RobotDoc Documentation

August, 2023

Noel Kronenberg	**Matthieu Decotignie**	**Rasan Younis**
Tamer Aktas	**Sebastian Aust**	**David Fränkle**
Ulas Akyildiz	**George Badour**	**Feidi Kallel**
Emirhan Sarioglu	**Annabelle Kröger**	**Geri Capo**
Irem Kement	**Till Kurek**	**Joris Gabrisch**

Contents

1 Introduction

According to the WHO, 5 people worldwide die in exactly this minute due to wrong treatment, wrong medication or simply a wrong diagnosis. Mistakes are human. Therefore, it is time to support the often critical process of diagnosing diseases with models that do not make mistakes.

This is the overall goal of the RobotDoc project, to aggregate all medical knowledge in one application in order to diagnose diseases without errors and to suggest appropriate treatment.

Furthermore, RobotDoc not only gives us the chance to prevent treatment errors, but also to provide access to the still expensive healthcare system for people around the world who previously had no access due to their life circumstances. Therefore, we not only had fun and learned a lot for ourselves while contributing to the RobotDoc project, but hopefully it will also have a great positive impact on humanity one day.

2 Structure

2.1 Team

We decided to structure our team into smaller subgroups to tackle the larger assignments in digestible bits. They are divided into four high-level functional groups:

- Software Architects: general architectural and organizational structures
- Data Team: everything related to data crawling and parsing
- Machine Learning Team: work on the core machine learning algorithms
- Application Team: backend and frontend structure of the application

2.2 System

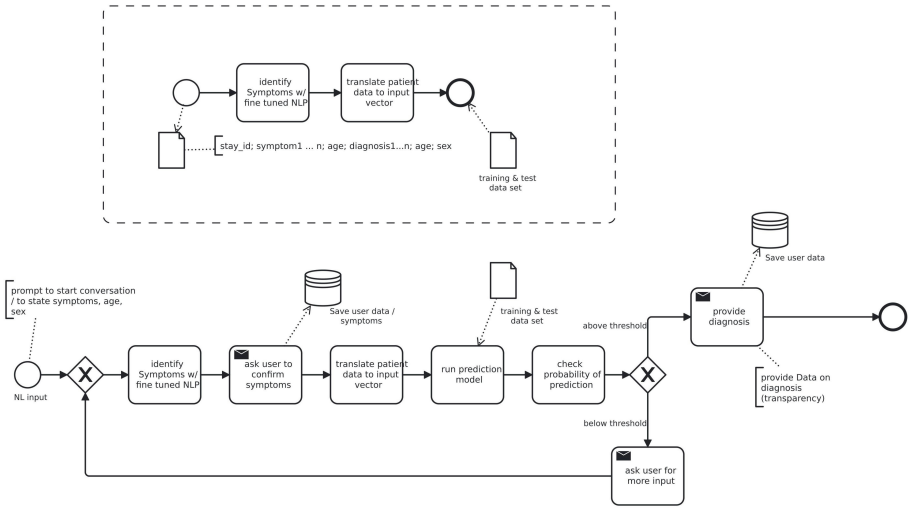

Figure 1: A broad overview of the system.

The first process describes the creation of the test & training data set. The initial data set was run through the system once to parse the data into the correct vector format.

The second process describes the running system. After being initiated by an input in NL, the system identifies all symptoms within the input message. A prompt can direct the user to more precise and usable inputs therefore overall increasing the user's experience. Identified symptoms get confirmed by the user, saved to the request / user file depending on GDPR, parsed into the

correct format and fed into the prediction model. If the prediction meets a certain threshold of accuracy the system provides the user the diagnosis. The knowledge graph is supposed to add transparency to this step by providing the user with an overview of the model's prediction process.

The KG could also be used in future versions as an interactive interface for the model's predictions and connections so that an expert / doctor could alter wrong predictions / data connections. Other Data connected with the prediction could also be provided to the user at this point.

3 Results

3.1 Data Team

As the Data Team, we had 3 bigger subtasks: Parsing, Crawling and Knowledge Graph. At the start, we were granted access to the MIMIC-III and MIMIC-IV dataset, which includes health-related data of patients admitted to intensive care units (ICUs). We chose the latter one, which was a more up-to-date version and contained more information about patients. During the project's timeline, our goals in chronological order were, understanding the MIMIC-IV dataset, preprocessing the data, supporting the ML Team and researching and creating a knowledge graph for the model.

3.1.1 Progress

The core task of the Data Team was to provide the ML Team with the correct Data for our Model to train on. Initially, a very basic data that maps symptoms to a diagnosis was found on the Internet for the ML team to design the first model. This data was very simple to work with in the beginning, so it made it easier to create a starting model. Due to a changing structure of the ML Model and limited access to the model this task relied heavily on close collaboration with the ML Team. During the project, the Data Team merged data from the original MIMIC IV data tables into one master table with all usable information. This final dataset included data that was related to the vital signs of a patient such as temperature, heart rate, resprate, o2sat, sbp, dbp and additional information about the patient's situation such as pain, acuity, gender and age which the model could not use in the final version. The main property that the training data of the model had to have was symptom and diagnose relation as the model's main goal was a symptom-to-diagnosis mapping. For this goal, we used UMLS (Unified Medical Language System) and ICD (International Classification of Diseases) codes. We realized that there were lots of symptoms and diagnoses in the data, which caused overfitting in the model. In order to reduce the dimensionality of feature space, we took only the most common 100 symptoms in the final dataset and used one hot encoding to show whether a patient has the symptom. Lastly, we parsed the data in a vector format for the ML Team to use.

Crawling:

We limited our crawling efforts to focus on our research for the knowledge graph and support ML with the correct Data in order to achieve a running system as early as possible.

We did not manage to find a data set comprehensive and large enough to replace MIMIC IV. Other interesting websites or data sets for future reference:

Amboss.com - most common learning software and dictionary for medical students of medical definitions, images of diseases and symptoms and tests to study. Could be used to further provide information (Pictures, First Aid treatments) about diagnosis in the future or test RoboDoc's effectiveness in medical exams. This could be achieved in collaboration with TutorAI to build on their work through effective knowledge sharing.

Knowledge Graph:

The initial phases of our research project revolved around developing a comprehensive Knowledge Graph. However, our team encountered challenges stemming from the restricted accessibility of resources on the subject matter, our team's limited familiarity with the domain, and an inherent lack of clarity in the structural design of our proprietary system.

Despite these obstacles, incremental headway was achieved in constructing a Knowledge Graph utilizing the NetworkX framework. The chosen dataset for this project was MIMIC-IV, and the methodology followed was similar to that of the PrimeKG model, which is linked for reference. Our efforts primarily centred on the explicit delineation of pertinent nodes and the relationships that interconnect them within the graph, opting for a transparent representation rather than embedding the information.

Interesting Sources: Medical KG (PrimeKG) paper and code.

3.1.2 Problems

The Data Team encountered two prominent challenges during the course of the project. Firstly, communicating and assigning our resources and work efforts correctly within the team and secondly sharing knowledge and collaborating with the other Teams efficiently.

The preliminary undertakings concerning the Knowledge Graph had initially constrained our assistance to the Machine Learning Team and its operational framework. Simultaneously, it also realigned our attention towards envisioning an unfamiliar prospective model prior to comprehending the extant model at hand.

Although the work was originally located within the Data Team, integrating a "feature" like the Knowledge Graph into the system should have been a cross-team effort. Intensive collaboration between the Data and the ML team would have been needed for the knowledge Graph to coexist with the currently running machine learning solutions.

The Data Team undertook a supportive role for a significant portion of the project, establishing a level of reliance on the Machine Learning Team. With the model's access restricted to a select few, predicting the necessary data adjustments to align with the ML Team's requirements proved to be a challenging attempt. As a result, formulating strategies and evenly distributing tasks within the Data Team encountered difficulties.

3.1.3 Evaluation

The Data Team fulfilled its supportive role for the ML Team. We failed to assign our work evenly at times and were often held up by planning, waiting on the other teams or simply a lack of communication. The Knowledge Graph resembles a more innovative result of our work but also took a lot of our time while not clearly understanding its role within the project in the early stages. It's an interesting new direction for future versions that also carries the danger of becoming a bottomless pit consuming most of the team's energy.

Meeting solely online during the project enhanced the difficulties regarding communication and planning in such a big team. Since most tasks within the data team but also within the whole RoboDoc Team were intertwined with each other it was difficult to work on topics individually which resulted in more planning and communication efforts. The team would have needed a person just responsible for team organization while still having enough time to work on and understand the project themselves. This could have been prevented by limiting the team's size or having more in-person meetings.

If we did the project today all over again we would need to have a person in charge of organizational tasks, accountability of team members for certain tasks, weekly small stand-up meetings with clear goals and an openly communicated prioritization of all tasks.

3.2 Machine Learning Team

3.2.1 Progress

As the machine learning team, we focused on orchestrating an interplay of NLP techniques, medical ontologies, and machine learning models to provide accurate medical diagnoses based on textual patient data.

In the NLP segment, we integrate a pre-trained model for NER tasks, based on SciBERT model fine-tuned for token classification to extract entities like age, gender, and symptoms from medical text. Fine-tuning consists of feeding the model a dataset, where each word is labelled according to the classification we aim to do.

We use a dictionary to store the patient information which we update after processing the NER model's output, making it easily accessible for further processing or prediction, also ensuring that the extracted information is complete and consistent.

To establish a seamless interaction between the NLP model and the prediction model, building a pipeline was essential. This was achieved using scispaCy, a Python package containing custom pipes and models for processing medical data. Preprocessing data was necessary, involving the conversion of data to UMLS code, as follows:

The UMLS is a set of files and software that brings together many health and biomedical vocabularies and standards to enable interoperability between computer systems. In our code, the customized nlp model loaded from scispaCy is a biomedical NER model that identifies medical entities in text such as diseases, symptoms, medications and more. In order to extract medical entities from the input text and map them to UMLS concepts we also need to load an entity linker that associates the recognized medical entities with UMLS concepts, providing standardized identifiers. This way the system can understand and work with medical terms in a standardized way.

3.2.2 Problems

Our progress encountered hurdles related to data availability and NLP training. The dataset "MACROBAT" surpassed SciBERT's token limit of 512, which required truncation that potentially leads to context loss. This posed challenges for NER accuracy and information extraction. Additionally, the maximum token limit of SciBERT impacted training data preprocessing, possibly leading to vital information loss.

3.2.3 Evaluation

Our project's evaluation reveals both accomplishments and areas for improvement. The NLP tool's performance depended on its ability to accurately extract age, gender, and symptoms from patient input. Metrics such as precision, recall, and F1-score could be used to assess the NER capabilities of the developed model.

Additionally, qualitative assessment could involve analyzing model outputs on real-world patient data to ascertain the correctness of extracted information. Addressing the challenges related to the truncation of training data is essential. Exploring techniques such as data augmentation, data selection, or context-aware tokenization could potentially mitigate the effects of data truncation.

This system can provide diagnoses based on the provided symptoms. However, we could identify limitations in accurately inputting symptoms due to non-expert user's' difficulty in using medical terminology.

3.3 Application Team

3.3.1 Progress

The application's current state fully encompasses all the essential core functionalities of the desired vision for the system.

We now have a hosted web-accessible application paired with an external database to store user data. A new user can easily create a persistent profile. After logging in, they have the choice between two types of assessments: general or personalized. The key distinction lies in the persistence of input data from the latter.

User interaction occurs through a simple and intuitive interface featuring text input and a slider to set the desired prediction confidence threshold. A user can enter their symptoms using natural language, after which the application sends this information to our natural language processing model. This model, in turn, forwards the identified symptoms to the prediction model for an actual diagnosis. The application then presents the diagnosis to the user, complete with a confidence percentage (of said diagnosis) and the option to remove incorrectly recognized symptoms from the input.

Finally, the user has the option to log out again. In case of an activated "remember me" token, the authentication will be kept for future sessions.

3.3.2 Problems

In the initial phase of our project, we encountered a considerable challenge in managing a database. Utilizing PostgreSQL posed complexities, especially when ensuring accessibility for team members during the application's offline development stage. Our solution came in the form of ElephantSQL, a complimentary online database service. Given its performance, we opted to leave the effort of hosting our own database. The capacity provided by ElephantSQL proved more than adequate for our requirements.

An essential part of developing the application was implementing the interface between the app itself and the database. For this purpose, we leaned on the Python ORM, SQLAlchemy, which serves as a bridge allowing SQL transactions to be executed seamlessly through Python

code. Flask, our framework of choice, also provides a little simplified usage of the latter. Despite this, we found the native documentation of SQLAlchemy (as Flask offers one of its own) to be indispensable for implementing a flawlessly working interface. Unfortunately, SQLAlchemy turned out to be a thorn in our side from start to finish of the project.

3.3.3 Evaluation

Upon reflection, we recognized that our approach might have parted from the core principles of 'agile' development. Introducing more regular, short-term deadlines might have accelerated our progress. We dedicated considerable attention to our milestone meetings, which, while valuable, might have overshadowed other priorities.

Team collaboration was an area where we experienced relative success. Efficient task distribution was mainly done by the Kanban Board, especially in the project's early stages. However, it's worth noting that our reliance on this tool diminished over time.

From the project's inception, a primary objective was to maintain a structured and organized directory. Through consistent adjustments, both minor and substantial, we believe we've achieved this aim commendably. The compact size of our subteam was advantageous, streamlining discussions and decision-making processes.

In terms of design, we're confident in our selection. It ensures clarity in feature presentation while also setting a consistent theme for future enhancements. Sad to say, we did not quite finish all of our goals but are content with the foundation we laid.

4 Outlook

4.1 Data Team

One of the things we would like to improve is finding datasets that allow for more accurate and reliable prediction models both for disease prediction and for treatment suggestion or finding a way to use the MIMIC IV data with increased size to train the model in reasonable time and memory with accuracy gain.

The knowledge graph adds a unique dimension to the project, providing a structured repository of the medical information - in a Network form. This graph could serve to enhance symptom descriptions and diagnoses. The AI's diagnoses, in turn, are expected to be comprehensive, detailed, and supported by evidence drawn from the knowledge graph. The project's success hinges on a robust evaluation and comparison strategy. Metrics could be established to measure the accuracy, relevance, and user comprehensibility of diagnoses. Additionally, human evaluation through user studies could offer a comparative analysis of the AI's performance against expert diagnoses and measure user satisfaction. Considering future directions, the project could expand to serve as a medical search engine bot, providing user's with relevant medical information beyond diagnoses. Techniques like graph batching and node pooling could be implemented to optimize the efficiency of graph-based computations. Active learning strategies could enhance diagnostic accuracy by prompting user's for additional information when the AI's confidence is low. To foster trust and transparency, interpretable AI methods could be developed to elucidate the reasoning behind diagnoses.

4.2 Machine Learning Team

At the moment the prediction model requires a description of the patient's symptoms as an input to make a diagnosis. The problem here is that normal people without a medical background normally don't know the technical terms to describe certain parts of their body, which makes an accurate diagnosis difficult. Looking at a medical examination nowadays, one solution to tackle this problem could be an instruction given by the application to put pressure on different places near the hurting and therefore locating the exact spot of the pain instead of having to describe it.

Ultimately this could look like you just have to film the part of your body that is hurting with your mobile device, the application automatically detects the body part e.g. left knee and then shows the different spots you have to palpate via augmented reality. Then you just rate the pain level at every spot and add other symptoms that are not related to pain in a particular part of your body and the symptoms should be way more accurate than trying to describe them without having knowledge of the proper language.

The implementation of the visual part could probably be done in cooperation with the BigBrother project. But to get the knowledge of how to palpate different parts of the body correctly and then draw the right conclusions from doing so, the support of some medical professionals is definitely necessary for this purpose.

4.3 Application Team

Looking back, we have faced a few struggles with database interactions. To make lives easier, it might be a good idea to split the database work from the main code. While we have made some progress in this direction, there is more ground to cover. By setting up a consistent way for our app to communicate with the database, it will not only have improved scalability but also be friendlier for any future students working on it.

On another note, our app currently lacks some key features in terms of admin and user management. Adding an 'admin panel' should fill this gap nicely. This would make tasks, like password resets, more straightforward without having to deal with native database tools.

Looking a bit further into the future, data protection is, especially in the field of diagnostics, of course very important. To this point, anyone with access to the database is able to read everything, encrypted passwords being the exception. Configurations are to be made in a way, that in a productive environment, personal data is protected adequately.

In cooperation with the data team, you also can work on a feature for doctors to correct and enhance the knowledge graph. By that falsely learned correlations can be eliminated, and missed ones can be connected.