and JavaScript files, which are necessary to run the web application (https://streamlit.io/). On the first page, the user of the application has the possibility to enter a voice-based input of symptoms. The speech input is processed by the speech-to-text model after the recording is completed and transcribed. Consequently, the text-based translation is passed to the Natural language processing model, which filters the most important words from the input and passes them to the classification model. To make it easier to use and clearer for the user, we decided against displaying the most important words. However, for the use of different target groups, e.g., doctors, the keywords can be displayed without any problems. On the next page, the user receives a list of the three most suitable categories, each with the percentage hit rate and a list of his input words that match this class. A "rerun" button returns the user to the initial page, accepting a new spoken input. Both pages of the web app are displayed in figures 31 and 32 in the appendix. Since one of our goals was to allow the users to have easy access to the application at anytime, anywhere, and for free, we have hosted it under the following URL:
https://huggingface.co/spaces/florentinhaugwitz/nlp_masterthesis_streamlit. Users can use this service therefore on their mobile devices without the need for any special requirements. This setup was chosen as it allows running the application on two virtual CPUs with a total of 16 GiB of RAM free of charge. Based on the size of the neural network models from the speech to text 22 and natural language tasks, these hardware specifications are required for a smooth user experience. Moreover, hosting the application on Hugging Face (https://huggingface.co/) permits an easy workflow for pushing updates to the application as it automatically pulls changes from a designated code repository. To summarize, this project introduces a free and online accessible application which allows users to enter their symptoms linguistically and receive recommendations for the best-suited medical departments based on their input. This can help to reduce anxiety and provide individuals with more accurate information, providing advice and guidance to those in need. Finally, by giving consumers a tool to self-diagnose and manage common health concerns, which can assist in avoiding needless doctor visits and hospitalizations, our free application can help to lessen the strain on the healthcare system and health issues, which can help to prevent unnecessary doctor visits and hospitalizations. 4.3 Practical Implications The realization of this project includes many use cases for different users. This includes the use of the application as well as the reuse and regeneration of the source code. Thus, the project serves a business benefit as well as a technical and academic benefit. 4.3.1 Use Cases Regarding the practical implications of this project, there are multiple use cases with different stakeholders who benefit from the results of this project. The first includes the use of the application from a patient's perspective, the second highlights a physician's interaction with the application, and finally, the benefit for the entire healthcare system is described in more detail. The first use case describes the use of a patient experiencing several symptoms. The main use of the application is to ascertain what type of disease may be present and what type of specialist should be consulted. In a few seconds and without answering several questions, as is often the case with existing solutions on the market, a patient can reflect on his symptom and feeling state. 23 Here, the use of technical terms is not necessary. Within a very short time, the most appropriate medical specialties are suggested to the patient in relation to their overall probability. The second use case illustrates how a doctor could interact with the application. Since the application classifies the medical departments which are mapped on ICD-10 category codes, the application can just as well provide these codes to a professional. This would allow the professional to already have a suggestion of which disease or diseases to examine for. The same application and application pipeline can also be used for specifying the diseases more granular, which is emphasized in section 7.2. The final use case tackles the problem of long waiting periods in the emergency department of hospitals. These long waiting periods cause a higher mortality rate and are caused, among other things, by an elaborate and timely referral process to other departments (Paling et al. 2020). The emergency department serves as a gateway to the hospital for many patients, even if they do not require immediate medical attention (Hogan and Bouknight 2002). An efficient solution to this problem is to incentivize the patients to inform themselves ahead of time to identify the correct specialty. Not only would the entire clinical system benefit from this, but also the individual would experience a better user experience as waiting times are shorter and the treatment is more specialized. The health system or the individual system could provide links to symptom checkers on their websites or natively include them to advertise them and thereby increase their popularity. 4.3.2 Reusability The value of reusing open-source code is immense for other data scientists, developers, and researchers. It enables the exchange and collaboration of ideas, knowledge, and code across individuals and organizations, fostering a culture of innovation and creativity (Mergel 2015). By allowing developers from all over the world to later collaborate on the project using open- source code on GitHub, it is possible to increase cooperation and transparency while improving 24 the possibility of developing innovative solutions. Additionally, the open-source code offers a platform for sharing knowledge, which can increase transparency and trust in the development process. Due to the frequent evaluation of open-source code by numerous developers, bugs and security flaws are frequently found and addressed rapidly, leading to more dependable products and higher code quality (Padhye, Mani, and Sinha 2014). Moreover, inaccurate results can be quickly identified and corrected, as the entire model consists of three transparent models each, and interim results can be examined at any stage. Additionally, visibility can be gained for this project which potentially leads to more users, contributors, and potential sponsors. 5 Individual Parts This section describes the methodology used to develop the individual models, with the intention of providing a detailed overview of the process. It is divided into the subsections of speech to text, natural language processing, and classification. Each section describes the functionality of the respective models which are used, the fundamental steps to train these, the evaluation metrics required, and the further implications of the results for the rest of the project. Additionally, it highlights our innovative approach, using state-of-the-art technology and trends to develop a very performant symptom checker while maintaining a high degree of explainability. 5.1 Speech to Text (Florentin von Haugwitz, 48174) In order to offer the user, the possibility to enter his symptoms via voice note, the first step of the project is to create a model that can convert spoken language into text. This step is included in the project as research has shown that speech input is three times faster than typing and more favorable with users (Ruan et al. 2016; Hauptmann and Rudnicky 1990). This adds to our overall goal of optimizing the user experience. Speech to text models belong to the group of automatic speech recognition (ASR) models, which are used in a variety of cutting-edge applications, such as voice search, voice assistants, and voice commands (Yu and Deng 2016). This

part of the project introduces two state-of-the-art ASR models and explains their architecture, setup, and functionality. In addition, this section illustrates how both models are fine-tuned on the MSTI dataset and how they are evaluated and assessed based on their usefulness in this project. As the output of the ASR model is the input for the NLP model in the application, it is critical that the model has a high performance. ASR models can be divided into two categories. The first category contains hybrid models, which are deep acoustic models. These models predict phonemes, the smallest unit of speech differentiating one-word element from another, from the audio input, which is transformed into a Mel Frequency Cepstral Coefficient (MFCC). The critical bandwidth of the human ear is known to vary with the frequency, which is the basis for MFCC and is prone to produce better results in ASR (Muda, Begam, and Elamvazuthi 2010). With the use of pronunciation dictionaries, phonemes are combined with choosing the most likely letter (Këpuska, Elharati, and others 2015). The other approach is to use a deep neural network, which is able to predict the words directly from the audio input or MFCCs. These types of models require more data and resources for training (Maas et al. 2012). For this project, Facebook's Wav2vec2.0 and Hubert models are used as base models, both belonging to the second category of ASR models. At first, the Wav2vec2.0 model is analyzed in detail, followed by the inspection of the HuBERT model. Afterward, the fine-tuning and optimization part for both models is depicted, resulting in the evaluation of the models and the implications for the rest of the project. 5.1.1 Wav2vec2.0 Model The Wav2vec2.0 model is a continuation of the Wav2vec model, both developed by the Artificial department of Facebook (Baevski, Conneau, and Auli 2020). The Wav2vec model was released in 2019, being the first application of a self-supervised, convolutional, pre-trained model for speech recognition purposes. The new approach resulted in an increase in performance, surpassing the then-best model DeepSpeech2 from Baidu (Schneider et al. 2019). Due to the 26 high resource overhead required to train the Wav2vec model and the associated optimization potential, Facebook released the improved Wav2vec2.0 model in late 2020. The Wa2vec2.0 model can be trained in less time, with less data, and achieve better results than its predecessor (Baevski et al. 2020). As mentioned, the Wav2vec2.0 architecture is based on a convolutional neural network (CNN) model, which is a type of model mostly used for finding complex patterns in images (O'Shea and Nash 2015). CNNs belong to the higher-level class of neural networks, which consist of input, hidden, and output layers. Neural networks with multiple hidden layers are categorized as deep learning networks. All of these different layers may contain multiple nodes which can be connected to nodes of the next layer (O'Shea and Nash 2015). Figure 10: CNN architecture in individual steps. The workflow shows how the input is passed through a convolutional layer to extract meaningful features to a pooling layer. There the size is reduced for the fully connected layer to map the features to an output. (Source: Figure created by authors, based on O'Shea and Nash (O'Shea and Nash 2015)). In the case of CNNs, these hidden layers mostly consist of a series of convolutional layers, followed by non-linearity layers, pooling layers, and then fully connected layers (Albawi, Mohammed, and Al-Zawi 2017). During the convolution part, a series of filters, which are generated during the training of the model, is applied to the input data to detect patterns (Krizhevsky, Sutskever, and Hinton 2017). To allow the identification of more complex patterns within the input data, the number of convolutional layers can be increased (Szegedy et al. 2015). In order to reduce the size of the output of the convolution, pooling layers are used. These pooling layers group semantically similar features, thereby improving the efficiency of the model and reducing noise within the input (LeCun, Bengio, and Hinton 2015). Succeeding a number of convolutional and pooling layers are a series of fully connected layers. Fully connected layers map the features from the convolutional layers to the desired output by ensuring that each input from the previous layer influences each node of the next layer (Krizhevsky, Sutskever, and Hinton 2017). A non-linearity layer is appended to the output of the fully connected and convolutional layers, increasing the performance and speed of the model (Krizhevsky, Sutskever, and Hinton 2017; LeCun, Bengio, and Hinton 2015). The Wav2vec2.0 model builds on the CNN structure, with additional components added to specialize it in the downstream task of converting speech to text. In detail, the model consists of three subcomponents, a feature encoder, a transformer module, and a quantization module, each containing multiple kinds of layers (Baevski et al. 2020). The feature encoder is responsible for extracting feature vectors from the raw audio by applying a CNN-based feature extractor to it. The transformer module is composed of multiple layers of self-attention, where each layer learns to capture different aspects of the audio signal. This is done by randomly masking input from the feature encoder and training the transformer to predict the correct output in regard to the quantization module. The quantization module is responsible for predicting the correct phonemes from an audio input. This module uses a Gumbel-softmax, a sampling technique for approximating categorical distributions, to predict which sound input relates to which letter (Jang, Gu, and Poole 2016). This task is performed by learning discrete speech units. These units are codewords sampled from codebooks, which are structures that contain all possible pairs of phenomes. Codewords are then concatenated to form the final speech unit (Jang, Gu, and Poole 2016; Baevski et al. 2020). Figure 11: Wav2vec2.0 architecture setup. The latent speech representation layer extracts feature from the raw audio waveform. With the help of the quantization module and the transformer layer, the masked input parts are predicted. Based on the performance of the contrastive loss function, the weights within the model are adapted. (Source: Figure created by authors, based on Baevski et al. (Baevski et al. 2020)) The model architecture for fine-tuning the model is based on Connectionist Temporal Classification (CTC). The CTC algorithm uses an input sequence to produce the most probable label sequence, which are words or phonemes for ASR. In order to maximize the likelihood of the predicted label sequence, the model's parameters are optimized in CTC to make a probabilistic prediction of the output (Hannun 2017). The pre-training of the Wav2vec2.0 model can be split into two different phases. The first phase of training the model consists of a self-supervised learning approach (Baevski et al. 2020). More particularly, the self-supervised approach is a contrastive task. Contrastive learning allows the model to detect patterns between input features. Although the data is not labeled, the model can find similarities and differences between the audio input (Tian et al. 2020). This enables training models without requiring only labeled data, which is in the area of automatic speech recognition highly infeasible and unpractical as this means that hours of speech data have to be transcribed manually. The data to pre-train the model was 960 hours of unlabeled recording data from the LibriSpeech corpus (Panayotov et al. 2015) and 53.2k hours of data from LibriVox (Kearns 2014). During this part of the training, the model detects and understands high-level patterns, such as extracting phenomes correctly. In addition, a diversity loss is added on top of the contrastive learning to the model during this phase. The diversity loss function is responsible for incentivizing the use of all codewords equally often and preventing the model to only choose from a fraction of codeword and codebook entries (Baevski et al. 2020). The second training phase consists of fine-tuning the model. During this phase, the model is trained with supervised learning. The training data is from the LibriSpeech corpus but with a label. In this phase, the model trains to connect words or phonemes from the input with the associated label (Baevski, Conneau, and Auli 2020). Due to the unique setup of the training architecture, the model achieves a high performance on a cleaned dataset with only 10 minutes of training (Baevski et al. 2020). Since the fine-tuning phase decides which specialization the model has, this phase is modified for the project. The pre-trained model is fine-tuned on the medical data. Thus, the model can transcribe medical data well. 5.1.2 HuBERT Model In 2021 Facebook released another model, designed for automatic speech recognition tasks, named HuBERT (Hidden unit BERT). This model was designed to tackle three challenges of self-supervised models, namely that each speech input contains multiple sound units, there is no pre-training lexicon of input sound units, and sound units have varied lengths without explicit segmentation (Hsu, Bolte, et al. 2021). The HuBERT model addresses these issues by making use of a K-means clustering phase to produce aligned target labels for a BERT-like prediction loss function (Hsu, Tsai, et al. 2021). The BERT model's prediction loss function is a combination of the cross-entropy loss and the next sentence prediction loss. Cross-entropy is used to measure the difference between the predicted output and the true labels. The next sentence prediction loss is an additional component of the BERT model, which is used to 30 measure the relationship between the two words or sentences (De Boer et al. 2005). The HuBERT model applies the prediction loss only over the masked regions, training the model combined on the acoustic and language input (Hsu, Bolte, et al. 2021) (More in section 5.2.2.). The pre-training HuBERT architecture resembles the architecture of Wav2vec2.0. Similarly, HuBERT encodes the waveform audio inputs with a CNN. In addition, a BERT encoder, a projection layer, and a code embedding layer are used (Hsu, Bolte, et al. 2021). The CNN

encoder is responsible for generating continuous speech features from the raw audio input. The speech features are afterward randomly masked and inserted in the BERT encoder to predict pre- determined cluster assignments. This allows the detection of long-range temporal relations in the input data. The HuBERT model can learn both auditory and language models from the continuous inputs because the prediction loss is calculated based solely on the masked regions (Hsu, Bolte, et al. 2021; Anuchitanukul and Specia 2022). Figure 12: HuBERT architecture setup. The latent speech representation layer extracts feature from the raw audio waveform. At the same time, a K-Means algorithm groups similar audio features. Both models combined create the context representation. (Source: Figure created by authors, based on Hsu, Bolte, et al. (Hsu, Bolte, et al. 2021)) Although the pre-training setup of both models shares a similar architecture, both models can be differentiated from one another by the fine-tuning process. The three main differences are that HuBERT uses cross-entropy loss, builds targets via a separate clustering process, and reuses embeddings from the BERT encoder to improve predictions. While Wav2vec2.0 uses the combination of contrastive learning and diversity loss, HuBERT utilizes cross-entropy loss. Since a cross-entropy loss function lacks performance stability for a limited labeled dataset and has a poor generalization ability, these issues are addressed by other features of the HuBERT model and by pre-training the model with unlabeled data (Gunel et al. 2020). As described in the section above, Wav2vec2.0 uses the Gumbel softmax function to learn how to predict its targets during training. HuBERT utilizes the k-means algorithm to create separate targets. This method increases generalization by increasing the number of targets and varying more often between them, decreases the speed to analyze similar input data, and effectively learns the semantic relationship between input sounds in an unsupervised manner (Y. Wang, Boumadane, and Heba 2021; Baevski et al. 2020; Hsu, Bolte, et al. 2021). The third differentiation between both models is the ability of the HuBERT model to reuse the embeddings from the BERT encoder, while the Wav2vec2.0 model only uses the outputs of the CNN for the quantization module. Since the BERT encoder is able to capture contextual information and capture long-term dependencies in the text, it helps the model to better understand the semantic meaning of the text and to better distinguish between semantically similar words. Subsequently, this intermediate step leads to a generation of more accurate predictions (Y. Wang, Boumadane, and Heba 2021; Baevski et al. 2020; Hsu, Tsai, et al. 2021). The process of pre-training the HuBERT model is divided into two parts. The first step consists of extracting the hidden units from the raw audio input. This is done by fragmenting the audio into 25 milliseconds long segments, which are used as inputs for the K-means algorithm. The K- means algorithm divides the fragments into K clusters by analyzing two input features. The first input feature is the MFCC of each audio fragment. The coefficients provide an efficient indication of the rough cluster the audio fragment belongs to. Afterward, the model reuses the representations of the BERT encoder from the previous iteration to categorize the audio fragments more granular and precise. All audio frames associated with each recognized cluster 32 will subsequently be given the unit label for the hidden unit that was created for it. After mapping each hidden unit to its appropriate embedding vector, predictions can be made in the second phase (Hsu, Bolte, et al. 2021). The second step involves training the model with the aim of a masked language model (more in section 5.2). The HuBERT model masks half of the transformer encoder input features, training the model to predict the targets for the masked features. To receive the prediction logs, which track changes in the model over time to assess the performance, the cosine similarity is computed for the predicted masked feature and every hidden unit embedding from all feasible hidden units. The cross-entropy loss function penalizes the wrong predictions, thereby providing feedback to the model to predict more accurately for further predictions. To increase the performance when using noisy labels, the cross-entropy loss function is only applied to the masked features, as this returns the best results (Hsu, Bolte, et al. 2021). 5.1.3 Model Fine-tuning The following paragraphs describe how the data had to be adjusted for the training and how the model was set up to achieve optimal results. The fine-tuning process is a complex process, as many parameters must be optimized, often in combination with each other. To train the model, the MSTI dataset is split into a train, evaluation, and test set with a size of 52.5%, 22.5%, and 25%. In addition, files that are shorter than 1 second and longer than 10 seconds are disregarded from the train set, as too-short audio files do not convey the same sentence structure, and too-long sentences result in memory issues during training. Subsequently, all audio files are resampled to 16kHz to match the preconditions for the Wav2vec2.0 and HuBERT models. Simultaneously, all letters are converted to lowercase, and the following special characters are removed from the various datasets “,”, “?”, “.”, “!”, “-”, “;” “:”, “[” and “]”. Since all these characters are silent in pronunciation, their removal ensures that the model is not evaluated on training a character which cannot be extracted from the acoustic 33 input. The apostrophe is not removed, although it is silent, since it can alter the meaning of the word, e.g., the words “its” and “it’s” have different definitions. The unique remaining letters denote the categories to be predicted by the model for each letter. The 38 final categories include the letters of the English alphabet, in addition to the numbers 0,1,2,3,5 and 7, the space character, the characters “`”, “\n”, and “``” and the additionally added special tokens “PAD” and “UNK”. In addition, each category gets a number assigned from 0 to 37. The “PAD” token is included as some input files require padding to compensate for their shortness, while the “UNK” token represents the unknown category, meaning the model cannot predict a known category. The table below shows the mapping of the keys to their numerical value. Key \n | ' 0 1 2 3 5 7 ` a b c d e f g h i Value 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 Key j k l m n o p q r s t u v w x y z [UNK] [PAD] Value 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 Table 1: Speech to text vocabulary mapping file. The "key" row represents the alphanumeric values found in the transcription column the "value" row shows the value they are assigned to for the processor. (Source: Table created by authors). For fine-tuning both models, the Hugging Face API was used. Hugging Face consists of multiple repositories and python packages which are designed for simplifying deep learning down-stream tasks. Their transformers library is used to load the models and perform the necessary steps to train and evaluate the model. Both models use the same processor to encode and decode the audio data and references. The processor for this task is downloaded from the Hugging Face library and applied to the raw audio files and their corresponding labels (Wolf et al. 2020). Every character in the label is transformed into its corresponding number. An example of this can be seen in table 2 below for the sentence, “my joints feel swollen”. Original word m y | j o i n t s | f e e l | s w o l l e n Processed 22 34 1 19 24 18 23 29 28 1 15 14 14 21 1 28 32 24 21 21 14 23 word Table 2: Example of a processed word. Each letter in the sentence “my joints feel swollen” is represented by its number, as seen in the mapping table (Table 1). (Source: Table created by authors). Since the length of the input samples for the model can vary, it is necessary to pad all samples 34 to the longest one of the respective batches. This is achieved by using a custom padding class, which finds the longest input sample regarding audio and label length. The remaining samples of the batch are padded from the right side until the length matches both features. The “UNK” token is used for this case. The pre-trained Wav2vec2.0 base model is downloaded from Facebook’s Hugging Face page (https://huggingface.co/facebook/wav2vec2-base). The base model is used for this study as the large model produced memory errors during training. The pre-trained HuBERT model was also downloaded from Facebook’s Hugging Face page (https://huggingface.co/facebook/hubert- large-ll60k). In this case, the large model did not produce any errors during training. Hugging Face provides an API to use the open-source model to fine-tune the pre-trained model further. This API includes a trainer class, which accepts the necessary input parameter to fine-tune the model. The trainer trains the model on the train dataset and evaluates the evaluation dataset with the provided evaluation function. To monitor the performance of the train dataset, the evaluation metrics (see section 5.1.4 for more detail) are computed after each epoch on the validation dataset. The metrics are calculated on the validation dataset to provide unbiased results in case the model is overfitted by the train data. For the fine-tuning of both models, the training strategy, floating point 16, and gradient checkpointing hyperparameters were set as constants. The training strategy is set to “epoch”. The floating point 16 boolean is set to true to reduce the memory size. This means that the precision during training is reduced from 32-bit floating points to 16-bit floating points. Setting the gradient checkpointing boolean to true also decreases the memory usage but increases the computation time. In addition, the feature encoder, which contains the CNNs of the model, is frozen. This is necessary as the pre-trained models already have the capability to extract meaningful features from the raw audio, and this step does not have to be fine-tuned (Baevski et 35 al. 2020). Since both models save checkpoints of the current state of the model after each epoch, the

last checkpoint was utilized for restarting the training. This allows restarting the training process where it ended and not having to compute all prior epochs as well. This is especially useful when training cannot finish due to memory errors. In addition to the hyperparameter values, a custom earlystopping callback is included in the training process. This is useful when a model improves its performance based on the train dataset, but the performance on the validation dataset stagnates or decreases. This is a typical indication of overfitting, which can be prevented with earlystopping. The earlystopping callback recognizes that the validation metrics do not improve, thus stopping the training process. For the speech-to-text models, the WER and CER are constant at 0 for the first epochs, as nothing is predicted correctly. To only start the earlystopping callback when the WER and CER have improved, the earlystopping callback was modified to only be called after 15 epochs (Prechelt 1998). Both models were trained on an A4000 machine, which provides 16 GB of GPU, and 8 vCPUs for training. The average GPU occupancy rate was over 95%. To optimize the models, the hyperparameters of both models are individually adjusted. For both models, the following hyperparameters were tuned and based on academic references (Mohamed and Aly 2021; Peng and Harwath 2022; Wang, Boumadane, and Heba 2021; Z.-C. Chen et al. 2022; Baevski et al. 2020; Hsu, Bolte, et al. 2021) and a trial-and-error approach. The following values are set for each model respectively. Hyperparameter Wav2vec2.0 HuBERT Explanation value value Batch size 32 16 This number expresses how many samples are propagated through the network at a time. A small batch size requires less memory, and the training is faster. A large batch size increases the accuracy of 36 Gradient accumulation steps Number of training epochs Learning rate Weight decay Warmup steps 2 30 1e-4 0.005 1000 the gradient (Smith et al. 2017). 2 This number defines how often the gradients and loss are calculated without the model parameters being updated. Afterward, the parameters are updated on the cumulated gradient of consecutive batches. The higher the number, the less likely it is to run out of memory, but also that the performance decreases (Hermans, Spanakis, and Möckel 2017). 25 By defining the number of training epochs, the length of the training is subsequently set (Osawa et al. 2018). 5e-4 This value controls how large of a step in the direction of the negative gradient the model's parameters must be updated in response to the outcome of the loss function each time. A small learning rate takes a long time to converge, while a too-high learning rate overjumps minima (Zeiler 2012). 0.005 By choosing a higher value, it can suppress some of the impacts of static noise on the targets and any irrelevant components of the weight vector, thereby generalizing the model (Krogh and Hertz 1991). 500 The number of steps at the beginning of the training at which the learning rate is much than the one which is set (Izsak, Berchansky, and Levy 2021). Table 3: Hyperparameters for speech to text models. Each hyperparameter is assigned the optimal value for fine-tuning the respective model and a brief explanation. (Source: Table created by authors). 5.1.4 Results and Evaluation This section covers the evaluation of both models on a set of different metrics. The evaluation is run on the test dataset to prevent the results from having a bias. The evaluation of the metrics leads to the selection of the model that will be used in the application. A total of four metrics are used to evaluate the two models: Word Error Rate (WER), Character Error Rate (CER), prediction speed, and model size. The WER and CER are very common evaluation metrics in the speech to text area as they determine how many of the words or characters are correctly transcribed in comparison to the reference value. Since both metrics are error rates, a lower value is more favorable as this means that fewer words or characters are predicted incorrectly, with 0 being the perfect score. The formula for the calculation of the WER is depicted below. $R+C+I\ WCR = M$ Equation 1: Word Error Rate (WER) formula The S refers to the number of substitutions, the D stands for the number of deletions, the I represents the number of insertions, and the N denotes the number of words in the reference. The formula for the CER is, on a high level, identical to the WER formula, although they carry different meanings. $R + C + I\ CCR = M$ Equation 2: Character Error Rate (CER) formula For the CER, the letters are counted on a character basis, and the M stands for the number of characters in the reference. For both the WER and CER, the values can be above the range of 0 to 1 if the prediction contains more words or letters respectively than the reference. For speech- to-text use cases, the CER value is often lower than the WER due to the facts of homophones. This implies that the model predicts words that sound similar to the reference but are spelled differently, although often in just a few letters. Since the WER would mark a word as an S which only has one incorrect letter, but the CER would still account that the rest of the word is predicted correctly, the CER value has a better score (Morris, Maier, and Green 2004). In table 12 in the appendix, an example of predictions and references with their CER and WER is provided to illustrate the different computations. The prediction speed is an important metric for this project as this implies how long a user of the application must wait for their input to be analyzed and an output generated. This is important since users are accustomed to receiving quick responses to their queries online and otherwise lose interest in the application. The time required for each model to compute the same 1,665 predictions can be seen in table 4. The size of the model is important for this project because the application is hosted on a free 38 platform, which only provides a certain amount of memory and has limited computing power (see section 7.1.). A larger model ties up more resources which can cause a worse user experience as the application starts to lag and the application has a higher failure rate. All four evaluation metrics are important for assessing the models, although the results of each metric are not weighted the same. The WER is the most important value, followed by the CER and the speed of the model. The accuracy of the predictions has a crucial impact on the performance of the natural language processing and classification model, which is why the outcome of the transcription is of utter importance. A better WER score can relativize a slightly longer waiting time for the user. The assessment of both models is performed for each evaluation metric and produces the following results. WER CER Speed Size (1665 samples) Wav2vec2.0 11.0% 4.7% 490 seconds 0.38 GB HuBERT 10.6% 5.3% 630 seconds 1.26 GB Table 4: Performance evaluation of both ASR models. Both models perform similarly well for the error rates, but the Wav2vec2.0 model outperforms the HuBERT model in terms of speed and size. (Source: Table created by authors). The above table shows that the HuBERT model is performing better regarding the WER but is outperformed in all other categories by the Wav2vec2.0 model. Especially interesting is the fact that the Wav2vec2.0 model has a lower CER value but a higher WER value than the HuBERT model. This means that the Wav2vec2.0 model misspells words more frequently than the other model. In addition, the fact that the Wav2vec2.0 is the faster and smaller of the two are two advantages for the implementation based on the resource limitations of the project. As both models are performing similarly in terms of WER, both models were tested in the Streamlit application. This resulted in favoring the Wav2vec2.0 model as this one provided a smoother interaction. The predictions were received noticeably faster, and the application did not encounter any disturbances. The HuBERT model, apart from occasionally running very well, 39 often had the issue of running out of memory. This resulted in having to restart the application server in the backend. Since this does not provide any benefit to a user, even as the WER score is better, the Wav2vec2.0 model was chosen for implementation in the final application pipeline. In conclusion, this section introduced an approach for fine-tuning state-of-the-art automatic speech recognition models on medical speech data. The models chosen, Wav2vec2.0 and HuBERT, both performed very well on the little amount of speech data which was available for this study and scored Word Error Rates of 11% and 10.6%. This underlines the versatility of both models and proves the high performance of the approach used in this study. Finally, the Wav2vec2.0 model was chosen for the implementation in the web application as it has a smaller size, predicts faster, and has a similar Word Error Rate and Character Error Rate as the HuBERT model. 5.2 Natural Language Processing (Tara-Sophia Tumbraegel, 48333) Medical errors cost $17 billion to $29 billion per year in the United States. These costs can, for instance, result from prolonged hospital stays in intensive care units or falsely described prescription drugs (Tanne 2008). In total, administrative costs, such as medical billing, account for 25% of the $200 billion in hospital spending. Given the increasing number of patient records, the manual assignment of codes is too costly, time-consuming, and error-prone (Singh et al. 2020). These problems are addressed by Natural language processing (NLP) which can be used to extract meaningful insights from a wide range of medical sources, analyze patient data, reduce errors, increase efficiency, save costs, and provide a better patient experience. To achieve this, artificial intelligence and machine learning techniques are being used to make NLP applications an indispensable tool for modern medical professionals (Singh et al. 2020, 1) This section focuses on how to derive the appropriate keywords from a medical text. Keyword extraction is important for medical records because it allows healthcare providers to find relevant information quickly and accurately in large amounts of medical data. NLP also has the

potential 40 to directly assign a text to a specific disease, as many other approaches have done before (Nuthakki et al. 2019). Since our goal is to increase the transparency of the process hence how the classification model came up with the predicted department, we felt it was important to identify the relevant keywords that the model uses to assign the data to a specific disease as a first step. Furthermore, the increased transparency allows that the model can later be easily used by others to develop it further or use it differently. In the following, at first, an explanation of the development of NLP is given, followed by an explanation of the models used. Afterward, the pre-trained NLP model Bio_ClinicalBERT is instantiated. For fine-tuning, we then use two different approaches – Masked language modeling and Sequence classification. Both models are first tested on the smaller MT Samples dataset and then after evaluating their performance on the MIMIC-III dataset. Their embeddings are stored and used in the next step for the downstream task. Finally, for the downstream task, the model KeyBERT is used to extract the important keywords that will be used for the classification model. Here again, KeyBERT is tested on MT Samples, and then only the better-performing model is used for MIMIC-III. 5.2.1 Current State and Innovations The development of natural language processing started in 1950. At that time, rule-based techniques were used, which were mostly concentrated on machine translation. It involves a person who hand-crafts a system of rules based on linguistic and grammar structures (Dorash 2017). The next innovation was made in 1990 with statistical learning techniques, progressing to deep learning methods in 2012 (Avasthi 2021). Statistical learning techniques are algorithms that learn to understand the language without being explicitly told. They are based on statistical functions (Dorash 2017). Afterward, with the discovery of recurrent neural networks as well as convolutional networks, the performance of NLP tasks increased drastically. They accounted for 41 the majority of NLP tasks up to 2017 (Lutkevich 2020). However, those models, such as Word2vec and GloVe, are only capable of reading text sequentially - left to right or right to left - and expressing all possible meanings of a word as a single vector. Its limitation is that the model does not catch the word's full meaning and cannot disambiguate word sense based on surroundings. An example of that would be catching the different meanings of address in “they address a problem” and “the address is” (Devlin et al. 2019). Furthermore, traditional models require large datasets of labeled data, whereas labeled datasets are often not accessible or need a huge amount of time to create. In 2017 Google eliminated these limitations with its neural network architecture based on a self-attention mechanism called Transformers (Uszkoreit 2017). Transformers is a deep learning model where every output element is connected to every input element, and the weightings between them are dynamically calculated based on their connection (Wolf et al. 2020). It enables data to be processed in any order, therefore, can read the text in both directions. This capability is known as bidirectionality. Additionally, the model can process the entire input as a sequence instead of processing it token by token. Thus, the model can be accelerated by GPUs, enabling training on larger amounts of data than ever before, revolutionizing the performance of NLP tasks (Joshi 2020). Moreover, the model can learn on huge unlabeled datasets eliminating the need for labeled data. The ability to analyze big datasets facilitated the creation of pre-trained models like BERT, which was trained on massive amounts of unlabeled language for multiple days. These models can then be fine-tuned for personalized tasks or used for different tasks with transfer learning (Devlin et al. 2019). According to research scientists in the field, ambiguity posed the greatest difficulty to traditional NLP techniques. BERT, instead, can parse text using common sense that is largely human-like by using the surrounding text to establish context. This distinguishes BERT from previous models, which are limited when interpreting context and polysemous words (Devlin et al. 2019, 2). Due to the revolutionizing performance of BERT in the NLP area, multiple variations, 42 editions, and further improvements of BERT have been developed. For instance, RoBERTa or ALBERT leverage the same architecture of BERT but were trained on different datasets or tasks. 5.2.2 BERT Model In 2018, Google published the open-source machine learning framework BERT, which is used for NLP and based on Transformers. BERT stands for Bidirectional Encoder Representation. Bert was initially trained on a Wikipedia and Google Books dataset containing 3.3 billion words and lasted four days of training on 64 TPUs (Muller 2022). Training the model on large datasets allows them to recognize patterns and extract meaningful information from the text. The use of pre-trained models is very popular because it is very time and resource-consuming to train an NLP model from scratch. Another advantage is that pre-trained models offer a significant accuracy and speed advantage over training from scratch, especially for large, complex datasets (Pestian et al. 2007). Before feeding the input to BERT, the input is converted into embeddings using three embedding layers. The first one is called token embedding, where each word in a sentence is a token in a list. There are also special tokens like the classification token CLS which is at the beginning of a sequence, and the separation token SEP, which is at the end of every sentence. Afterward, the tokens are converted to embeddings, meaning that each word is represented in a vector of size (30522, 768). The first dimension is the vocabulary dimension. BERT models have a pre-trained vocabulary with different sizes of vocabulary files. Our model has over 30,000 vocabularies. The second dimension is the embedding dimension, meaning the number of features which are used to represent a word. 768 are the number of features that have been created in the 768 hidden layers of BERT (Pestian et al. 2007). The embedding for a given token will capture in numerical form the meaning of that token in context to other tokens. The Token Id is assigned to each token and represents its index in the defined vocabulary (Pestian et al. 2007). The next layer is called Segment embedding; it is used to distinguish between two 43 sentences. The segment embedding layer returns the segment Id, which is used to distinguish between different sentences (Biseda et al. 2020). The third embedding, called Position embedding, is necessary to show the token position within the sequence. It is used to indicate that similar words don’t need to have the same output representations. For example, “The coding language python is efficient” and “Python, the snake, is dangerous” - due to the positional embedding, BERT can differentiate the meaning. The positional embedding size, in our case, is represented as (512, 768) since the model uses the input sequences sequential characters and learns a vector representation for each point. The first row represents the vector representation of the word in the first position, the second row represents any word in the second position, and so on. Overall, BERT has a limitation of 512 characters (Narein 2021). Input [CLS] I am sick [SEP] I have fever [SEP] Token Embedding Segment Embedding Position Embedding ECLS EI + + EA EA + E0 + E1 Eam ESick + + EA EA + + E2 E3 ESEP EI + + EA EB + + E4 E5 EHave Efever ESEP + EB + E6 + EB + E7 + EB + E8 Figure 13: BERT’s three embedding layers. The first layer creates a token for every word, as well as for the special tokens. The second layer returns a 0 or 1. 0 if it belongs to the first sentence and 1 if it belongs to the second sentence. The position embeddings, at last, sequentially count the position of each element. (Source: Figure created by authors, based on Narein (Narein 2021)). BERT has been pre-trained on two processes simultaneously, Masked Language Modelling and Next Sentence prediction (Tang et al. 2019). Masked Language Modeling forces bidirectional learning from the text by randomly masking about 15% of the tokenized words in a sentence. This forces BERT to bidirectionally use the words on either side of the masked word to predict the hidden word based on the content alone. Thus, BERT is required to learn the meaning of the word within the context of the text instead of learning the meaning of the word independently of its context. This is important because the meaning of a word can often change in a sentence (Tang et al. 2019). Figure 14: BERTs Masked Language Modelling. BERT randomly masks 15% of the words and tries to find the right word with the context of the other words. From all its possibilities, the one with the highest probability is chosen. (Source: Figure created by authors, based on Gurevych and Reimers (Gurevych and Reimers 2019)). The objective of Next Sentence Prediction training is to learn about relationships between sentences, whether they have a logical, sequential connection or whether the relation is purely random. Many downstream tasks are based on understanding the relationship between sentences, which is not captured by traditional language modeling. BERT accomplishes this by predicting whether a given sentence will follow a preceding one. Hence 50% of the right sentences are paired with random sentences during training (Devlin et al. 2019). SENTENCE A The boy is sick. The boy is sick. SENTENCE B He needs to go to the doctor. Dogs are animals. LABEL Is Next Sentence Not Next Sentence Figure 15: Next sentence prediction. The algorithm pairs 50% of the sentence randomly and needs to understand if the context of the second sentence fits the previous one. (Source: Figure created by authors). Overall the model tries to improve and learn by reducing its losses which are calculated as the sum of the mean masked language model likelihood and the mean next sentence prediction likelihood (Devlin et al. 2019). The architecture of BERT is called Transformers. It is responsible for efficient parallelized training. This results in the

ability to train on millions of data points in a relatively short amount 45 of time, therefore being uniquely suited for unsupervised training, eliminating the need for big, labeled datasets. Transformers employ an attention mechanism to notice how words relate to one another (Wolf et al. 2020). It produces various weights that indicate which words in a sentence are most important for the next step in the process. This is done by successively processing an input through a stack of transformer layers, commonly referred to as an encoder (Devlin et al. 2019). In more detail, BERT, in its base size, consists of 12 attention heads, 12 layers of transformers blocks, and 768 hidden layers. Transformer layers are the number of transformer blocks that convert a sequence of word representations into a sequence of contextualized words. Figure 33 in the appendix shows the information flow of a word in BERT. It starts with its embeddings representations from the embedding layer (Seth 2019). In every layer, there is multi- headed attention computation on the word representation of the previous layer. Each layer consists of mathematical functions that assign weights to words to achieve the desired result (Kumar 2021). A weight is a scalar that is multiplied by the input. The 12 layers are stacked on top of each other. The output of one layer is the input of the next layer, and so on (Seth 2019). the BERT base model has 12 layers a word token will have 12 intermediate representations, each having the same size. 5.2.3 Medical Corpus BERT There are different variants of the BERT model depending on which datasets or domains it was fine-tuned on, for instance, the medical domain. Considering that domain-specific models outperform non-domain-specific models on medical texts - this work focuses only on a BERT model trained on a medical corpus (Alsentzer et al. 2019). Compared to conventional models, BERT is more profound and contains many more parameters, which allows it to process greater representational power. Compared to other models based on Transformers, BERT has been found superior, especially in the clinical domain, which is why this approach only examines Bio_ClinicalBERT (Si et al., 2019). The necessity for specific clinical models trained on medical records is driven by the linguistic discrepancies between clinical text/health records' linguistic features and common English linguistic characteristics (Biseda et al. 2020). Since healthcare data is largely unstructured, a large amount is unused, as traditional machine learning models often require high-quality annotated datasets. As they are not available in the healthcare industry, they are minimally explored in this specific text type (Alsentzer et al. 2019; Rasmy et al. 2021). Emily Alsentzer created the first clinical BERT called Bio_ClinicalBERT, on which the models from this paper are built on. It is based on the BioBERT (Bidirectional Encoder Representations from Transformers for Biomedical Text Mining) framework, initialized with the original BioBERT weights but trained on the full MIMIC-III database with 2 million notes (Alsentzer et al. 2019). With almost the same architecture, Bio_ClinicalBERT largely outperforms BERT on biomedical corpora (Lee et al. 2020). For training Bio_ClinicalBERT, the necessary computational resources were significant. The training took 17 days of computational runtime with a single GeForce GTX TITAN X 12 GB GPU power (Alsentzer et al. 2019). It outperforms the conventional BERT as well as BioBERT. 5.2.4 Bio_ClinicalBERT Fine-tuning This approach fine-tunes the Bio_ClinicalBERT model on a small extract from the discharge summaries. The reason for this is that the discharge summaries are the entries with the most useful patient information. Since the model's runtime increases exponentially when working with long text, we focused only on the subsection with the current illnesses and symptoms. Also, BERT has a token limitation of 512, whereas the length of the discharge summary is much greater (Hou et al. 2022). Before starting the training, the data needs to be preprocessed, as explained in the previous data exploration chapter 3.2.3. Afterward, the sentences are grouped into larger chunks and passed to the BERT model to generate embeddings. For training, the initial parameters of 47 Bio_ClinicalBERT, including its pre-trained weights, vocabulary - and configurations file, were loaded. For fine-tuning, we choose as a first approach Masked Language Modelling (MLM) and as a second approach sequence classification modeling. MLM is a type of unsupervised learning in which the model is trained by randomly masking some words in a sentence and must predict the masked words. The training process was done in accordance with the original work, where 15% of the words were randomly masked. Research states that it is very useful for fine- tuning for multiple reasons, like, for instance, to shift model distribution to the domain-specific. Such a basis creates a strong basis for downstream tasks (Gugger et al. 2020). The second fine-tuning model is called Sequence Classification Model (SCM) and is a supervised learning task that predicts a class label for a sequence, thus can be used to classify text. It has an additional linear layer on top of BERT, which classifies a sequence into a label. In the case of the MT Samples dataset, the labels are diseases, while the labels are the departments for the MIMIC-III dataset. The reason we are considering the SCM is that recent research considers keyword extraction as a sequence labeling task (Çelikten, Ugur, and Bulut 2021). When contextual embeddings are used together with the sequence labeling approach, these models have shown state-of-the-art performance. Furthermore, research provides that these models have a significant performance on keyword extraction since sequence labeling captures long-term semantic dependencies in a document using the whole document information (Çelikten, Ugur, and Bulut 2021). Our approach will compare both performances and chooses the model which performs best. The best-performing model, either the masked language model or the sequence classification model, is then embedded in KeyBERT on the MIMIC III dataset. The implementation was performed using Hugging Face. As explained in the introduction, Hugging Face is an architecture that provides a wide range of pre-trained models for NLP. It is used to train and fine-tune these models on a wide range of tasks like text classification and question answering. Initially, we started by loading the datasets with the dataset library from 48 Hugging Face. The dataset library is created for NLP purposes when working with big datasets. It uses powerful data processing methods backed by the Apache arrow format. It is able to process large datasets with zero-copy reads and without any memory constraints for optimal speed and efficiency (Lhoest et al. 2021). After loading the data, the next step is to divide it into train and test datasets. Since the large MIMIC-III dataset was not available until a later stage of our thesis, the models were first trained on the MT Samples dataset and then in a later step on the MIMIC-III dataset. For MT Samples, a total of 3,100 reports with 11 different classes were analyzed. Of the rounded 3,100 reports, 2,800 were used to train the model, and 300 were used for evaluation. The training and testing data were randomly split. The MIMIC-III dataset includes rounded around 58,000 reports with 16 classes, of which 10% are used as test data. These datasets are then tokenized, allowing the model to understand text data by transforming it into numerical data. The tokenization and embedding steps are the three embedding layers, as explained in the embedding chapter. In the next step, the trainer class from Hugging Face is used. The training procedure, including setting hyperparameters, was followed by the procedure from various papers (Biseda et al. 2020; Devlin et al. 2019). While instantiating the class, the following training arguments have been customized. Parameter Epochs Training Batch Testing Batch Weight Decay Value 10 for MT Samples / 4 for MIMIC-III 4 4 0.01 Explanation Epoch means one pass through the entire training dataset. Due to memory constraints as well as runtime limitations, the epochs needed to be reduced for MIMIC-III. Batch size is the number of individual pieces of data that are processed together at once. The size of the batch affects the accuracy and speed of the training, as larger batches tend to provide more accurate results but take longer time to process (Brownlee 2022). Due to memory and runtime constraints, the number needed to be smaller than the original 8 proposed in other papers. Due to memory and runtime constraints, the number needed to be smaller than the original 8 proposed in other papers. The optimizer weight decay will add a penalty term to the loss function that will shrink the weights of the model toward zero. It Learning Rate Evaluation Strategy Metrics for the best model 2e-5 Epoch Loss was set to in regard to the original Bert paper. The learning rate was set following the process of the original paper. The evaluation strategy classifies how often the model will be evaluated during training. In this case, the model will be evaluated after each epoch The parameter "metric for the best model" was set to the lowest loss. Table 5: Hyperparameter of Sequence classification model and masked language model. The table shows the hyperparameter name, the value, and an explanation if necessary. (Source: Table created by authors). In fine-tuning, the model tries to minimize the loss function. The loss is defined as the difference between the predicted value and the actual value. Hence the smaller the loss, the better. In MLM, the loss function is used to measure how well the model predicts the masked words, while for SCM it measures how well the model is able to classify the right label. It is calculated by comparing the predicted words/labels to the actual words/labels. Both models use the loss function cross entropy which measures the distance between the predicted label/word and the true label/word (C.-S. Wu et al. 2020). Based on this loss, the required changes are calculated and the weights of the model are optimized (Briggs 2021). Since we trained four times, each model

on each dataset, the figures 34-37 in the appendix show four loss functions. The main finding is that the loss decreases for all models, becoming flattered at different points. Both models flatten out faster for the MT Samples data set. A decreasing loss function that initially decreases steeply and then flattens as it approaches the minimum is called a convex loss function. This is an indication that the model is converging to a good result. The reason is that the loss function decreases as the model is trained since the model learns from its mistakes and becomes better at predicting the correct outcome (Hajra 2019). Other attributes of the trainer class are the evaluation function to compute metrics. It allows computing user-specified metrics. For both model the evaluation metric is accuracy. For the sequence classification the meaning is how accurately the model can predict from the text the correct label, for MLM the meaning is how accurately the model can predict the right word for the masked words (Dauria 2019). For our final dataset MIMIC-III and our final model SCM we can see an accuracy of over 73%. Considering that the dataset has 17 different classes, an accuracy of 73% means that the text can predict 73% of the labels correctly, which is a great starting point. To further improve the scores, hyperparameter training could be useful, or earlystopping, to stop at the highest accuracy point. Since we have already achieved very good results, we decided to use the models only with the current fine-tuning for the following downstream task. In the next step, the fine-tuned models are implemented in the KeyBERT model to extract the right keywords from MT Samples. Based on their performance, the better model is used in KeyBERT for the MIMIC-III dataset. 5.2.5 Results and Evaluation with KeyBERT Keyword extraction became an indispensable tool for natural language processing and a common downstream task. The difficulty in extracting keywords in the biomedical field lies in the specific terminologies of this domain, such as anatomical information, disease terms, medical terms used in diagnosis, and treatment methods (Çelikten, Ugur, and Bulut 2021). Due to the rapidly growing amount of biomedical literature, it becomes essential to develop automated methods for extracting key phrases representing the main points of the text from the biomedical literature. Despite this need, few researchers have focused on extracting key phrases from biomedical texts (Çelikten, Ugur, and Bulut 2021). Keyword extraction approaches vary from supervised to unsupervised methods, where supervised techniques require a substantial amount of labeled data. Furthermore, supervised methods performed poorly outside the domain represented in the training corpus. This is a big issue since the domain of new documents may not be known at all. Unsupervised key phrase extraction address information constraints by relying on in-corpus statistical information, e.g., the inverse document frequency of the words and the current document (Sushil, Suster, and Daelemans 2021). Past work focuses on unsupervised techniques, 51 employing statistical methods like RAKE (Rose et al. 2010) and YAKE (Campos et al. 2020), graph-based methods like TextRank (Mihalcea and Tarau 2004), or embedded based like KeyBERT (Grootendorst 2020). Out of all methodologies, KeyBERT has been widely accepted for its best performance. KeyBERT is an easy-to-use keyword extraction technique that leverages pre-trained BERT embeddings and is built on a neural network architecture. The keyword extraction is done by finding the sub-phrases/candidates in a document which are most similar to the document itself. The similarity is calculated using the cosine similarity between vectors. It is assumed that the most similar candidates of the document are good key phrases. The downside of the cosine similarity is that all key phrases can be very similar to each other. Therefore, there are two different algorithms to diversify the result, max sum similarity and maximal marginal relevance (Grootendorst 2021). The result is a balance between the accuracy of the keywords and the diversity between them. The outcome of the KeyBERT model is a list of the important keywords and their relevance score. The words are organized based on their score, starting with the most important one. For preprocessing, the size of the transcription column has to be reduced to 512 characters, as transformer models have a token limit of 512 characters (Tang et al. 2019). According to our own research, as well as in line with other papers using MIMIC-III and MT Samples dataset - the most important words are at the beginning of the transcription column. Thus, the information loss is very small (Devlin et al. 2019). Afterward, the data is tokenized with the tokenizer of the before pre-trained models for KeyBERT to understand the textual data. After preprocessing, KeyBERT is instantiated with the embedding model of the two pre-trained MLM and SCM. The following table shows the hyperparameters which are chosen for the model. Hyperparameter Value MMR TRUE Explanation The second method to diversify the results is called maximal margin relevance. It tries to minimize redundancy and maximize the diversity Max Sum Number candidates Top n Diversity N_gram TRUE 40% of the words in the text but a maximum of 35 words 60% of the number of candidates 0.5 1 of results in text. It starts by selecting keywords that are most similar to the document, then literately selecting new candidates that are similar to the document and not similar to the already selected keywords/key phrases (Grootendorst 2021). The formula is described in section 5.3.4 The max sum similarity calculates the maximum distance between pairs of data. For this approach, the goal is to maximize candidate similarity to the document whilst minimizing the similarity between candidates (Grootendorst 2021). Due to runtime issues, we could only consider a maximum of 35 keywords. If a low number of candidates is set, the result will be similar to the original cosine similarity method. However, setting a high number creates more diverse key phrases. Top n is the number of most important keywords we want to return from the number of candidates (Grootendorst 2021). Diversity means how different the keywords should be from each other. The higher the diversity, the more different the keywords are (Grootendorst 2021). The default value is 0.7. This approach uses 0.5 since the keywords should be allowed to be slightly similar to each other (e.g., "heart failure" and "kidney failure"). N-gram stands for the number of words in a keyword (1 means 1 word, 2 means 2 words, etc.) (Grootendorst 2021). For n_gram, only 1 word is considered due to the fact that run time increases significantly for two or more words. It increased so much that we were not able to run it on MIMIC-III. Table 6: Hyperparameter of KeyBERT. The table shows the name of the hyperparameters, their values, and, if needed an explanation. (Source: Table created by authors). The run time for the models on MT Samples took around 7 hours, while the runtime for MIMIC- III took around 14 hours after reducing ngram from a range of one to three words to only one word. Even though we reduced the size of extracted words to only one single word, we did not see a loss in accuracy in the classification part. Choosing the optimal number of extracted keywords is a balance between runtime and accuracy since fewer words reduce the accuracy of the classification task significantly, with a decrease in runtime. Since our approach is to evaluate the best model on MT Samples, the following two figures focus on the comparison of the MLM in KeyBERT versus the SCM in KeyBERT. For MT Samples, our result is that the average number of extracted words varies from 17 to 19. Furthermore, figure 38 in the appendix shows that both models extract about 30-50% of the same words, with different rankings of importance. This shows that even though both models were trained on the same dataset, they assess different words to be important, which explains the different performance on the classification task. The following word cloud graphic visualizes how often the two models select which keywords. To make it clearer, the word cloud refers only to the category of gynecology. It shows the extracted words based on MLM embeddings on the left and TC embeddings on the right. Both algorithms consider different keywords more important but also extract similar words. We can see the words uterus, ultrasound, or preoperative in both clouds, but they differ significantly in the most frequently extracted words. Figure 16: Word cloud of the extracted Keywords for Figure 17: Word cloud for the extracted Keywords for class Gynecology based on the MLM. The two most class Gynecology based on the SCM. The words common words are "anesthesia", "uterus", and "female", "preoperative", and "anesthesia" occur "postoperative". (Source: Figure created by authors). most often. (Source: Figure created by authors). After looking at the classification result, which is described in the classification part 5.3. we concluded that the SCM outperforms the MLM. Therefore, we are running the KeyBERT model with the SCM embeddings on the MIMIC -III dataset. The table below shows an exemplary final output for the MIMIC-III dataset. Even though we are only extracting single words, in the case of two-word symptoms like “Rip pain”, both words are extracted. Cleaned Text Mimic Extracted Keywords Motorcycle crash left sided rib pain driver helmeted motorcycle crash 'crash', 'helmeted', 'motorcycle', splenic lac released Hospital Hospital transferred area Hit splenic 'driver', 'rib', 'hematoma', 'pain’, hematoma free fluid LUQ pain Month BM flatus sob fevers chills 'released', 'fevers' Table 7: Cleaned Text extract from MIMIC-III with the extracted Keywords. The extracted keywords are the most important words. (Source: Table created by authors). To summarize the entire process of this section, the following figure 18 illustrates the six sequential steps. The individual parts took different amounts of time. While training the MLM and SCM on MT samples took only about an hour each, it increased to 4 hours each for the MIMIC-III dataset. For the downstream task,

keyword extraction for MT samples, each model took about 5 hours, while it took over 14 hours for the MIMIC-III dataset. Apart from the fact that the training took a considerable amount of time, the prediction of a single sentence or two sentences on the website is a matter of seconds and not a concern. Figure 18: Full process of the keyword extraction part. The entire end-to-end process includes six steps from extracting the right keywords from the medical text. (Source: Figure created by authors). Overall, there are several papers on MIMIC-III with excellent results, as explained earlier, but none of them focused on the transparency of the whole process, so a great added value for customers, physicians, and programmers emerges. There were also no other keyword extraction procedures performed on MIMIC-III and MT Samples, making it difficult to compare the models. Most papers focus only on the outcome of classifying the texts of a disease, whereas our focus is on increasing the transparency of the entire finding. The aim is that the customer can better understand how the physician came up with the solution, the physician can judge where errors come from, and new coders can easily reuse our code as they can be guided along the steps, reducing the black box scenario. In the end, we were able to fine-tune two models on two different datasets that showed good learning curves during training. Their embeddings are used 55 in the downstream keyword extraction part for MT Samples, while only the better-performing model SCM is used as an embedding for MIMIC-III. The keyword extraction model extracted valuable and meaningful keywords that provided a good database for classification. # 5.3 Classification (Hannah Petry, 48458) Classification algorithms are essential tools for solving complex problems in today's world by allowing to better diagnose, predict, and prevent diseases. This part of the project proposes an approach to classify a list of symptoms to a medical specialty, and it aims to select the best predictive model for medical specialty detection. Classification is the recognition of classes described by a set of features in a dataset (Maglogiannis 2007). Disease diagnosis is a multi- class classification problem that deals with high-dimensional datasets. Various supervised and unsupervised machine learning methods have been applied by medical academics to detect and predict diseases and medical specialties (Pestian et al. 2007; Chitra and Seenivasagam 2013; Antony et al. 2021). Contrary to other existing symptom checkers, for this project, it was decided to predict medical specialties instead of specific diseases. First, only medical specialties exist as labels for the MT Samples dataset. Secondly, for the MIMIC-III dataset, the titles of the ICD-9 codes are medical terminology and not necessarily of use for the non-professional user. Therefore, concerning this study, it is more helpful to assign symptoms to a medical department that can be consulted or visited by a user. However, the model could also predict ICD-9 codes directly depending on the future use case. As stated in the previous NLP part of this paper, several input data versions for the features were used to test and compare them. The NLP output from the MT Samples dataset is used as an indicator to check the correctness and initial performance of the models. Hence the output from the MIMIC-III dataset is the final version to use. The MT Samples and MIMIC-III datasets receive keywords based on the masked language and the sequence-classified NLP models, respectively. Since the evaluation of the masked language model relies on the performance of 56 the classification, the aim is to train all models with all the input data versions. Therefore, the performance of the different NLP input data versions for all classification models must be compared, and the classification model with the highest performance is chosen. 5.3.1 Performance and Explainability Trade-off There is a trade-off between the accuracy performance and the explainability of a model. The goal is not only to make good predictions for a given user input but also to build trust in the models. Without insight into how a model functions, biases can occur that are so detrimental that they negate the marginal gain in predictive power. To overcome such eventualities, models must be explainable to the user. This is especially important in health care because data is sensitive, and decisions based on the model's statements are highly critical and can entail far-reaching consequences. One solution our study suggests is separating, as mentioned above, the three steps, speech-to- text, NLP, and classification, to show intermediary results. Another aspect is to explain to the user why the model made a prediction so that the user can understand the output and attribute the model's prediction to his input. For the classification task, there are two ways to include this. In addition to the prediction probabilities of a class, a user will receive those symptoms per class from the list of symptoms leading to the prediction of the different classes and a list of the most critical symptoms per class that the model used in the training set for prediction if the model allows it. This helps the user to understand the model's decision. However, to use the tools just mentioned for the model explainability, a trade-off must be accepted at the expense of the model's performance. Some best practices for feature reduction cannot be applied then, for example, principal component analysis (PCA), since the features need to be traced back for those two additional outputs that represent the explainability part of the model. Typically, a best practice for dimensionality reduction and feature selection is PCA. PCA can improve algorithm performance, reduce overfitting, and remove correlated features 57 (Hasan and Abdulazeez 2021). However, after PCA, the original features will become principal components which are not as readable and interpretable and, if not selected with care, can lead to information loss compared to the original features. For the reasons mentioned above, PCA is not included. Generally, it is recommended not to have correlated features in the dataset (Guyon 2008). Those features will not bring additional information but will increase the algorithm's complexity and create multicollinearity. Therefore, one of two features that are very similar and highly correlated, like the uterus and ovary, for example, could be removed to reduce the variance of the model. However, in this use case, if a patient then describes his symptoms and uses the word uterus, which would have been removed instead of the ovary, the model will not recognize this feature or the relationship between the removed feature and the feature still used for training. Consequently, none of the correlated features is removed, but regularization and a penalty term are used for feature selection. 5.3.2 Preprocessing After receiving a CSV file with the relevant columns, namely classes and keywords, some pre- processing on the keywords list column is carried out to further work with the input data and hand it into the machine learning pipeline, a sequence of steps to build and deploy a machine learning model. The pre-processing consists of applying a method that safely evaluates strings containing Python values from unknown sources, here the CSV file, without having to parse the values and applying the join method that returns a string by joining all the elements of the list, separated by the given separator. The data is then split into a train and test set with a test set share of 20%. From a technical perspective concerning readability, usability, and transparency, it is a best practice to use pipelines to spend less time reproducing the same tasks for different values and more time developing new models (Garreta and Moncecchi 2013). A machine learning pipeline 58 can implement and automate processes to speed up, reuse, manage and extend machine learning models and standardize machine learning projects (Hapke and Nelson 2020). In this case, it is especially useful because different classifiers are deployed, but the pre-processing stays the same. The input data will first be passed to the pre-processing block and then to the classification block, which is a neat and representable way of handling a workflow. An exemplary pipeline can be seen in figure 19. Figure 19: Illustration of an exemplary model pipeline showing each step the training data goes through, starting from Count Vectorizing the features to balancing out the data using SMOTE to training the classification model (Source: Figure created by authors). In the prediction part of the project, one can call the different steps of the pipeline individually to access, for example, the model's coefficients or the feature names generated by the vectorizer. Since the data is imbalanced, imbalanced-learn, an open-source MIT-licensed library based on scikit-learn, is used (The imbalanced-learn developers 2022; Anwar et al. 2020). The implemented pipeline consists of a Count Vectorizer for vectorizing the features, a Synthetic Minority Oversampling Technique (SMOTE) for balancing out the dataset, and a classification model for feature extraction and as a classifier, respectively. Count Vectorizing is used as the first part of the pre-processing pipeline because otherwise, the model cannot perform classification algorithmically since it cannot process string-like inputs, which is the output of the NLP task, a list of strings. This list must be represented so that it is understandable to the machine learning classifier. A Count Vectorizer transforms this input into a sparse matrix where each word is represented as a vector by the number of times it occurs (Basarkar 2017). Even after data cleaning, the number of the most frequently occurring classes differs strongly from that of the least frequently occurring classes. To deal with those imbalanced classes, the smote package is used. SMOTE is a standard method for learning from imbalanced data. It creates synthetic samples from the minority class to increase its representation in the dataset. SMOTE creates the synthetic samples based on "interpolation between several minority class instances […] within a defined neighbourhood" (Fernandez et al. 2018). This means it randomly selects a minority class

instance and its nearest neighbors and then combines them to create a new sample. This process is repeated until the desired number of synthetic samples is created. 5.3.3 Classification Models Different supervised classification models are trained and tested for comparison, including Multinomial Logistic Regression, Decision Tree, Random Forest, Support Vector Machine, and eXtreme Gradient Boosting. All those models are set so that they can conduct multi-label classification tasks and are frequently mentioned in the literature (S. Wang et al. 2022; Moral, Nowaczyk, and Pashami 2022). By providing the models with an input list of symptoms, they can predict the probability of belonging to a medical specialty. As an extension to the basic concept of logistic regression, the Multinomial Logistic Regression model (MLR), also called Softmax Regression, provides native support for multiclass classification where there is no natural ordering of the classes. In contrast to the one-vs-rest method, where a single classifier is trained for each class by distinguishing between that class and the other classes using the logistic regression cost function, in the MLR algorithm, multiple classes are supported directly without having to train and combine multiple binary classifiers. In MLR, the loss function is converted to cross-entropy loss, and the probability 60 distribution into a multinomial probability distribution. The model is trained using a dataset with known class labels, and the model outputs a probability for each class. A Softmax function is used to predict a probability for each class, considering the scores for other classes. At the same time, the one-vs-rest method calculates probabilities of classes entirely independently and then just picks the label with the highest score (Rese 2000). The MLR has the advantage of allowing for more accurate prediction since it considers all possible outcomes simultaneously instead of making predictions for each class separately. It minimizes computational complexity by not requiring separate models to be fitted for each class. Additionally, MLR is more resistant to class imbalance than the One vs. Rest method. Moreover, the MLR model's results are simpler to grasp because the coefficients can be directly interpreted as the amount of impact of each feature on the probability of each class. The model is trained using a dataset with known class labels, and the model outputs a probability for each class (Bayaga 2010). Feature importance per class measures how important each feature is for predicting the data point's class label. The feature importance is calculated for each class separately. The coefficients in multinomial logistic regression can be interpreted as for a single unit change in the predictor variable, the log of odds or logits, a logistic transformation of the odds, will change by a factor indicated by the beta coefficient, given that all other variables are held constant (El- Habil 2012). Compared to other models, logistic regression is easy to interpret and very efficient to train. In this study, MLR is the only classifier which can be interpreted through its model coefficients as indicators of feature importance per class, meaning one can show which features from the train set the model used for predicting a specific class (Muslim et al. 2020). The Decision Tree Classifier (DT) is a non-parametric supervised learning method. The tree- structured classifier, as illustrated in figure 20, allows the prediction of the value of a target variable by following the decisions in the tree from the root down to a leaf node. A tree consists 61 of internal decision nodes representing the features of a dataset, branches representing the decision rules where the predictor's value is compared to a trained weight, and finally, the leaf nodes representing the outcome, the decision taken after computing all attributes. Figure 20: Structure of the Decision Tree algorithm starting from the root node until the leaf node. Each node in the tree represents a possible decision, and each branch represents the decision rules. The leaf nodes of the tree are the possible outcomes that the decision tree aims to predict. (Source: Figure created by authors). Scikit-learn uses an improved version of the Classification and Regression Tree (CART) Training Algorithm (Priyam et al. 2013). The CART algorithm splits the training set into two subsets using a single feature k based on a threshold value of this feature. After successfully splitting the training set into two parts, it uses the same logic to split the subsets, then the sub-subsets, and so on, recursively. The split at each node is based on the feature that gives the maximum information gain. It stops recursing once it reaches the maximum depth, which will be defined during hyperparameter finetuning, or if it cannot find a split that will reduce impurity (Aly 2005). Measures that can be used to capture the impurity of a split are the Gini index and the entropy. While Gini measures the impurity of a split, calculating the misclassification probability when randomly selecting a sample from within the node, entropy measures it by calculating the amount of information needed to classify a sample from within the node. The CART algorithm is greedy because it makes an optimum local choice at each node. It searches 62 for an optimum split at the top level and then repeats the process at each level. It does not check whether or not the split will lead to the lowest possible impurity several levels down. One of its advantages is the easy-to-understand concept behind the DT, however, it is prone to overfitting issues which can be resolved by the Random Forest algorithm (Charbuty and Abdulazeez 2021). For DTs, the weighted decrease in node impurity divided by the probability of reaching that node determines a feature's importance. The node probability can be computed by dividing the total number of samples by the number of samples that reach the node. The feature is more significant the higher the value. The importance of a node i in a single decision tree is computed as follows, assuming only two child nodes: $nii = wiCi - wlddr(i)Clddr(i) - wridhr(i)Cridhr(i)$ Equation 3: Node Importance formula With wi being the weighted number of samples reaching node i as a share of all weighted samples, Ci the impurity value of node i and $ieet(i)$ and $rieht(i)$ its respective children nodes (Ronaghan 2019; X. Liu and Bakshi 2018). The overall feature importance of feature i is then calculated as follows: $\sum i{:}nndd\ i\ rnlirr\ nn\ ddarrrd\ nii\ eii = \sum icall\ nnddr\ nii$ Equation 4: Feature importance of feature i formula Random Forests (RF) are a type of ensemble machine-learning algorithm where multiple learners are trained and combined to solve the same problem that can be used for classification tasks capable of identifying nonlinear patterns in the data. The driving principle behind the Random Forest algorithm is to build multiple estimators independently and then average their predictions. Combined learners usually outperform single-base learners because their variance is reduced, which reduces the risk of overfitting the data. The Random Forest algorithm works as a combination of a set of tree-structured classifiers, where each tree as the base classifier is a 63 weak learner producing slightly better than random predictions, and combining the predictions of all the trees creates a strong learner (Y. Liu, Wang, and Zhang 2012). The random forest algorithm can be used for multiclass classification. Each tree in the forest has a unit vote, allocating each input to the most likely class label, and then the predictions of all of the trees are combined to form the final predictions (Chaudhary, Kolhe, and Kamal 2016). The bagging algorithm is called "random" since it randomly selects a subset of the features to use when building each decision tree. Hence, some data could be selected more than once, and other is wholly neglected, which increases the stability and robustness of the classifier. Random Forest is known to run efficiently on large databases, can handle thousands of input variables without variable deletion, and requires less computational power than other ensemble methods like Boosting. However, compared to Decision Trees, the Random Forest algorithm is difficult to interpret, and a large number of decision trees can slow down the training process (Rodriguez- Galiano et al. 2012). In contrast to the Random Forest algorithm, a bagging technique, the eXtreme Gradient Boosting algorithm (XGBoost) is a boosting technique. Proposed by Tianqi Chen in 2016, it is a state-of-the-art ensemble learning method that combines multiple decision trees to create a more accurate model. The scalable, portable, and distributed gradient boosting framework uses a gradient descent algorithm and is a powerful tool for multi-class classification (Chen and Guestrin 2016). While the Random Forest and XGBoost algorithms fall under the category of ensemble learning, they differ in their approach to strengthening the model. Using Random Forest generates many trees in parallel, each with leaves of equal weight within the model to obtain better performance. The result is obtained by averaging the responses of the number of learners. On the other hand, XGBoost introduces leaf weighting by sequentially generating base learners to penalize those that do not improve the model predictability. The idea is to fix previous mistakes in the model, learn from them and improve performance in the next step. This continues 64 until there is no further improvement. Regularization is a vital feature of this type of algorithm. The final boosting ensemble uses a weighted average putting more weight on those with better performance on the training data. Both algorithms generally decrease the variance, while boosting also improves the bias. Additionally, the XGBoost algorithm is known for its speed and accuracy (Carmona, Climent, and Momparler 2019). For Random Forest and Gradient Boosting, the feature importance is calculated by averaging each importance obtained over all trees as explained in the DT paragraph (Ronaghan 2019). Learning by example, the Support Vector Machine (SVM) algorithm initially designed for binary classification is a powerful tool for solving classification problems (Noble 2006). The primary goal of SVM is to separate several classes in the training data by finding a hyperplane that maximizes the margin

between the classes to maximize the generalization ability. The margin is the distance between the hyperplane and the training data closest to the hyperplane. A kernel, a function, is used in SVM to transform the data into a higher dimensional space that can be more easily separated. A linear classifier can be used to separate the data since the kernel achieves a non-linear mapping of the data. The most straightforward kernel function is the linear kernel that datasets with many features usually use because they become linearly separable problems. For the classical SVM, one can set the kernel to linear and probability to true, and class memberships probability estimates are enabled (T.-F. Wu, Lin, and Weng 2004). Another widely used kernel is the Radial Basis Function (RBF), a non-linear function. For this kernel, it is essential to consider the two hyperparameters, gamma and C. The C parameter, common to all SVM kernels, balances the misclassification of training examples against the simplicity of the decision surface. A low C smoothens the decision surface, while a high C is intended to classify all training examples correctly. Gamma defines how much influence one training example has. The larger the gamma, the closer other examples must be to be impacted (Cervantes et al. 2020). 65 5.3.4 Model Training This section describes the model training and tools used to achieve optimal results. The fine- tuning process that is carried out to find the optimal values for the hyperparameters, which is a complex process, is explained. Furthermore, the adjusted evaluation metrics that fit our context of predicting more than one medical department are described. GridSearch cross-validation (GridSearch CV) is a hyperparameter optimization technique that exhaustively searches a given parameter space for the best-performing model. This study uses it as a strategy to find the optimal set of hyperparameters for a given model, such as for MLR or SVM. Automated hyperparameter selection is less time-consuming than manually adjusting the hyperparameters and can achieve better results (Ahmad et al. 2022). A 5-fold cross- validation which refers to the process of splitting the dataset into five distinct subsets, performing a grid search on each subset, and then aggregating the results of all five subsets to determine the best model parameters is chosen for verification, and a verbose of two is selected to display the computation time for each fold and parameter candidate. Furthermore, for each model, the parameter setting that gave the best results on the hold-out data within the grid search and the best score achieved on the hold-out data, a part of the train data, is used for other purposes. Standard scoring methods used for GridSearchCV and for evaluating the test set would be accuracy, precision, recall, or F1-score when one label in a classification task is predicted. However, this study is interested in finding the top diseases with the highest probabilities because, in reality, a given list of symptoms could belong to more than one medical specialty; therefore, the user should have alternatives that are also likely to be the case and not only be provided with one prediction. If the classifier predicts the class "Cardiovascular/Pulmonary", for example, as its first guess and "Surgery" as its second guess, that does not mean that "Surgery" would be wrong. Therefore, the model evaluation metrics are adjusted, and it is evaluated how often the correct 66 disease was among the top three diseases by computing Accuracy@k, which can be calculated as the number of correct predictions among the top three predicted classes divided by the total number of predictions. This is especially important since sometimes there will not be a distinct prediction with a probability of more than 70% for the first class, for example. It is likely that the top three predictions have a 40%, 35%, and 25% probability, respectively, and it is, therefore, very valuable for the user to see the first three predictions instead of only one. Furthermore, the Mean Reciprocal Rank (MRR) is computed to compare the model performance. The MRR is computed as the inverse of the rank of the first relevant item (Voorhees 2001; Le 2019). $|Ksess|\ MRR = 1 \sum 1\ |Krdrr|\ i{=}1\ ranii$ Equation 5: Mean Reciprocal Rank formula $Krdrr$ refers to all the classification tasks in the testing keyword lists set and, $ranii$ to the position of the correctly predicted category. The higher the rank of the correctly predicted category, meaning relevant results are close to the top results, the higher the MRR. Considering the top three predictions, the MRR will indicate where the primary category ranks. If the rank is, on average, two, then the MRR would be around 0.5, and for an average rank of three, it would be around 0.3, respectively. The goal is to have a high rank (Kavita 2022). 5.3.5 Results and Evaluation This section presents the results by comparing the model's performance, the features selected, and the model's explainability. The model achieving the best scores is chosen as the final model for our symptom checker prototype. To determine the best model, the models are ranked based on Accuracy@3 and MRR@3. The models are first tested with the NLP outputs created from the MT Samples dataset. All models are trained on the sequence classified and masked language keyword lists. Table 8 shows the 67 results obtained for all models on the MT Samples test set with the sequence classified keywords list and table 9 with the masked language keywords list, respectively. Models Hyperparameters Accuracy@3 MRR@3 Multinomial Logistic Regression Decision Tree Random Forest XGBoost Support Vector Machine multiclass = "multinomial", penalty = "l1", solver = "saga", max_iter = 1000, C = 1 max_depth = 17, criterion = "gini", max_features = 0.3 max_depth = 12, criterion = "entropy", max_features = "sqrt" objective = "multi: softprob", n_estimators = 300, max_depth = 5 Probability = true, C = 1, gamma = 0.01, kernel = "rbf" 0.8894 0.6047 0.8693 0.8777 0.9012 0.5670 0.409 0. 567 0.5463 0.6159 Table 8: Achieved results per classification model for the MT Samples sequence classified NLP model output showing the finetuned hyperparameters, the accuracy@3 and the MRR@ (Source: Table created by authors). Models Hyperparameters Accuracy@3 MRR@3 Multinomial Logistic Regression Decision Tree Random Forest XGBoost Support Vector Machine multiclass = "multinomial", penalty = "l1", solver = "saga", max_iter = 1000, C = 1 max_depth = 12, criterion = "entropy", max_features = 0.3 max_depth = 18, criterion = "entropy", max_features = "sqrt" objective = "multi: softprob", n_estimators = 300, max_depth = 7 Probability = true, C = 1, gamma = 0.01, kernel = "rbf" 0.8459 0.5779 0.8157 0.8409 0.8459 0.5321 0.404 0.5031 0.5888 0.5128 Table 9: Achieved results per classification model for the MT Samples masked language NLP model output showing the finetuned hyperparameters, the accuracy@3 and the MRR@ (Source: Table created by authors). The performance of all models gives better results for the sequence classified keyword lists, and it is therefore decided for the MIMIC-III dataset to train the models on this NLP output finally. When comparing the five models to each other, in table 8, one can witness the best performance both in Accuracy@3 and MRR@3 for the Support Vector Machine model with an Accuracy@3 of 90.12% and an MRR@3 of 0.6159, respectively. The model that is the most explainable is the Multinomial Logistic Regression model, which is the second-best performing model with an Accuracy@3 of 88.94%. As stated before, it is the only model that provides information about the model's feature importance per class by looking at the coefficients. Overall, based on our approach, the Decision Tree model performed rather 68 poorly throughout and hence is a bad candidate for medical specialty detection. The final results from training the models on the MIMIC-III dataset are shown in table 10. Models Accuracy@3 MRR@3 Multinomial Logistic Regression 0.8510 0.7454 Decision Tree 0.4295 0.3737 Random Forest 0.8449 0.7481 XGBoost 0.8796 0.7894 Support Vector Machine 0.8591 0.7676 Table 10: Achieved results per classification model for the MIMIC-III sequence classified NLP model output using the same hyperparameters as in table 8 (Source: Table created by authors). Being constrained by the capacity of the resources used and experiencing some hyperparameter tuning of more than 40 hours, it was decided to take the parameters from the previous fine-tuning of the models for the sequence classified keyword lists, as these give very good results, and it is reasonable to assume that this applies to the new data as well. It can be noted that although 16 instead of 11 classes are predicted for the MIMIC-III dataset, the models perform only slightly worse on average in comparison to the MT Samples dataset. The best result is achieved by the XGBoost model with an Accuracy@3 of 87.96%, closely followed by the Support Vector Machine and the Multinomial Logistic Regression Model with 85.91% and 85.1%, respectively. Therefore, XGBoost is chosen as the final model to be deployed for the prototype. Its Accuracy@3 per medical specialty is shown in table 13 in the appendix. To compare those results with the performance of existing symptom checkers, a systematic review of ten online symptom checker tools found that the Accuracy@3 for the top three predictions ranges from 19% to 38% (Wallace et al. 2022). In our study, a mean Accuracy@3 of 77.28% is achieved across all models for the MIMIC-III dataset, with the highest being 87.96% which certainly exceeds the performance of the reviewed symptom checkers by the systematic review paper. Additionally, we are not only interested in the model's results but also in the post-hoc analysis of the models, which includes the presentation of how those models can be explained. One aspect is the feature importance. Feature importance is essential to evaluate machine learning models because it allows to determine the most important features in predicting the output. It can aid in identifying potential model flaws like bias or overfitting. It is more comprehensible how the model works, and if needed, make improvements by knowing which features

have the greatest influence (Bhatt et al. 2020). The feature importance plots and number of features used plots are shown in tables 14 and 15 and figures 39 to 57 in the appendix for all models for which the feature importance or coefficients are available. For the MLR model, the largest positive coefficients indicate which features per class were most important for the model to predict a medical department. For example, 265 coefficients out of the 20749 features extracted from the training set were nonzero for the class Oncology. The words "tumor", "meningioma", and "mass", a lump in the body, are the most important ones to predict this class. In contrast, "rectus", "knee", "cranioplasty", or "scoliosis" are the essential words for the category Orthopedic surgery. For the XGBoost model, the feature importance of the 3920 features used can be shown in general. Here, "gp", "gestation", and "spleen" are the most important words. For model explainability, to better understand the behaviour of the classification model for user input, the open-source Local Interpretable Model-Agnostic Explanations (LIME) framework is used, published by Ribeiro et al. in 2016 (Ribeiro, Singh, and Guestrin 2016). Lime explains how the features of a specific instance, in this case, a list of symptoms from one transcription, influence the model's decision. The technique can be applied to any black-box algorithm and is therefore available for all models used in this study. Lime attempts to understand them by perturbing the instance's components and grasping how the predictions change (Dieber and Kirrane 2020). It is crucial in making the output of the algorithm interpretable and explainable. Out of the three explainers, namely image, tabular, and text explainer, the Lime text explainer provides information on which symptoms were most important to predict a class. The function 70 from the lime package explains the model's predicted probabilities on all features of the instance. This is done by choosing the length of the symptoms in the list as the number of features to compare the feature importance for the top three predicted classes. For instance, for the symptoms of "mass", "fatigue", and "cough", the XGBoost model predicts that the patient should see the oncology department with a probability of 87% and the Pulmonology department with a probability of 7%. Using Lime, one receives the information that symptoms of "mass" and "fatigue" are portrayed as contributing to the oncology department, while cough is evidence against it. Figure 21:Example of Lime output for a user's input with the keywords of "mass", "fatigue", and "cough". Showing which symptoms influenced the prediction of each medical department. In this case, the XGBoost model predicts an 87% probability of belonging to the Oncology department based on the features of mass and fatigue (Source: Figure created by authors). In conclusion, this part of the project presented a study on automated disease prediction from a list of keywords using five classification models trained on two datasets. We chose the XGBoost model as our final classification model since it has the best Accuracy@3 of 87.96% and an MRR@3 of 0.8794 for the MIMIC-III dataset. As stated before, compared to other existing symptom checkers, our model significantly outperforms them in terms of Accuracy@3 (Wallace et al. 2022). Moreover, we offer the explainability feature on our website which increases the user's trust into the predictions they receive. 6 Summary Each part of the model development phase results in high-quality outcomes and contributes to a good user experience. By concatenating the individual models, a data and model pipeline is created, which allows the user to give speech input about their symptoms and receive predictions for the optimal medical department. The most important contributions of each model are 71 highlighted below. The first section of the model developments covers the speech to text models. This section described a method for optimizing cutting-edge automatic speech recognition models on data from medical speech. The two models, Wav2vec2.0 and HuBERT, scored impressive Word Error Rates of 11 and 10.6%, respectively, on the scant quantity of voice data that was available for this study. Ultimately, the Wav2vec2.0 model was selected for the implementation in Streamlit since it performs more smoothly, predicts more quickly, and the error rates for both models are comparably good. The second part uses natural language processing techniques to extract keywords, which are then used as input data for the classification model. Therefore, we use the state-of-the-art Bio_ClinicalBERT, which has been pre-trained on medical data. In this approach, we fine-tune the model with two different methods, Masked Language Modelling and Sequence Classification, on two different datasets called MT Samples and MIMIC-III. The training/loss curve of all models is convex, meaning the model learns from its mistakes. The embeddings of both models are then used in the downstream task to extract the keywords with the KeyBERT model. The result is, on average, 17-19 significant, important medical keywords. For filtering the keywords on the app, the model is very fast, allowing a good customer experience with no waiting time. The last part of the model development involved training five classification models to predict medical departments based on symptoms. Five classification models were trained with different inputs. The XGBoost model achieved the best result with an Accuracy@3 of almost 88% using the sequence classified keywords from the final MIMIC-III dataset. Overall, very strong results were achieved compared to previous studies where models performed less well. It turns out that for a business-relevant prototype, the explanatory power of the models must not be neglected. 7 Discussion The outcomes of this paper have provided insight into the development and implementation of a symptom checker. Strong results are achieved, which leads to the conclusion that this prototype can be turned into a viable product. Nevertheless, the results should be interpreted with caution due to the limitations of the present study that will be presented in this chapter. A reflection on the research process is provided, and several limitations and potential consequences of the study's design are discussed. The section ends with several recommendations and an outlook for future research. 7.1 Limitations During this study, we faced several limitations. The first is that the performance of the model is highly dependent on the input data. Since the NLP model has been trained using different medical records, it has learned specific complex words that are common to doctors' speech features very well but are not found in everyday speech features. Consequently, the model works well when given specific medical nuances, while it is not so familiar with the more generalized client speech, which has different linguistic features. Hence, the model is not always able to understand the client's language. A major improvement could be to have the model trained on patient data as well. Second, the MIMIC-III dataset contains only intensive care records from a single healthcare facility, but since there are significant differences in care and practices, additional records from different facilities could improve the generalizability of the model. In addition, the model has only a limited number of diseases it has seen and can classify, which may result in poor performance for unseen data and diseases. Furthermore, clinical guidelines and practice constantly change. When new guidelines with new procedures and diagnoses are implemented, the algorithm's performance could be affected, and thus, the algorithm should be recalibrated. Regarding the technical limitations, we faced a lack of computing power for the NLP model. Since the models were running on our laptop or on a virtual server called Paperspace (Paperspace 2022), which was limited to six hours, the runtime was an additional technical constraint. Some models required too many hours to tune the hyperparameters, which could not be done because the required runtime and storage capacity was not available. Since natural language processing models typically require a significant amount of data and computational power to be trained, a laptop may not be able to handle larger-scale models. In addition, laptops typically do not have powerful GPUs or TPUs, which can significantly affect the accuracy and speed of a natural language processing model. Another limitation was our domain knowledge. Without medical domain knowledge, it is challenging to understand medical concepts and theories hence making it harder to analyze the extracted keywords and the predicted classifications. 7.2 Outlook Our models can be used as a solid baseline for future work. As stated in the limitations section, there is still a need for improved data quality for further machine learning processes. Extensive data collection, including high-quality electronic health records from multiple hospitals, is necessary to increase the performance and generalizability of the models. Additionally, with the increased use of our application, the data entered will be returned to the model for training and can enhance the performance even more. Moreover, if a person with medical domain knowledge was on the team, then the insights of the post-hoc analysis of the classification could also be used to remove other non-relevant features during the cleaning process and improve the model results even more to increase its meaningfulness. Currently, predictions are made on the highest level of medical specialties. In the future, to provide more use cases, diagnoses can be predicted using a more granular classification of ICD codes. Moreover, with the integration of specific user info like age, gender, or historical medical 74 records into the symptom checker interface, a better contextual understanding of individual symptoms can be provided. Furthermore, other areas in the field of disease detection could be explored, such as the potential to combine the information from clinical notes with other data sources, including image data and laboratory test data, to predict diagnoses. 8 Conclusion Our study provides valuable

insights into the development of a symptom checker. The prototype of a symptom checker that was built has shown to be effective in predicting medical departments based on a user's oral description of their symptoms. Our results exceeded expectations, indicating a remarkable level of achievement in comparison to existing work. We believe that the proposed use cases of this symptom checker can help to ease the burden on healthcare professionals and the healthcare system while also providing patients with more information to better make informed decisions on which medical department to consult. In conclusion, this symptom checker is an important and invaluable tool in the pursuit of understanding and managing one's health. Bibliography "5 Best Medical Symptom Checkers." 2022. OpenMD.Com. November 22, 2022. https://openmd.com/directory/symptoms. Ahmad, Ghulab Nabi, Hira Fatima, Shafi Ullah, Abdelaziz Salah Saidi, and Imdadullah. 2022. "Efficient Medical Diagnosis of Human Heart Diseases Using Machine Learning Techniques With and Without GridSearchCV." IEEE Access 10: 80151–73. https://doi.org/10.1109/ACCESS.2022.3165792. Akrout, Mohamed, Amir-massoud Farahmand, Tory Jarmain, and Latif Abid. 2019. "Improving Skin Condition Classification with a Visual Symptom Checker Trained Using Reinforcement Learning." In Medical Image Computing and Computer Assisted Intervention – MICCAI 2019, edited by Dinggang Shen, Tianming Liu, Terry M. Peters, Lawrence H. Staib, Caroline Essert, Sean Zhou, Pew-Thian Yap, and Ali Khan, 549–57. Lecture Notes in Computer Science. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-32251-9_60. Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. 2017. "Understanding of a Convolutional Neural Network." In 2017 International Conference on Engineering and Technology (ICET), 1–6. https://doi.org/10.1109/ICEngTechnol.2017.8308186. Alsentzer, Emily, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. 2019. "Publicly Available Clinical BERT Embeddings," April. https://doi.org/10.48550/arXiv.1904.03323. Aly, Mohamed. 2005. "Survey on Multiclass Classification Methods." Neural Netw 19 (1): 9. Anaconda. 2016. "Anaconda Software Distribution." Anaconda Documentation. 2016. https://anaconda.com/. Antony, Linta, Sami Azam, Eva Ignatious, Ryana Quadir, Abhijith Reddy Beeravolu, Mirjam Jonkman, and Friso De Boer. 2021. "A Comprehensive Unsupervised Framework for Chronic Kidney Disease Prediction." IEEE Access 9: 126481–501. https://doi.org/10.1109/ACCESS.2021.3109168. Anuchitanukul, Atijit, and Lucia Specia. 2022. "Burst2Vec: An Adversarial Multi-Task Approach for Predicting Emotion, Age, and Origin from Vocal Bursts." ArXiv Preprint ArXiv:2206.12469. Anwar, Mubbashra, Nadeem Javaid, Adia Khalid, Muhammad Imran, and Muhammad Shoaib. 2020. "Electricity Theft Detection Using Pipeline in Machine Learning." In 2020 International Wireless Communications and Mobile Computing (IWCMC), 2138–42. https://doi.org/10.1109/IWCMC48107.2020.9148453. Avasthi, Ananya. 2021. "The Evolution of NLP." October 5, 2021. https://datasaur.ai//blog- posts/the-evolution-of-nlp. Baevski, Alexei, Alexis Conneau, and Michael Auli. 2020. "Wav2vec 2.0: Learning the structure of speech from raw audio." September 24, 2020. https://ai.facebook.com/blog/wav2vec-20-learning-the-structure-of-speech-from-raw- audio/. Baevski, Alexei, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. "Wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations." arXiv. https://doi.org/10.48550/arXiv.2006.11477. Basarkar, Ankit. 2017. "DOCUMENT CLASSIFICATION USING MACHINE LEARNING." Master of Science, San Jose, CA, USA: San Jose State University. https://doi.org/10.31979/etd.6jmu-9xdt. Bayaga, Anass. 2010. "Multinomial Logistic Regression: Usage and Application in Risk Analysis." Journal of Applied Quantitative Methods 5 (2). Bhatt, Umang, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M. F. Moura, and Peter Eckersley. 2020. "Explainable Machine Learning in Deployment." In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, 648–57. FAT* '20. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3351095.3375624. Biseda, Brent, Gaurav Desai, Haifeng Lin, and Anish Philip. 2020. "Prediction of ICD Codes with Clinical BERT Embeddings and Text Augmentation with Label Balancing Using MIMIC-III." arXiv. http://arxiv.org/abs/2008.10492. Briggs, James. 2021. "Masked-Language Modelling With BERT." Medium. September 2, 2021. https://towardsdatascience.com/masked-language-modelling-with-bert- 7d49793e5d2c. Brownlee, Jason. 2022. "Difference Between a Batch and an Epoch in a Neural Network - MachineLearningMastery.Com." August 2022. https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/. Carmona, Pedro, Francisco Climent, and Alexandre Momparler. 2019. "Predicting Failure in the U.S. Banking Sector: An Extreme Gradient Boosting Approach." International Review of Economics & Finance 61 (May): 304–23. https://doi.org/10.1016/j.iref.2018.03.008. Çelikten, Azer, Aybars Ugur, and Hasan Bulut. 2021. Keyword Extraction from Biomedical Documents Using Deep Contextualized Embeddings. https://doi.org/10.1109/INISTA52262.2021.9548470. Cervantes, Jair, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, and Asdrubal Lopez. 2020. "A Comprehensive Survey on Support Vector Machine Classification: Applications, Challenges and Trends." Neurocomputing 408 (September): 189–215. https://doi.org/10.1016/j.neucom.2019.10.118. Charbuty, Bahzad, and Adnan Abdulazeez. 2021. "Classification Based on Decision Tree Algorithm for Machine Learning." Journal of Applied Science and Technology Trends 2 (01): 20–28. Chaudhary, Archana, Savita Kolhe, and Raj Kamal. 2016. "An Improved Random Forest Classifier for Multi-Class Classification." Information Processing in Agriculture 3 (4): 215–22. https://doi.org/10.1016/j.inpa.2016.08.002. Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–94. https://doi.org/10.1145/2939672.2939785. Chitra, R., and Dr V. Seenivasagam. 2013. "Heart Disease Prediction System Using Supervised Learning Classifier." Bonfring International Journal of Software Engineering and Soft Computing 3 (issue 1): 01–07. https://doi.org/10.9756/BIJSESC.4336. Costa, Luana Bonome Message, and Moacir Godinho. 2016. "Lean Healthcare: Review, Classification and Analysis of Literature." Production Planning & Control 27 (10): 823–36. Dauria, Erika. 2019. "Accuracy, Recall & Precision." Medium (blog). December 8, 2019. https://medium.com/@erika.dauria/accuracy-recall-precision-80a5b6cbd28d. Davenport, Thomas, and Ravi Kalakota. 2019. "The Potential for Artificial Intelligence in Healthcare." Future Healthcare Journal 6 (2): 94–98. https://doi.org/10.7861/futurehosp.6-2-94. De Boer, Pieter-Tjerk, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. 2005. "A Tutorial on the Cross-Entropy Method." Annals of Operations Research 134 (1): 19– 67. Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." arXiv. http://arxiv.org/abs/1810.04805. Dieber, Jürgen, and Sabrina Kirrane. 2020. "Why Model Why? Assessing the Strengths and Limitations of LIME." arXiv. http://arxiv.org/abs/2012.00093. Dorash, Maryna. 2017. "Machine Learning vs. Rule Based Systems in NLP." Friendly Data (blog). December 26, 2017. https://medium.com/friendly-data/machine-learning-vs- rule-based-systems-in-nlp-5476de53c3b8. El-Habil, Abdalla M. 2012. "An Application on Multinomial Logistic Regression Model." Pakistan Journal of Statistics and Operation Research, March, 271–91. https://doi.org/10.18187/pjsor.v8i2.234. Fernandez, Alberto, Salvador Garcia, Francisco Herrera, and Nitesh V. Chawla. 2018. "SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-Year Anniversary." Journal of Artificial Intelligence Research 61 (April): 863– 905. https://doi.org/10.1613/jair.1.11192. Garreta, Raul, and Guillermo Moncecchi. 2013. Learning Scikit-Learn: Machine Learning in Python. Packt Publishing Ltd. Gilbert, Stephen, Alicia Mehl, Adel Baluch, Caoimhe Cawley, Jean Challiner, Hamish Fraser, Elizabeth Millen, et al. 2020. "How Accurate Are Digital Symptom Assessment Apps for Suggesting Conditions and Urgency Advice? A Clinical Vignettes Comparison to GPs." BMJ Open 10 (12): e040269. https://doi.org/10.1136/bmjopen-2020-040269. Grootendorst, Maarten. 2021. "KeyBERT - KeyBERT." January 2021. https://maartengr.github.io/KeyBERT/api/keybert.html#keybert._model.KeyBERT. Gunel, Beliz, Jingfei Du, Alexis Conneau, and Ves Stoyanov. 2020. "Supervised Contrastive Learning for Pre-Trained Language Model Fine-Tuning." ArXiv Preprint ArXiv:2011.01403. Gurevych, Irina, and Nils Reimers. 2019. "MLM — Sentence-Transformers Documentation." November 2019. https://www.sbert.net/examples/unsupervised_learning/MLM/README.html. Guyon, Isabelle. 2008. "Practical Feature Selection: From Correlation to Causality." Mining Massive Data Sets for Security: Advances in Data Mining, Search, Social Networks and Text Mining, and Their Applications to Security, 27–43. Hajra, Suvadeep. 2019. "Making Convex Loss Functions Robust to Outliers Using e- Exponentiated Transformation." arXiv. http://arxiv.org/abs/1902.06127. Hapke, Hannes, and Catherine Nelson. 2020. Building Machine Learning Pipelines. O'Reilly Media, Inc. Hasan, Basna Mohammed Salih, and Adnan Mohsin Abdulazeez. 2021. "A Review of Principal Component Analysis Algorithm for Dimensionality Reduction." Journal of Soft

Computing and Data Mining 2 (1): 20–30. Hauptmann, Alexander G, and Alexander Rudnicky. 1990. "A Comparison of Speech and Typed Input." In Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990. "Health. Powered by Ada." 2022. Ada. November 22, 2022. https://ada.com/. Hochberg, Irit, Raviv Allon, and Elad Yom-Tov. 2020. "Assessment of the Frequency of Online Searches for Symptoms Before Diagnosis: Analysis of Archival Data." Journal of Medical Internet Research 22 (3): e15065. https://doi.org/10.2196/15065. Hogan, Andrew J, and Reynard R Bouknight. 2002. "The Emergency Department as a Gateway for Hospitalization: Evidence from the National Medical Expenditure Survey." Journal of Health Care for the Poor and Underserved 13 (3): 288–97. Honigman, Leah S, Jennifer L Wiler, Sean Rooks, and Adit A Ginde. 2013. "National Study of Non-Urgent Emergency Department Visits and Associated Resource Utilization." Western Journal of Emergency Medicine 14 (6): 609. Hsu, Wei-Ning, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. "Hubert: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units." IEEE/ACM Transactions on Audio, Speech, and Language Processing 29: 3451–60. Hsu, Wei-Ning, Yao-Hung Hubert Tsai, Benjamin Bolte, Ruslan Salakhutdinov, and Abdelrahman Mohamed. 2021. "HuBERT: How Much Can a Bad Teacher Benefit ASR Pre-Training?" In ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 6533–37. IEEE. Huang, Jinmiao, Cesar Osorio, and Luke Wicent Sy. 2019. "An Empirical Evaluation of Deep Learning for ICD-9 Code Assignment Using MIMIC-III Clinical Notes." Computer Methods and Programs in Biomedicine 177 (August): 141–53. https://doi.org/10.1016/j.cmpb.2019.05.024. Hügle, Maria, Patrick Omoumi, Jacob M van Laar, Joschka Boedecker, and Thomas Hügle. 2020. "Applied Machine Learning and Artificial Intelligence in Rheumatology." Rheumatology Advances in Practice 4 (1): rkaa005. https://doi.org/10.1093/rap/rkaa005. Humbert-Droz, Marie, Pritam Mukherjee, and Olivier Gevaert. 2022. "Strategies to Address the Lack of Labeled Data for Supervised Machine Learning Training With Electronic Health Records: Case Study for the Extraction of Symptoms From Clinical Notes." JMIR Medical Informatics 10 (3): e32903. https://doi.org/10.2196/32903. Jang, Eric, Shixiang Gu, and Ben Poole. 2016. "Categorical Reparameterization with Gumbel-Softmax." ArXiv Preprint ArXiv:1611.01144. Johnson, Alistair, Tom Pollard, and Roger Mark. 2015. "MIMIC-III Clinical Database." PhysioNet. https://doi.org/10.13026/C2XW26. Jones, Zachary M. 2013. "Git/GitHub, Transparency, and Legitimacy in Quantitative Research," 2. Joshi, Prateek. 2020. "Transfer Learning NLP|Fine Tune Bert For Text Classification." Analytics Vidhya (blog). July 20, 2020. https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/. Kalliamvakou, Eirini, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2016. "An In-Depth Study of the Promises and Perils of Mining GitHub." Empirical Software Engineering 21 (5): 2035–71. https://doi.org/10.1007/s10664-015-9393-5. Kavita, Ganesan. 2022. "Build Your First Text Classifier in Python with Logistic Regression - Kavita Ganesan, PhD." November 21, 2022. https://kavita-ganesan.com/news- classifier-with-logistic-regression-in-python/#.Y3twTi-l1QL. Kearns, Jodi. 2014. "Librivox: Free Public Domain Audiobooks." Reference Reviews. Këpuska, Veton Z, Hussien A Elharati, and others. 2015. "Robust Speech Recognition System Using Conventional and Hybrid Features of MFCC, LPCC, PLP, RASTA- PLP and Hidden Markov Model Classifier in Noisy Conditions." Journal of Computer and Communications 3 (06): 1. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2017. "ImageNet Classification with Deep Convolutional Neural Networks." Commun. ACM 60 (6): 84–90. https://doi.org/10.1145/3065386. Kujala, Sari, Tarja Heponiemi, Iliris Hörhammer, and Riitta Hänninen-Ervasti. 2020. "Health Professionals' Experiences of the Benefits and Challenges of Online Symptom Checkers." Digital Personalized Health and Medicine, 966–70. https://doi.org/10.3233/SHTI200305. Kumar, Raman. 2021. "BERT for Natural Language Processing |All You Need to Know about BERT." Analytics Vidhya (blog). May 27, 2021. https://www.analyticsvidhya.com/blog/2021/05/all-you-need-to-know-about-bert/. Le, Duc-Trong. 2019. "Finding Relevant Files for Bug Reports Based on Mean Reciprocal Rank Maximization Approach," 7. https://doi.org/10.1145/2635868.2635874. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep Learning." Nature 521 (7553): 436–44. Lee, Jinhyuk, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. "BioBERT: A Pre-Trained Biomedical Language Representation Model for Biomedical Text Mining." Bioinformatics 36 (4): 1234–40. https://doi.org/10.1093/bioinformatics/btz682. Lhoest, Quentin, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, et al. 2021. "Datasets: A Community Library for Natural Language Processing." In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 175–84. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-demo.21. Li, David, Kulamakan Kulasegaram, and Brian D. Hodges. 2019. "Why We Needn't Fear the Machines: Opportunities for Medicine in a Machine Learning World." Academic Medicine 94 (5): 623–25. https://doi.org/10.1097/ACM.0000000000002661. Liu, Xinyu, and Bhavik R. Bakshi. 2018. "Extracting Heuristics for Designing Sustainable Built Environments by Coupling Multiobjective Evolutionary Optimization and Machine Learning." In Computer Aided Chemical Engineering, edited by Mario R. Eden, Marianthi G. Ierapetritou, and Gavin P. Towler, 44:2539–44. 13 International Symposium on Process Systems Engineering (PSE 2018). Elsevier. https://doi.org/10.1016/B978-0-444-64241-7.50418-3. Liu, Yanli, Yourong Wang, and Jian Zhang. 2012. "New Machine Learning Algorithm: Random Forest." In International Conference on Information Computing and Applications, 246–52. Springer. Lutkevich, Ben. 2020. "What Is BERT (Language Model) and How Does It Work?" SearchEnterpriseAI. January 2020. https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model. Maas, Andrew, Quoc V Le, Tyler M O'neil, Oriol Vinyals, Patrick Nguyen, and Andrew Y Ng. 2012. "Recurrent Neural Networks for Noise Reduction in Robust ASR." Maglogiannis, Ilias G. 2007. Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in EHealth, HCI, Information Retrieval and Pervasive Technologies. IOS Press. Maier, Andreas. 2020. "Will We Ever Solve the Shortage of Data in Medical Applications?" Medium. June 11, 2020. https://towardsdatascience.com/will-we-ever-solve-the- shortage-of-data-in-medical-applications-70da163e2c2d. McFee, Brian, Colin Raffel, Dawen Liang, Daniel Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. "Librosa: Audio and Music Signal Analysis in Python." In , 18–24. Austin, Texas. https://doi.org/10.25080/Majora-7b98e3ed-003. Mergel, Ines. 2015. "Open Collaboration in the Public Sector: The Case of Social Coding on GitHub." Government Information Quarterly 32 (4): 464–72. https://doi.org/10.1016/j.giq.2015.09.004. Mooney, Paul. 2018. "Medical Speech, Transcription, and Intent." Moral, Pablo Del, Sławomir Nowaczyk, and Sepideh Pashami. 2022. "Why Is Multiclass Classification Hard?" IEEE Access 10: 80448–62. https://doi.org/10.1109/ACCESS.2022.3192514. Morris, Andrew, Viktoria Maier, and Phil Green. 2004. "From WER and RIL to MER and WIL: Improved Evaluation Measures for Connected Speech Recognition." In . MTHelpLine. 2022. "Transcribed Medical Transcription Sample Reports and Examples - MTSamples." November 16, 2022. https://mtsamples.com/. Muda, Lindasalwa, Mumtaj Begam, and Irraivan Elamvazuthi. 2010. "Voice Recognition Algorithms Using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques." ArXiv Preprint ArXiv:1003.4083. Muller, Britney. 2022. "BERT 101 - State Of The Art NLP Model Explained." March 2022. https://huggingface.co/blog/bert-101. Müller, Regina, Malte Klemmt, Hans-Jörg Ehni, Tanja Henking, Angelina Kuhnmünch, Christine Preiser, Roland Koch, and Robert Ranisch. 2022. "Ethical, Legal, and Social Aspects of Symptom Checker Applications: A Scoping Review." Medicine, Health Care and Philosophy 25 (4): 737–55. https://doi.org/10.1007/s11019-022- 10114-y. Muslim, Aries, Achmad Benny Mutiara, Rina Refianti, Cut Maisyarah Karyati, and Galang Setiawan. 2020. "Comparison of Accuracy between Long Short-Term Memory-Deep Learning and Multinomial Logistic Regression-Machine Learning in Sentiment Analysis on Twitter." International Journal of Advanced Computer Science and Applications 11 (2). Narein, Adith. 2021. "Embeddings in BERT." OpenGenus IQ: Computing Expertise & Legacy. October 5, 2021. https://iq.opengenus.org/embeddings-in-bert/. Noble, William S. 2006. "What Is a Support Vector Machine?" Nature Biotechnology 24 (12): 1565–67. https://doi.org/10.1038/nbt1206-1565. Nuthakki, Siddhartha, Sunil Neela, Judy W. Gichoya, and Saptarshi Purkayastha. 2019. "Natural Language Processing of MIMIC-III Clinical Notes for Identifying Diagnosis and Procedures with Neural Networks." arXiv. http://arxiv.org/abs/1912.12397. O'Shea, Keiron, and Ryan Nash. 2015. "An Introduction to Convolutional Neural Networks." ArXiv Preprint ArXiv:1511.08458. Padhye, Rohan, Senthil Mani, and Vibha Singhal Sinha. 2014. "A Study of External

Community Contribution to Open-Source Projects on GitHub." In Proceedings of the 11th Working Conference on Mining Software Repositories, 332–35. MSR 2014. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2597073.2597113. Paling, Steven, Jennifer Lambert, Jasper Clouting, Júlia González-Esquerré, and Toby Auterson. 2020. "Waiting Times in Emergency Departments: Exploring the Factors Associated with Longer Patient Waits for Emergency Care in England Using Routinely Collected Daily Data." Emergency Medicine Journal 37 (12): 781–86. Panayotov, Vassil, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. "Librispeech: An Asr Corpus Based on Public Domain Audio Books." In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 5206–10. IEEE. Panhuis, Willem G. van, Proma Paul, Claudia Emerson, John Grefenstette, Richard Wilder, Abraham J. Herbst, David Heymann, and Donald S. Burke. 2014. "A Systematic Review of Barriers to Data Sharing in Public Health." BMC Public Health 14 (1): 1144. https://doi.org/10.1186/1471-2458-14-1144. Paperspace. 2022. "Notebooks Tutorial | Paperspace." 2022. https://docs.paperspace.com/gradient/tutorials/notebooks-tutorial/. Park, Andrea. 2021. "Symptom Checking Platform Developer Ada Health Inks Partnerships with Sanofi, Takeda, Alnylam." Fierce Biotech. June 14, 2021. https://www.fiercebiotech.com/medtech/symptom-checking-platform-ada-health-inks- partnerships-takeda-sanofi-alnylam. Perez-Riverol, Yasset, Laurent Gatto, Rui Wang, Timo Sachsenberg, Julian Uszkoreit, Felipe da Veiga Leprevost, Christian Fufezan, et al. 2016. "Ten Simple Rules for Taking Advantage of Git and GitHub." PLoS Computational Biology 12 (7): e1004947. https://doi.org/10.1371/journal.pcbi.1004947. Pestian, John P., Chris Brew, Pawel Matykiewicz, DJ Hovermale, Neil Johnson, K. Bretonnel Cohen, and Wlodzislaw Duch. 2007. "A Shared Task Involving Multi- Label Classification of Clinical Free Text." In Biological, Translational, and Clinical Language Processing, 97–104. Prague, Czech Republic: Association for Computational Linguistics. https://aclanthology.org/W07-1013. Priyam, Anuja, Rahul Gupta, Anju Rathee, and Saurabh Srivastava. 2013. "Comparative Analysis of Decision Tree Classification Algorithms." International Journal of Current Engineering and Technology, 4. Putnam, Cheryl. 2010. "Mayo Clinic." Journal of Consumer Health on the Internet 14 (4): 392–400. https://doi.org/10.1080/15398285.2010.524122. Rasmy, Laila, Yang Xiang, Ziqian Xie, Cui Tao, and Degui Zhi. 2021. "Med-BERT: Pretrained Contextualized Embeddings on Large-Scale Structured Electronic Health Records for Disease Prediction." Npj Digital Medicine 4 (1): 1–13. https://doi.org/10.1038/s41746-021-00455-y. Rese, Mario. 2000. "Logistische Regression." In Multivariate Analysemethoden: Eine anwendungsorientierte Einführung, edited by Klaus Backhaus, Bernd Erichson, Wulff Plinke, and Rolf Weiber, 104–44. Springer-Lehrbuch. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-662-08893-7_3. Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. "'Why Should I Trust You?': Explaining the Predictions of Any Classifier." In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1135–44. KDD '16. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2939672.2939778. Rodriguez-Galiano, Victor Francisco, Bardan Ghimire, John Rogan, Mario Chica-Olmo, and Juan Pedro Rigol-Sanchez. 2012. "An Assessment of the Effectiveness of a Random Forest Classifier for Land-Cover Classification." ISPRS Journal of Photogrammetry and Remote Sensing 67: 93–104. Ronaghan, Stacey. 2019. "The Mathematics of Decision Trees, Random Forest and Feature Importance in Scikit-Learn and Spark." Medium. November 1, 2019. https://towardsdatascience.com/the-mathematics-of-decision-trees-random-forest-and- feature-importance-in-scikit-learn-and-spark-f2861df67e3. Ruan, Sherry, Jacob O Wobbrock, Kenny Liou, Andrew Ng, and James Landay. 2016. "Speech Is 3x Faster than Typing for English and Mandarin Text Entry on Mobile Devices." ArXiv Preprint ArXiv:1608.07323. Saltzer, Jerome H. 2020. "The Origin of the 'Mit License.'" IEEE Annals of the History of Computing 42 (4): 94–98. Seth, Yashu. 2019. "BERT Explained – A List of Frequently Asked Questions." Let the Machines Learn (blog). June 11, 2019. https://yashuseth.wordpress.com/2019/06/12/bert-explained-faqs-understand-bert- working/. Singh, A. K. Bhavani, Mounika Guntu, Ananth Reddy Bhimireddy, Judy W. Gichoya, and Saptarshi Purkayastha. 2020. "Multi-Label Natural Language Processing to Identify Diagnosis and Procedure Codes from MIMIC-III Inpatient Notes." arXiv. http://arxiv.org/abs/2003.07507. Son, Jaemin, Joo Young Shin, Hoon Dong Kim, Kyu-Hwan Jung, Kyu Hyung Park, and Sang Jun Park. 2020. "Development and Validation of Deep Learning Models for Screening Multiple Abnormal Findings in Retinal Fundus Images." Ophthalmology 127 (1): 85–94. https://doi.org/10.1016/j.ophtha.2019.05.029. Sottile, Anthony. 2017. "Pre-Commit." A Framework for Managing and Maintaining Multi- Language Pre-Commit Hooks. https://pre-commit.com/. Sphinx. 2022. "Sphinx Documentation," 641. Sushil, Madhumita, Simon Suster, and Walter Daelemans. 2021. "Are We There yet? Exploring Clinical Domain Knowledge of BERT Models." In Proceedings of the 20th Workshop on Biomedical Language Processing, 41–53. Online: Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.bionlp-1.5. "Symptom Checker - Mayo Clinic." 2022. November 22, 2022. https://www.mayoclinic.org/symptom-checker/select-symptom/itt-20009075. "Symptom Checker with Body from WebMD - Check Your Medical Symptoms." 2022. WebMD. November 21, 2022. https://symptoms.webmd.com/. Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. "Going Deeper with Convolutions." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1–9. Tang, Matthew, Priyanka Gandhi, Md Ahsanul Kabir, Christopher Zou, Jordyn Blakey, and Xiao Luo. 2019. "Progress Notes Classification and Keyword Extraction Using Attention-Based Deep Learning Models with BERT." arXiv. http://arxiv.org/abs/1910.05786. The imbalanced-learn developers. 2022. "Pipeline — Version 0.9.1." November 16, 2022. https://imbalanced-learn.org/stable/references/generated/imblearn.pipeline.Pipeline.html. Tian, Yonglong, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. 2020. "What Makes for Good Views for Contrastive Learning?" Advances in Neural Information Processing Systems 33: 6827–39. Uszkoreit, Jakob. 2017. "Transformer: A Novel Neural Network Architecture for Language Understanding." 2017. https://ai.googleblog.com/2017/08/transformer-novel-neural- network.html. Voorhees, Ellen M. 2001. "The TREC Question Answering Track." Natural Language Engineering 7 (4): 361–78. https://doi.org/10.1017/S1351324901002789. Wallace, William, Calvin Chan, Swathikan Chidambaram, Lydia Hanna, Fahad Mujtaba Iqbal, Amish Acharya, Pasha Normahani, et al. 2022. "The Diagnostic and Triage Accuracy of Digital and Online Symptom Checker Tools: A Systematic Review." Npj Digital Medicine 5 (1): 1–9. https://doi.org/10.1038/s41746-022-00667-w. Wang, Shangzhou, Haohui Lu, Arif Khan, Farshid Hajati, Matloob Khushi, and Shahadat Uddin. 2022. "A Machine Learning Software Tool for Multiclass Classification." Software Impacts 13 (August): 100383. https://doi.org/10.1016/j.simpa.2022.100383. Wang, Yingzhi, Abdelmoumene Boumadane, and Abdelwahab Heba. 2021. "A Fine- Tuned Wav2vec 2.0/Hubert Benchmark for Speech Emotion Recognition, Speaker Verification and Spoken Language Understanding." ArXiv Preprint ArXiv:2111.02735. Weik, Martin. 2012. Communications Standard Dictionary. Springer Science & Business Media. Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, et al. (2018) 2020. "Transformers: State-of-the-Art Natural Language Processing." Python. Association for Computational Linguistics. https://www.aclweb.org/anthology/2020.emnlp-demos.6. Wu, Chien-Sheng, Steven Hoi, Richard Socher, and Caiming Xiong. 2020. "TOD-BERT: Pre-Trained Natural Language Understanding for Task-Oriented Dialogue." arXiv. http://arxiv.org/abs/2004.06871. Wu, Patrick, Aliya Gifford, Xiangrui Meng, Xue Li, Harry Campbell, Tim Varley, Juan Zhao, et al. 2019. "Mapping ICD-10 and ICD-10-CM Codes to Phecodes: Workflow Development and Initial Evaluation." JMIR Medical Informatics 7 (4): e14325. https://doi.org/10.2196/14325. Wu, Ting-Fan, Chih-Jen Lin, and Ruby C. Weng. 2004. "Probability Estimates for Multi- Class Classification by Pairwise Coupling." The Journal of Machine Learning Research 5 (December): 975–1005. Yu, Dong, and Li Deng. 2016. Automatic Speech Recognition. Vol. 1. Springer. Zhan, Xianghao, Marie Humbert-Droz, Pritam Mukherjee, and Olivier Gevaert. 2021. "Structuring Clinical Text with AI: Old vs. New Natural Language Processing Techniques Evaluated on Eight Common Cardiovascular Diseases." medRxiv. https://doi.org/10.1101/2021.01.27.21250477. List of Abbreviations AI Artificial Intelligence API Application Programming Interface ASR Automatic Speech Recognition BERT Bidirectional Encoder Representations from Transformers CART Classification and Regression Tree CER Character Error Rate CLS Classification CNN Convolutional Neural Network CPU Central Processing Unit CSS Cascading Style Sheets CSV Comma-separated values CTC Connectionist Temporal Classification DT Decision Tree EHR Electronic Health Records GB Gigabyte GiB Gibibyte GPU Graphics Processing Unit GridSearchCV

GridSearch cross-validation HTML Hypertext Markup Language HuBERT Hidden unit Bidirectional Encoder Representations from Transformers ICD International Classification of Diseases ICU Intensive Care Unit kHz Kilohertz LIME Local Interpretable Model-Agnostic Explanations MFCC Mel Frequency Cepstral Coefficients MIMIC-III Multi-parameter Intelligent Monitoring in Intensive Care MIT Massachusetts Institute of Technology MLR Multinomial Logistic Regression MRR Mean Reciprocal Rank MSTI Medical Speech, Transcription, and Intent MT Samples Medical Transcription Samples NaN Not a Number NLP NOVA SBE PCA RAM RBF RF SEP SMOTE SVM TPU ULMFiT vCPU WAV WER WHO XGBoost Natural Language Processing NOVA School of Business and Economics Principal Component Analysis Random-access Memory Radial Basis Function Random Forest Separation Synthetic Minority Oversampling Technique Support Vector Machine Tensor Processing Unit Universal Language Model Fine-tuning Virtual Central Processing Unit Waveform Audio File Format Word Error Rate World Health Organization eXtreme Gradient Boosting Appendix 3.2.1 Speech to Text Figure 22: Boxplot of speech to text audio quality. The audio quality ranges from 3 to 5 points, with an average of 3.68 points. (Source: Figure created by authors). Figure 24: Histogram of the length of the speech to text audio files. The majority of the audio files are between 0 and 100 seconds long. The distribution is skewed to the right, which means that there are a few audio files that are very long, and most of them are between 2.5 and 5.0 seconds. (Source: Figure created by authors). Figure 23: Histogram of speech to text number of words in the label column. Most of the phrases are 8 words long, although the longest phrases contain up to 29 words. (Source: Figure created by authors). Figure 25: Bar chart showing how often each sample rate occurs. The sample rate is the number of samples of audio carried per second, measured in Hertz (Hz). The sample rate is the number of times a sound wave is measured per second. The most common audio sample rate is 48000, followed by 44100. (Source: Figure created by authors). Figure 26: Waveform of sample audio file. The vertical axis of the waveform shows the amplitude of the sound, while the horizontal axis shows the time. The peaks and valleys indicate a dynamic sound as each peak represents a word, while a valley represents silence. (Source: Figure created by authors). Figure 27: Mel-frequency cepstral coefficients of sample audio file. The MFCC figure displays the strength of the audio signal's various frequency components as measured at various points in time. (Source: Figure created by authors). Figure 28: Spectrum of sample audio file. An audio file's spectrum is a graphic depiction of the sound's many frequencies. It displays how the sound's various frequency components stack up in terms of relative strength. The spectrum can help you get a feel of the sound's general character and balance, as well as any standout elements or patterns that might be present. (Source: Figure created by authors). 3.2.2 MT Samples Figure 29: Number of Samples per Medical Specialty in the MT Samples dataset. The figure shows only the top 10 medical specialties by the number of samples. Overall, there are 16 different medical specialties. (Source: Figure created by authors). 3.2.3 MIMIC-III ICD Chapter code Medical Specialty Certain Conditions Originating In The Perinatal Period Complications Of Pregnancy, Childbirth, And The Puerperium Congenital Anomalies Diseases Of The Blood And Blood-Forming Organs Diseases Of The Circulatory System Diseases Of The Digestive System Diseases Of The Genitourinary System Diseases Of The Musculoskeletal System And Connective Tissue Diseases Of The Nervous System And Sense Organs Diseases Of The Respiratory System Diseases Of The Skin And Subcutaneous Tissue" Obstetrics & Gynaecology Obstetrics & Gynaecology Primary Care Hematology Cardiothoracic & Vascular Gastroenterology Urology Orthopedic surgery Neurology Pulmonology Dermatology Endocrine, Nutritional And Metabolic Diseases, And Immunity Endocrinology Disorders Infectious And Parasitic Diseases Infectious Disease Specialty Injury And Poisoning Emergency Department Mental Disorders Psychiatry Neoplasms Oncology Supplementary Classification Of External Causes Of Injury And Emergency Department Poisoning Supplementary Classification Of Factors Influencing Health Status And Internal Medicine Department Contact With Health Services Symptoms, Signs, And Ill-Defined Conditions Internal Medicine Department Table 11: Mappings of ICD-9 chapter codes and medical specialties. 19 different chapter codes are mapped to 16 different medical specialties. The mapping process is described in section 3.2.3. (Source: Table created by authors). 4.1 GitHub Structure Figure 30: Structure of GitHub repository. Depicted are the most important folders of the repository and the structure for the "src" folder. This figure illustrates the transparent structure and the possibility of recycling code based on the modular setup. (Source: Figure created by authors). 4.2 Streamlit Application Figure 31: Layout of starting page of the application. The user can record their audio input, listen to it, and see the performed transcription. If they are satisfied with their input, they can click on the red button to see the most suitable medical departments. (Source: Figure created by authors). Figure 32: Layout of the classification page of the application. The user sees the three most suitable medical departments based on their input. Each department recommendation can be expanded to show the probability percentage and the keywords which led to the respective decision. (Source: Figure created by authors). 5.1.4 Results and Evaluation Reference Phrase Predicted Phrase WER Score CER Score my stomach hurts my stomag hurts 0.333 0.125 i don't have pain i do not have pain 0.5 0.118 my knee pains me a lot my nee paints me a lot 0.333 0.091 Table 12: Evaluation metrics for three samples. This shows that small mistakes mostly have a large influence on the WER but a rather smaller one on the CER. The first example has only one spelling mistake, the second example predicts an entire word wrong, and the third one has two spelling mistakes. (Source: Table created by authors). 5.2.2 BERT Model T1 T2 … TN Trm Trm Trm Trm Trm Trm E1 E2 … EN Figure 33:BERT information flow. The first layer E is the embedding layer where the process starts. The Trm are the connected intermediate layers, and the final T layer is the output layer. This graphic visualizes how all intermediate layers are connected. (Source: Figure created by authors, based on Devlin et al. (Devlin et al. 2019)). 5.2.4 BERT Model MT Samples Dataset MIMIC-III Dataset Figure 34: MLM for MT Samples. The graphic shows the loss curve while training. The loss curve decreases and then flattens quite early. (Source: Figure created by authors). Figure 35: MLM for MIMIC-III. The graphic shows the loss curve while training. The loss curve is decreasing steadily. The flattening process takes longer than for the MT Samples dataset. (Source: Figure created by authors). Figure 36: SCM for MT Samples. The graphic shows the loss curve while training. The loss curve decreases a bit later than the one from MLM on MT Samples (Source: Figure created by authors). Figure 37: SCM for MIMIC-III. The graphic shows the loss curve while training. The loss curve decreases and then flattens faster than for MLM on MIMIC- III. (Source: Figure created by authors). 5.2.5 Results and Evaluation with KeyBERT Figure 38: Percentage of the same keywords extracted from both models. We can see that the percentage of equal keywords is, on average, 50% at maximum and 30% at minimum. (Source: Table created by authors). 5.3.1 Results and Evaluation Medical Specialties Accuracy@3 Cardiothoracic & Vascular 0.9804 Emergency Department 0.9640 Gastroenterology 0.9158 Infectious Disease Specialty 0.8858 Pulmonology 0.8752 Oncology 0.8446 Psychiatry 0.7654 Obstetrics & Gynaecology 0.6875 Endocrinology 0.6564 Neurology 0.5899 Orthopedic surgery 0.5882 Urology 0.4645 Primary Care 0.4103 Hematology 0.2632 Dermatology 0.2069 Internal Medicine Department 0.1803 Table 13: Accuracy@3 per medical specialty for the final XGBoost model in descending order (Source: Table created by authors). Models # Features used out of 20749 Multinomial Logistic Regression 17103 Decision Tree 633 Random Forest 7232 XGBoost 3920 Table 14: Number of features used per model for the MIMIC-III dataset where Count Vectorizer extracted in total 20,749 features from the list of keywords (Source: Table created by authors). Medical Specialties Number of non-zero coefficients Cardiothoracic & Vascular 2218 Emergency Department 2097 Gastroenterology 1426 Infectious Disease Specialty 1924 Pulmonology 1627 Oncology 1272 Psychiatry 437 Obstetrics & Gynaecology 265 Endocrinology 888 Neurology 937 Orthopedic surgery 621 Urology 993 Primary Care 377 Hematology 587 Dermatology 471 Internal Medicine Department 963 Table 15: Number of non-zero features per class for MLR model, meaning the number of features used per medical specialty (Source: Table created by authors). Figure 39: Feature importance bar plot based on the XGBoost feature importance built-in function ((Source: Figure created by authors). Figure 41: Feature importance bar plot based on the Decision Tree feature importance built-in function (Source: Figure created by authors). Figure 40: Feature importance bar plot based on the Random Forest feature importance built-in function (Source: Figure created by authors). Figure 42: MLR model's most important words for class Cardiothoracic & Vascular (Source: Figure created by authors). Figure 44: MLR model's most important words for class Emergency Department (Source: Figure created by authors). Figure 43: MLR model's most important words for class Dermatology (Source: Figure created by authors). Figure 45: MLR model's most important words for class Endocrinology (Source: Figure created by authors). Figure 46: MLR model's most important words

for class Gastroenterology (Source: Figure created by authors). Figure 48: MLR model's most important words for class Infectious Disease Specialty (Source: Figure created by authors). Figure 50: MLR model's most important words for class Neurology (Source: Figure created by authors). Figure 52: MLR model's most important words for class Oncology (Source: Figure created by authors). Figure 47: MLR model's most important words for class Hematology (Source: Figure created by authors). Figure 49: MLR model's most important words for class Internal Medicine Department (Source: Figure created by authors). Figure 51: MLR model's most important words for class Obstetrics & Gynaecology (Source: Figure created by authors). Figure 53: MLR model's most important words for class Orthopedic Surgery (Source: Figure created by authors). Figure 54: MLR model's most important words for class Primary Care (Source: Figure created by authors). Figure 56: MLR model's most important words for class Pulmonology (Source: Figure created by authors). Figure 55: MLR model's most important words for class Psychiatry (Source: Figure created by authors). Figure 57: MLR model's most important words for class Urology (Source: Figure created by authors). 1 2 3 12 14 16 18 25 27 28 29 31 37 44 46 49 50 52 53 54 59 69 72 73 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100