Peining Yang, Erika Fox, Yaze Gao, Nancy Huang
Automatic Extraction of Patient Problems with Clinical Notes Text

Clinical Notes Extraction and Evaluation
Capstone: Duke Health and OIT

Abstract

This project aims to make use of Natural Language Processing techniques to take Electronic Health Records and translate them into a set of features fit for prediction in order to more efficiently and more accurately track the progress of medical students. This paper details a methodology pipeline to do this, consisting of the following steps: Labeling, Extraction and Detection, Entity Recognition, and Feature Engineering. These steps led to a feature set estimated to be an accurate reflection of 83.3% of the total collection of EHRs provided. This paper concludes with a discussion of how this feature set can be used to predict diagnosis codes for some of the patients, as well as how this work can eventually lead to an automatic system for generating summaries of medical students' completed/yet-to-be-completed requirements.

Introduction

The booming development of data science has had an enormous influence on healthcare. Data science can improve healthcare quality by helping doctors understand clinical situations more thoroughly. Machine learning algorithms have significantly contributed to the early detection of numerous diseases[1]. Data science can also benefit medical education, which is the focus of our Capstone.

Duke Health is one of the leading medical schools in the country. Its mission is to provide high-quality education to medical students by strategically optimizing their training experience during clinical rotations. Duke Medical School is a pioneer in the country where medical students can write Electronic Health Records (EHR) during their clinical rotations. These clinical

notes also help the medical school keep track of what kind of medical encounters their students have completed and have yet to do before they graduate. However, reading each EHR is not only labor-intensive but also allows a lot of room for human error. To tackle this issue, Duke Health has been collaborating with the Duke Office of Information Technology (OIT) to create an application that can access a student's clinical notes, along with visual summaries of what kinds of encounters they have already accomplished and have yet to complete. This task of summarizing and evaluating clinical notes is deceptively complicated and difficult. Currently, the team must use a completely unstructured text string for each clinical document, sometimes referred to as a bag-of-words. These bag-of-words do not make helpful inputs into the machine learning model the application requires to do its job. Therefore, there is an urgent demand for Duke Health to convert clinical notes into structured data. In other words, convert the bags-of-words into a data frame with multiple, well-processed features that a model can combine to make accurate predictions on what type of encounter a clinical note represents.

Luckily, most clinical notes are written with a standardized structure in mind that we can work to separate. Namely, SOAP (stands for Subjective, Objective, Assessment and Plan) is a commonly used documentation form. These notes also tend to have what are referred to as "diagnosis codes" or, more formally, "ICD-9 medical codes", which are short phrases marked with a pound (#) symbol in the Plan section of a note. Doctors use these diagnosis codes to indicate a patient's issues and classify the problem group (type of encounter) to which a particular note belongs.

Background

Previously, methods such as Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN) have been used to do similar work. Much of this work focused on the ICD-9

medical codes, where researchers have tried to map free-text clinical automatically[2]. Others parsed notes to extract data into structured form with the help of regular expressions[4]. Finally, some previous success has been achieved by processing text with the clinical Text Analysis and Knowledge Extraction System (cTAKES)[5]. This is a comprehensive, robust, and open-source NLP system designed to process and extract semantically essential clinical information from EHR. Each of these will inform our approach to this problem.

Methods and Results

Pipeline

We think of the tasks for this project in the structure of a pipeline. This pipeline consists of labeling, extraction, entity detection, and feature engineering, in that order. The following describes the details of each of these.

A. Labeling

Electronic Health Records do not look the same for every medical department, which means the way we work with them won't be exactly the same either. Because of this, we need to make sure that each of our health records are mapped so we can properly treat them accordingly. However, our data does not come with one concrete field that indicates a note's problem group, but instead it has three fields that hint at it: "title", "admission_service", and "specialty". Most records are missing values in at least one of these categories, so there isn't really one of those three that could stand on its own. However, none of them are missing a value in all three, so it is possible to combine them in order to give each record a value. We worked with our client to determine a hierarchy

for how informative each field is. Together, we decided that we would create a new column in the data for the official specialty type, and we would start by assigning the value in that column to the same value in “title”, as long as that value was not null. If the value in “title” were to be missing, we turn to the value in “admission_service”, and if that value was also null, we would use “specialty”. We defined success for this task as having a type assigned to each record in our data, which we were able to achieve by a series of simple statements in our code where we utilized the where() function from python’s NumPy module. However, once we achieved this original goal, we discovered that more work was necessary, as there were several cases that needed to be handled, such as differing names for the same thing (i.e. Ear/Nose/Throat and Otolaryngology are the same problem group but they are listed differently like this in the data), or inconsistent notation (i.e. Pediatrics vs. PEDIATRICS). Once we made these adjustments, we were left with 87 different encounter categories.

B. Detection and Extraction

As described earlier. Electronic Health Records are composed of four components referred to as SOAP, which stands for Subjective, Objective, Assessment and Plan. Subjective refers to the patients’ history, Objective refers to diagnostic data, Assessment covers potential problem diagnoses, and Plan refers to proposed treatments/courses of actions. This final section, Plan, includes the previously introduced diagnosis codes (ICD-9 medical codes) marked with a pound (#) symbol . However, these components were not easily accessible to us with the format that our data came to us in. We received all of the electronic health records as unseparated freeform text, so if we wanted to take

advantage of the SOAP components to help us find meaningful separation, we would need to do this ourselves.

Our methodology for doing this separation consisted of four stages, detailed below.

Stage 1: Filtering

We quickly discovered that some notes clearly ignore the standard SOAP format, and therefore should be removed for this process. The majority of the notes that fell into this category were COVID-19 vaccinations. There were also quite a few notes from the OBGYN problem specialty, including visits such as Obstetric Antepartum/Postpartum Discharges, Fetal Heart Monitoring, Labor Progress Updates, Newborn Summaries, Studies, and Clinical Skills Foundation Notes. There were a few other encounter types that we found *often* did not follow SOAP (as opposed to previously described categories where the notes *always* ignored SOAP). However, we decided to leave them in our data pool with hopes of extracting the few that did follow. Many of these were also in the OBGYN category. After this filtering, we were left with 69.34% of the total pool of medical notes.

The following three stages made up somewhat of an iterative process, in that they were repeated until we achieved what we believe is our final and maximum extraction rate: 88.11%.

Stage 2: Detection

To extract each component, we first needed to be able to locate them, which we did using Regular Expressions (RegEx) and by considering the logic of the medical field. We designed our RegEx expressions to optimize extraction, which means trying to avoid false positives. This is described in more detail in the Stage 4 section.

The following table describes the phrases we used to detect each component. In our code, we accounted for these phrases in upper case and lower case, and sometimes a combination of both.

Subjective	Objective	Assessment	Plan
● Subjective ● Interval Hx:/ Interval Events:/ Interval History ● HPI/ History of Present Illness ● Hospital Day:/ Hosp Day ● FIRST_NAME_FULL	● Objective ● Examination/ Exam ● Physical Exam ● Current Vital Signs ● Vitals	● Assessment (cannot be preceded by “Safety” or “General”)	● Recommendations ● Plan

While determining this list of phrases, we discovered that often, Assessment and Plan are grouped together and treated as one component (about 67% of the time, to be exact), so here are the phrases we used to find those:

Assessment/Plan combination

- Assessment and Plan
- Plan and Assessment
- Assessments and Plans
- Assessment & Plan
- Assessment/ Coordination of Care
- Assessment and Recommendations

With our regular expressions, we constructed a column in our dataframe for each component that saved its beginning index, which we were then able to use for extraction/separation.

Stage 3: Separation

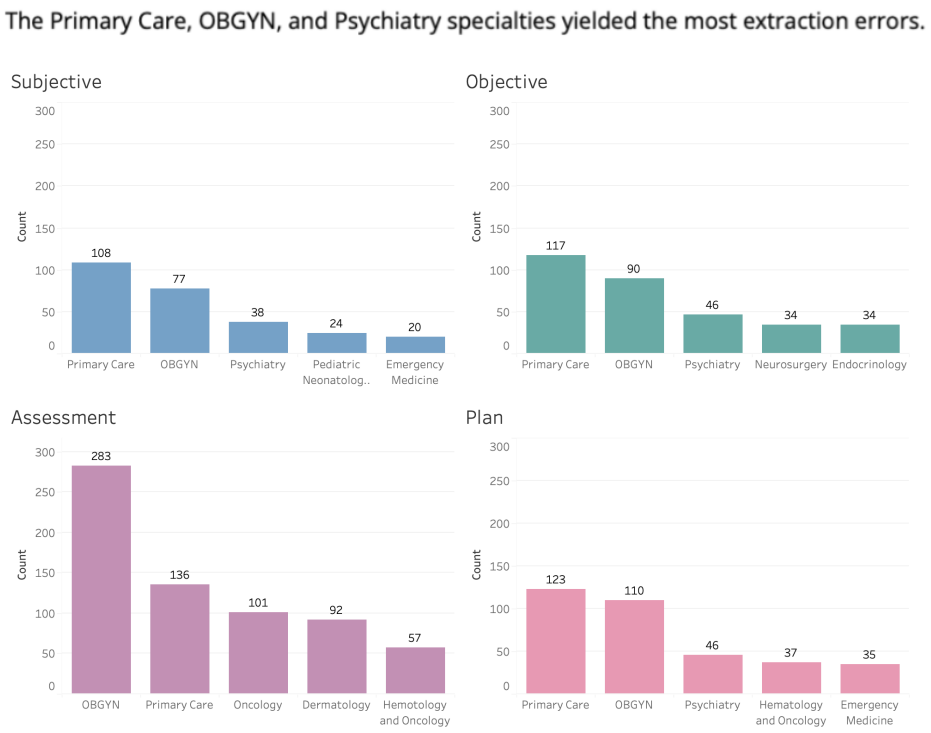
Our extraction methodology assumes that the four major SOAP components are present in the notes, so for this step, we filtered out the notes we found to be missing one or more. Using the indexes provided by the detection, we proceeded to separate the detected sections, following the assumptions that 1) the end of a section is when a new section starts, or the note ends 2) the components are in order (Subjective first, Objective second, and so on). These assumptions allowed us to make feature columns for each of the SOAP components by assigning each as a substring of the original note. Namely, each component would become the substring that starts at the index of the phrase we used to identify that component, and ends at the index of the phrase we used to identify the component we expect next (or just captures the rest of the note if it is the last component).

However, once this was done, we found that our second assumption, where we expected the SOAP components would be in order, was erroneous. So then, we adjusted our approach to create a column containing a dictionary of all that note's

components mapped to its corresponding starting index for each note. From there, we converted these standard dictionaries into 'OrderedDict's (imported from collections), which allowed us to put the components in the correct order, which then let us assign a proper ending index to each component for extraction.

Stage 4: Analyzing Failures

This step consisted of evaluating our performance on the previous two stages and looking for opportunities to improve.



The above visualization provides an example of how we might have found shortcomings in how we were detecting components. For instance, if we saw that

this chart displayed an abnormally high number of failures in a certain problem group, that would prompt us to take a closer look at that type of note, which would sometimes lead to a new discovery of how a particular problem group referred to a component (i.e. this helped us find that NEUROSURGERY used “Hospital Day:" to indicate subjective, so we added this to our detection schema). Here, one can see that PRIMARY CARE and OBGYN have consistently high failure rates across all four components. Even after extensive work to improve detection over these categories, we found them to be the most irregular in terms of formatting.

Another way we would find failures is to analyze instances where one of the components came out to be an empty string or a suspiciously short string. This failure was a result of a false positive in the detection stage. In other words, because we assume that the end of a component section is the same as the start of another, we would sometimes end up with a blank component if we incorrectly detected the start of a category. An example of this is that we originally found that the anonymized age of a client, formatted as X y.o. (blank years old), would sometimes mark the beginning of the Subjective section. However, we discovered that sometimes this same phrase would be used early on in the assessment section, like so: “Assessment: patient is X y.o. …”. Therefore, if we used “X y.o.” to detect Subjective, it would fail here and the assessment section would work out to be simply “Assessment: patient is” (not to mention that Subjective would be completely incorrect). Findings like this would prompt us to remove “X y.o.” from our detection schema.

The following are the final detection rates for the four components of SOAP.

Subjective	97.9%
Objective	96.9%
Assessment	93.5%
Plan	96.5%

Additional Separation

Having each of the SOAP components isolated made it more possible to locate the information that is the most useful for determining what kind of issues the medical students have encountered, which is what is often referred to as the “Physical Exam Constitutional”(PEC) section. This is the part of Objective where medical professionals include a detailed summary of the relevant patient’s symptoms. This is the part of the note that we used as input for the following “Entity Recognition” section.

The process for extracting the PEC was very similar to the one detailed above, but instead of separating the entire note into Subjective, Objective, Assessment and Plan, this time we were separating Objective into its three parts: “Vitals”, PEC, and “Labs and Medications”.

This is also when we extracted the “diagnosis codes” out of the Plan component, a.k.a. patient problems marked by pound(#) symbols.

The rate of which we were able to extract these are given below.

Physical Exam Constitutional	94.5%
Diagnosis Codes	48.5%

C. Entity Recognition

The physical exams category separated from “Objective” discussed in the previous section contains the symptoms a patient is presenting or not presenting. To help educators evaluate whether the medical students are making the correct assessments given the patient’s physical exam results, an NLP model was constructed to perform a Named Entity Recognition (NER) task that will detect the location of the symptoms and handle the negation of symptoms in the notes. The primary source used in this step was SpaCy, a free open-source library for NLP in Python. We applied the NER model pretrained on a biomedical corpus from SpaCy to the data and iteratively revised it to improve the fit.

Figure 1 shows the entire pipeline of this detection model. The first step was to preprocess the unstructured text so that it was capable of being understood and analyzed by the machine. The two main preprocessing techniques used in this pipeline were tokenization and lemmatization. Tokenization was to break the text into meaning units known as tokens. In our case, each word in a sentence would be a token. Lemmatization was to identify the root of a word. For example, “runs,” “ran,” and “running” would all be converted into “run.” The NER task was then applied to the processed data to detect and label “entities” in a document. In this case, an entity would be either a symptom or a

disease. The pre-trained spaCy model already had a list of entity patterns that the model was able to detect. However, some symptoms that appeared in our notes were not on this list. Hence, these patterns were manually added to the entity list so that when refitting the model, it was able to detect the customized patterns. While clinical notes document whether a symptom exists in a patient, it also documents its absence. It is important to recognize the negation cases in our data as they can often pose a challenge to natural language processing algorithms. For example, the symptom in the note could've been written as "no heart murmurs", to which a simple search of whether "heart murmurs" exists in the clinical notes would likely result in misclassification. Therefore, our last step in the detection model pipeline was to correctly identify the negative existence of symptoms in a patient. We utilized the NegEx package from SpaCy which covered various negation cases including preceding negations like "no" and "not", following negations like "unlikely" and "declined," pseudo negations like "no further" and "without further," and terminations like "but" and "although." If the negation patterns were detected, then the entities would be identified as "negative entities."

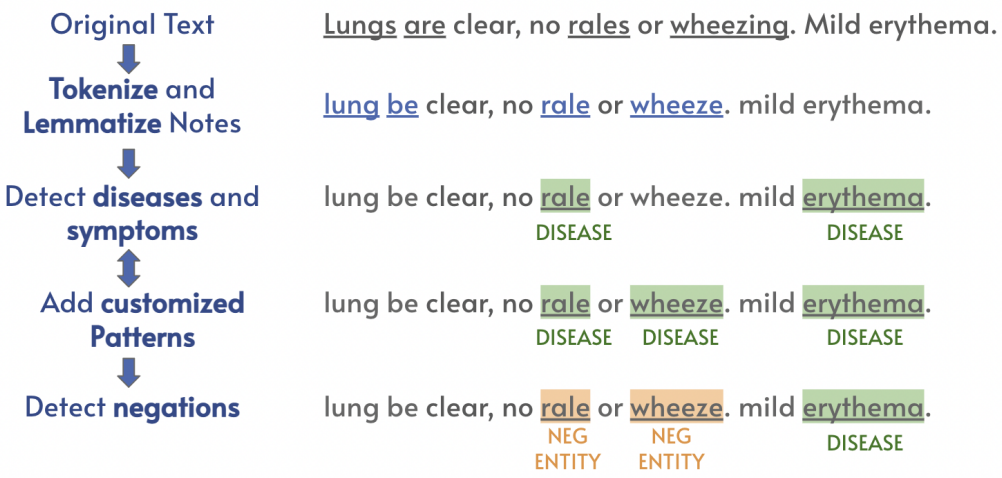


Figure 1: Detection Model Pipeline and Example

The final product of this model was a visualization showing the keywords it detects as shown in Figure 3. The visualization was very important to our project because it enabled our client to extract the key information from a medical student's note which then makes the evaluation of a medical student easier.

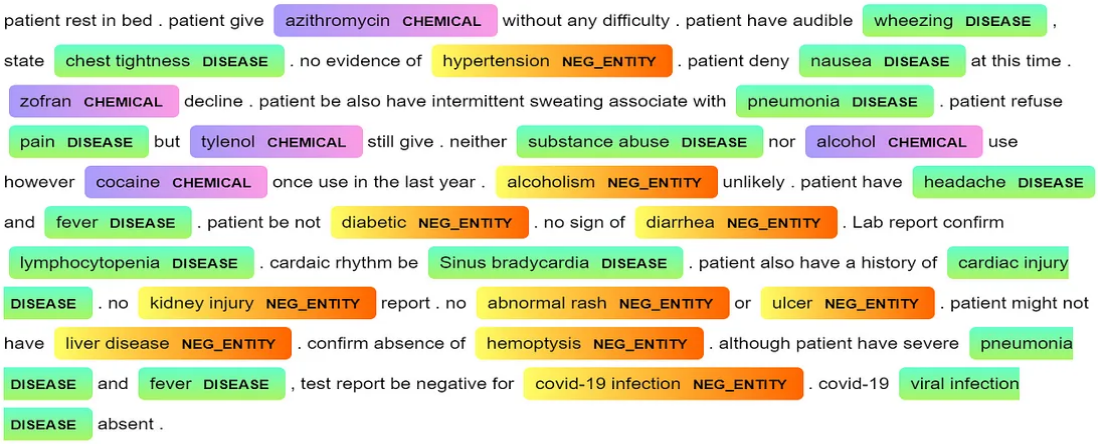


Figure 2: Detection Model Visualization

Although the ground truth was not available, we were able to randomly pick around 50 notes to estimate the accuracy of the model. The model was able to identify most of the diseases and symptoms correctly. For those that couldn't be identified, we manually added them to the entity list and refitted the model. Through this iterative step to fit the model and add patterns, we eventually got all entities in the selected notes successfully detected by the model. In this procedure, we also noticed that some general entities from the original list like "pain" or "weakness" were not very useful features. For example, "not assessed due to weakness" appeared multiple times in several notes, but "weakness" here did not refer to any specific symptoms or diseases. We ended up removing those terms to make our detected entity list more concise.

The model was also very accurate in terms of detecting negation cases. However, when the note had negative terms in the previous sentence but it did not end with a period, the model would get confused and identify the entity in the next sentence as a negative entity. This was not a very common issue so overall the model was good enough to detect both positive and negative entities.

D. Feature Tables

Once our model was able to generate a list of symptoms or diseases for each note, we were able to convert these extracted features into an indicator table that shows the presence or absence of each entity for every medical note using “1” or “0”. An example table is shown below.

Note number	Weakness	Bleeding	Wheeze	Adenopathy
1	1	1	0	0
2	0	1	0	1

The issue we initially encountered with our feature extraction was the vagueness in the entities extracted. For example, the identification of “weakness” is rather useless unless it is localized to a body part. Therefore, we further investigated adjective-noun tagging, and were able to allocate a specific system for each symptom/disease.

Note number	Weakness (Motor)	Bleeding (Abdomen)	Wheeze (Respiratory)	Adenopathy (Cervical)
1	1	1	0	0
2	0	1	0	1

When generating this table, we took note to remove any duplicate features within individual notes. This resulting table can be generated for each student's portfolio of medical notes, which could then produce a summary report of their academic progress.

E. Future Work: Predictions

Although our client is pleased with our resulting feature table and how it will be able to help with their problem, we can't help but look for ways to improve. We previously mentioned that only 48.5% of the notes contain diagnosis codes. However, our newly created feature table may be exactly what we need to fix this, as the data is now in a format where modeling is more realistic. The notes containing diagnosis codes can be treated as the training set and the rest of the notes can be treated as the testing set. The input of the model will be the indicator table of diseases and symptoms; and the output of the model will be the diagnosis code. Because the "test set" here does not have a diagnosis code to begin with, this means we won't have a ground truth to use to assess the accuracy of this model. Instead we will have to work with our client in order to determine if the diagnosis codes our model assigns to patients are reasonable.

References

1. Narasimman, P. (2022, June 12). *Data Science in healthcare – applications, roles and benefits* . Data Science in Healthcare – Applications, Roles and Benefits . Retrieved October 19, 2022, from https://www.knowledgehut.com/blog/data-science/data-science-in-healthcare
2. Huang, J., Osorio, C., & Sy, L. W. (2019). An empirical evaluation of deep learning for ICD-9 code assignment using Mimic-III Clinical Notes. *Computer Methods and Programs in Biomedicine*, *177*, 141–153. https://doi.org/10.1016/j.cmpb.2019.05.024
3. Omer, M.-A. (2021). Extracting Structured Data from Free-Text Clinical Notes: The impact of hierarchies in model training. *KTH Library*. Retrieved September 22, 2022, from urn:nbn:se:kth:diva-305568.
4. C. Flint, A., Melles, R. B., Klingman, J. G., Chan, S. L., Rao, V. A., & Avins, A. L. (2020). Automated extraction of structured data from text notes in the Electronic Medical Record. *Journal of General Internal Medicine*, *36*(9), 2880–2882. https://doi.org/10.1007/s11606-020-06110-8
5. Savova, G. K., Masanz, J. J., Ogren, P. V., Zheng, J., Sohn, S., Kipper-Schuler, K. C., & Chute, C. G. (2010). Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association : JAMIA*, *17*(5), 507–513. https://doi.org/10.1136/jamia.2009.001560