Machine Learning for Healthcare Final Project 1–14

OpenNotes: Medicine Explainer

Vibha Agarwal AGARWALV@MIT.EDU
Massachusetts Institute of Technology
Cambridge, MA, USA

Shinjini Ghosh SHINGHOS@MIT.EDU
Massachusetts Institute of Technology
Cambridge, MA, USA

Stuti Khandwala STUTIK@MIT.EDU
Massachusetts Institute of Technology
Cambridge, MA, USA

Anil Palepu APALEPU@MIT.EDU
Massachusetts Institute of Technology
Cambridge, MA, USA

Editor: Shinjini Ghosh

Abstract

The OpenNotes system aids in making clinical data and notes easily accessible to patients. However, there still exists a significant knowledge gap resulting in patients being unaware of their conditions and the medical purposes of their treatments. We employ machine learning and natural language processing techniques to develop a system which extracts useful information regarding a patient's medications and conditions from their discharge summaries, and then utilises external resources to help better explain their treatment plan and motivations, with an easy-to-use interface, focused keywords, and comparison with other similar cases.

1. Introduction

The OpenNotes system makes clinical data and inpatient clinical notes easily accessible for patients to review. However, patients are often prescribed multiple medications at once, and may not fully comprehend the medical purpose or treatment potential behind each of them, leading to decreased adherence and medication efficacy. Bridging this knowledge gap is an important problem in machine learning and healthcare, and would help make prescriptions more transparent, intelligible and comprehensible for patients, in turn increasing their adherence to the medication plan.

Automated addressing of this problem is challenging because it involves taking human created discharge notes, which are not necessarily in computer-readable, or regulated formats, and accessing a plethora of external resources based on the former, which all need to work in conjunction for the medications and conditions to become explainable. It is at this juncture that we can take aid from machine learning techniques to help automate this process, and further the clarity of medications to patients.

Kim et al. (6) have delineated how Electronic Health Records (EHR) data, primarily created for non-research purposes, has expanded its utility in recent years under the influence

of artifical intelligence techniques, but still remains far from being a regulated, complete EHR, and its applications remain limited so far.

We start with a singular disease example and illustrate the importance of treatment explanation. Once diagnosed with a serious condition such as hypertension, patients are often prescribed several medications to treat their condition, such as diuretics, which remove excess water volume from the vasculature, ACE inhibitors, which block angiotensin from causing vessel constriction, and beta blockers, which help the heart beat slower and less forcefully (1). Clinicians use their best judgement in prescribing these, alongside many other potential drug classes (for this condition, and various others for other conditions), and are generally very effective in managing the majority of high blood pressure cases. In fact, Bronsert et al. (2) demonstrated how the use of an anti-hypertensive drug results in an average systolic blood pressure reduction of over 12 mmHg, which is around 10% of the normal systolic blood pressure. However, despite pharmacological treatments being relatively effective, many patients fail to see significant improvement after starting on such medication. While a small fraction of these cases may truly be medically refractory, a larger percentage of such cases are due to suboptimal adherence. In this specific example, adherence with pharmacotherapy for hypertension a year after initiation is seen to be reported at under 50% Burnier (3). Encouraging the rest of the patients to follow their treatment and medication plan could see crucial improvement on their health, and explanation of medications and conditions is a vital step in the same.

With the new OpenNotes system, inpatient clinical notes are now easily accessible for patients to review—in particular, their hospital course, diagnoses, and prescribed medications are typically detailed in their discharge summary. Patients are often prescribed multiple medications, with limited or no explanation on their functionality. We see an opportunity to fill in this knowledge gap. We believe we can help patients with greater adherence to their medication course by helping them understand their conditions, the rationale for each medication in the discharge summary, as well as what is to be expected in terms of side-effects and condition improvement. The same framework can be extended to a variety of medical treatments to boost patient trust and understanding.

In this work, we utilise machine learning and natural language processing techniques to extract useful information regarding a patient's medications and conditions from their discharge summaries, and then utilise articles from medical websites to display information regarding the same to a patient, helping better explain their treatment plan and motivations. For every medication that a patient is prescribed, we create a report of the medication's utilities and effects, focusing on keywords that may be easy for a patient to remember. We also create a report for how each medication can improve a patient's condition in comparison with patients with similar conditions, symptoms, and medications.

Generalizable Insights about Machine Learning in the Context of Healthcare

- Suboptimal performance in one stage can be remedied by later stages of an algorithm. In this application, the discharge summaries were highly diverse and therefore difficult to process, resulting in many messy looking medication and condition strings. We make up for this by leveraging robust search engines which allow us to find relevant descriptions despite these flawed medication and condition strings.

- We can collect new data and leverage data from sources outside our dataset to gain knowledge about our data. Rather than trying to discover correlations within the MIMIC-III data, we utilize external sources, such as Medlineplus, WebMD and even Wikipedia articles, to better understand the purpose behind medications prescribed to patients, and to mobilize it for our application.

- A model that is useful for practical purposes does not necessarily need to always be perfect and comprehensive. We might be missing certain keywords for some conditions, however in our end goal of helping a patient understand why they were prescribed a certain medication, we can still significantly bridge the missing knowledge gap, even in the absence of some keywords. In this case, we follow the policy of something being preferred over nothing.

- Currently, our model returns these keywords as the final output. In other projects, we can utilize these keywords as intermediary tools. For example, we could potentially use these keywords to featurize medical notes data for prediction or clustering problems, or use it to quantify similarity between medications and conditions.

2. Related Work

Patients are generally eager and curious to learn more about their conditions and medications, and tend to conveniently look up medical jargon on the internet. However, we see in (8) how randomly discovered sources of information can be misleading or inaccurate. Moreover, looking up keywords may yield incohesive information about a patient's overall health, drugs and disease interactions, as any search query is subject to a search engine's algorithms yielding the correct results. Hence, it would be of immense value to offer medically sound and trustworthy information for a patient to be able to refer to. Moreover, as mentioned above, adherence to a medication plan is of utmost importance to a patient and an important factor in the efficacy of treatments in actually alleviating the disease (for example, as is mentioned in Burnier and Egan (4). In a study surveying OpenNotes participants DesRoches et al. (5), it was indeed found that having access to their online records indeed improved patients' adherence to medication.

Moreover, as mentioned before with regard to (6), electronic health records (EHR), while more abundantly and easily accessible than before, were created with non-research purposes in mind, resulting in a plethora of noisy and incomplete data. While the machine learning and natural language processing arenas for parsing and real-world usage of EHR records are rapidly evolving, they are still of limited practical application.

Additionally, casual inference of medications on health outcomes on a per-population basis is an evolving field in the age of personalised and decision-based medicine. Several works (Yazdani and Boerwinkle (9), Moura et al. (7)) highlight this in the context of various diseases. Developing a system which can link the effects of specific medication on various conditions in a specific group of patients is hence of crucial significance in furthering this field.

3. Methods

In this section, we outline the various steps in our data pipeline—from extraction to model building, prediction and evaluation. We go over the query used to extract discharge summaries, the logic behind obtaining a patient's medication and condition information from the dataset, web scraping to gather the medication description, side effects and so on, and finally, natural language processing methods for keyword identification.

3.1. Data Selection

Google BigQuery was used to extract data from the MIMIC-III database. The query used is as follows:

```
SELECT p.subject_id, a.hadm_id, p.expire_flag, n.category, n.text
FROM physionet-data.mimiciii_clinical.admissions a
INNER JOIN physionet-data.mimiciii_clinical.patients p
    ON p.subject_id = a.subject_id
LEFT JOIN physionet-data.mimiciii_notes.noteevents n
    ON n.hadm_id = a.hadm_id
WHERE n.category is not null
AND p.subject_id < 2000
AND p.expire_flag = 0
AND (n.category = 'Discharge summary' or n.category = 'Discharge Summary')
GROUP BY p.subject_id, a.hadm_id, p.expire_flag, n.category, n.text
ORDER BY p.subject_id, a.hadm_id
```

Notably, discharge summaries, which are typically unique to a `hadm_id` and identified by the `noteevents` category, were extracted for all patients that had them. Patients that died during their hospital were not included, since they would have no use for a discharge medication explainer. Additionally, the size of the extracted data was artificially limited by selecting patients whose `subject_id` was under 2000. This was done to simplify storage and testing, but the following methods could be seamlessly applied to all patients from the MIMIC III database with discharge summaries.

3.2. Medication & Condition Extraction

A list of discharge medications and discharge conditions was extracted from each discharge summary. We used two helper functions to get the medications and conditions from each row, and given a patient ID, we then proceeded to extract medications and conditions for every row pertaining to them, and concatenated them up at the end.

For the medications, the first step was to identify the relevant section of the discharge summary. This was done by searching for one of several possible strings that indicated the start of a relevant section, such as "discharge medications" or "medications on discharge". From this index, we search the rest of the text for an indicator of the end of the relevant section, such as "Discharge" or "[**", and then extract the relevant section.

The next step is to parse each line of the selected text, and extract just the medication without any extraneous details that may be present in the text. To do this, we make the assumption here that any valid medication will have some numerical/unit information on dosage in the notes, and use regex to find and extract text that meets this assumption. We also assume that each line will only have one of these matches, and only extract the first match per line.

We follow a very similar process to extract condition information, with some minor adjustments. Notably, the conditions are very diverse in how they are recorded, so we simply extract the line up until the first punctuation or end of line, without any additional requirements on the formatting unlike with the medications. In the case where medication or condition extraction fails, the corresponding list is empty.

3.3. Web-scraping

To get the descriptions and other relevant information for our list of medications and conditions, we leveraged the cloudscraper and beautifulsoup packages in Python. Despite our best efforts, the medication and condition strings from the prior step were riddled with errors, often with extraneous information, and sometimes completely incomprehensible. To remedy this, we leveraged robust searching from Google and the Medlineplus website to allow us to find information on a medication or condition even if the string we started with was very messy. In the cases where this still fails, we conclude that the string is likely not a medication or condition at all, and we simply remove it from our list.

The web-scraping process is as follows. For each medication, we first save the html data from a search of Medlineplus, refining the search by excluding all but the drugs & supplements results. We iterate through each resulting link, until we find one that meets various requirements that were manually determined through trial-and-error. These include requirements like the URL containing the string "druginfo" and the page name not containing the word "injection" if the medication was not an injection.

Next, we enter the chosen page, and grabbed the relevant sections: for medications, this is the drug name, other brand names, description of the medicine, and side effects when available. If the medication string does not ever appear in the selected Medlineplus article, potentially indicative of a very messy string or simply an incorrect link being chosen, we instead opt to use the far more robust Google search. We will also search Google if we didn't initially find any Medlineplus links matching our requirements or if the description extraction fails.

If we end up searching Google, the web-scraper grabs the first WebMD drug page in the search results, and extracts the medication description from that page. We search Google rather than WebMD because our web-scraping tool is unable to bypass cloudflare security measures when using the url results from the WebMD search. If this process still fails, the medication is removed from the patient's list of medications under the assumption that all proper medications will be found on WebMD.

The process for conditions is the same as for medications, except that we are looking at the health topics section in Medlineplus, and we do not impose any additional requirements for a valid URL.

3.4. Keyword Identification

For each medication, we hope to provide the patient with some understanding as to how it will treat the various conditions they suffer from. We attempt to achieve this by generating a short list of keywords for each medication-condition pair.

These keywords are words that are present in both the medication and condition descriptions, and to be useful, these words should be medically relevant to the patient's condition. To achieve this, we first extracted the nouns and proper nouns from the text, using the nltk package in Python. We filtered for nouns & plural nouns, both singular and plural, and threw the rest out.

We then generated 3 separate corpora utilizing the webscraping tools described in the previous section. The first was a list of every medication description available on Medlineplus, 1604 descriptions, which we got by extracting every unique URL from the A to Z list of medications on Medlineplus, and applying our medication description web-scraping code. We also scraped a dataset of condition descriptions from Medlineplus by extracting every unique URL from each body/systems page on the health topics section of the website, and applying our condition description web-scraping code. The final corpus was generated by taking advantage of the "Special:Random" function of Wikipedia to scrape the first 150 words of 5000 random Wikipedia pages. All of these corpora were passed through the nltk noun filter.

To select keywords for a given medication-condition pair, we first extracted nouns that appear in both our medication and condition description. Then, using the corpora we generated, as well as the `CountVectorizer` function from scikit-learn, we calculated how many times every word appears in each description. For each corpus, we calculated what percent of descriptions contained each word at least once, and the average frequency of each word, accounting for different description lengths by dividing by length of the description. If a word appears in more 25% of medication or condition descriptions, it is presumed to be an unimportant word and filtered out. Similarly, if the ratio of the percent presence of a word in the Wikipedia articles to the condition descriptions is more than 1%, we presume it is not a medical word, since medical words should not be commonly found in a random set of Wikipedia articles. Finally, we filter out words that appear less in the given condition or medication than the average condition or medication. The resulting list of words are our keywords for the inputted medication-condition pair, and if this list is empty, we claim the medication is unrelated to the medication.

4. Results on Real Data

4.1. Evaluation Approach/Study Design

To evaluate how effective our method is at explaining the medications to a patient, we focused on the results of 4 metrics: robustness, accuracy, usability, and performance.

In order to measure robustness, we gathered two cohorts of patients from the MIMIC-III dataset, one with 12 patients, and one with 150. For each of these cohorts, we determined how many patients we were able to find discharge medications and discharge conditions. For the smaller 12 patient cohort, we also manually inspected the discharge summaries to ensure that no discharge information was missed by our algorithm.

Then, we tested how accurate our MedicineExplainer actually was at identifying which conditions are being treated by each medicine. To do this, we wrote 60 test cases of positively related medicine and condition pairs to make sure keywords were found, and 60 test cases of unrelated medicine-condition pairs to make sure keywords were NOT found. Second, we selected patients at random from the cohort of 12 patients to inspect in depth. For each medication, we examined its description as well as the descriptions of each of the conditions found by the NLP algorithm, and determined whether the condition was actually a plausible reason for needing that medication. We also looked through the remaining conditions to check if the algorithm missed any related condition, which was especially important for the patient medications in which no condition was found.

Our next criteria to test was the usability of our tool because as a user-facing tool, it should be intuitive to use. In particular, we evaluated whether we were able to meet our original design goals that focused on user flow. Although this method relies heavily on the opinions of our team members and is largely qualitative in nature, if we were to take our app to production we would plan to test it on a large sample of users and get quantitative measures, such as time taken to accomplish a certain task using our app.

Finally, we measured the performance of our system by testing how long it takes to fetch the medications and conditions for a patient from the Anvil server, and how long it takes to display the information for each medication.

4.2. Results on Detecting Medications and Conditions from Discharge Summaries

The first step in our method required us to collect the discharge medications and discharge conditions of each patient; these are the medications that a patient will have to take on their own at home, and the conditions they have upon exiting the hospital. We were not interested in the inpatient medications or treatments as the medical staff administers those. Because of this, we utilized the discharge summaries associated with each patient stay and used NLP and regex techniques to identify individual medications and conditions. As mentioned above, we tested the results of these search algorithms on two cohorts of patients.

The first cohort had 12 patients. Our algorithm was able to find medications for 8 of these patients and conditions for 9 of these patients. Upon visually examining the discharge summaries, there weren't any discharge medications listed for the remaining 4 patients, nor any discharge conditions for the remaining 3. This meant that our algorithm robustly and

accurately identified the medications and/or conditions from the discharge summaries IF they were indeed present.

We then tested our algorithm for robustness on a larger cohort of 150 and 125 patients, although we did not visually verify the results for accuracy as we did for the smaller cohort of 12. For these patients, we were able to find discharge medications for 109 out of 150 of the patients, and discharge conditions for 105 out of 125 patients. This means that our tool would be useful for at least 105 patients out of 150 (or 70%), because we would be able to tell them what their medications are for, and the side effects they may experience, regardless of whether we can parse their discharge conditions or identify the relevant conditions.

4.3. Results on Finding Keywords from Medication and Condition Descriptions

Figuring out what a medication is treating is a difficult task without domain knowledge in medicine. However, by turning it into an NLP task, we were able to find whether there was a relationships between a given medication and a given condition by finding keywords that were mutually in the descriptions for both the medication and condition, and were also unique (infrequently present in other descriptions).

We created 60 positive and 60 negative test cases in order to evaluate the accuracy of our keyword identifier. Each positive case consists of a condition and a medication that treats that condition, and each negative case consists of a condition and a medication that does not treat and/or is unrelated to that condition. For example, "diabetes type 2" and "insulin" would be a positive case, but "depression" and "insulin" would be a negative case. Of these cases, our algorithm correctly found a correlation for 58 out of the 60 positive cases, and incorrectly found a correlation for 16 out of 60 cases (false positives). Of these 16 cases, however, our algorithm detected words such as "kidney" or "blood pressure" or "diabetes" that may have some relationship to the medication and condition, even if it's not a direct treatment. After examining these false positives and manually removing such words, we would get just 5 false positives. A confusion matrix is shown in Table 1 with the results of the algorithm on the 120 test cases (without the manual adjustment).

		Predicted relationship		
		Positive	Negative	Total
Actual relationship	Positive	58	2	60
	Negative	16	44	60
	Total	74	46	120

Table 1: Confusion matrix of keyword relationship identification

Additionally, we randomly selected 5 patients from our 12 patient cohort to manually examine. For each of these patients, we checked whether there actually was a relationship between the medicine and the conditions it identified for that medicine. Our algorithm was correctly able to identify the medications and conditions we included in our test set, as well as numerous others. It performed particularly well on identifying medications associated with depression. However, some conditions listed were not necessarily treatable conditions, such as "s/p Fall" and "postop confusion". These conditions might have had a related

medicine but were not able to find descriptions on Medlineplus, and therefore did not have associated medications. Furthermore, there were some matches that may have been unrelated in actuality, but we did not have the domain expertise to say with certainty whether or not that match was correct, or should not have been matched. Similarly, there may be medications that have additional use cases not mentioned on Medlineplus, and those conditions would also be missed. In order to improve the accuracy of our methods, we would need to extensively with clinicians who have the domain knowledge to help us make our system both more robust and accurate.

4.4. Results on Front-end User Experience

The front end user experience was a key component of our MedicineExplainer, and we evaluated it based on its usability and performance.

Figures 1 and 2 display the user flow once our application is run. The panels for medications, conditions, and medicine description area all empty. There is a text box at the top of the page to enter in a patient's hospital admission ID into. When integrated into a hospital's patient portal, the admission ID would be automatically known to the system, but our tool requires a manual entry. Currently, only the admission IDs that are in our downloaded patient cohorts are searchable due to data limits in Google Colab, but any patient could be searched once added to our database.

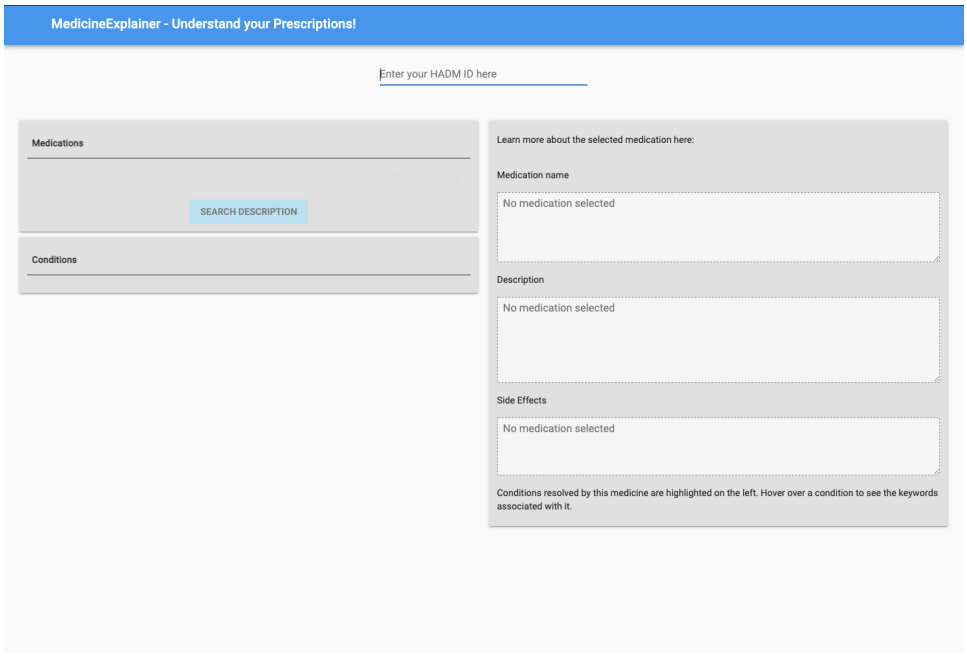
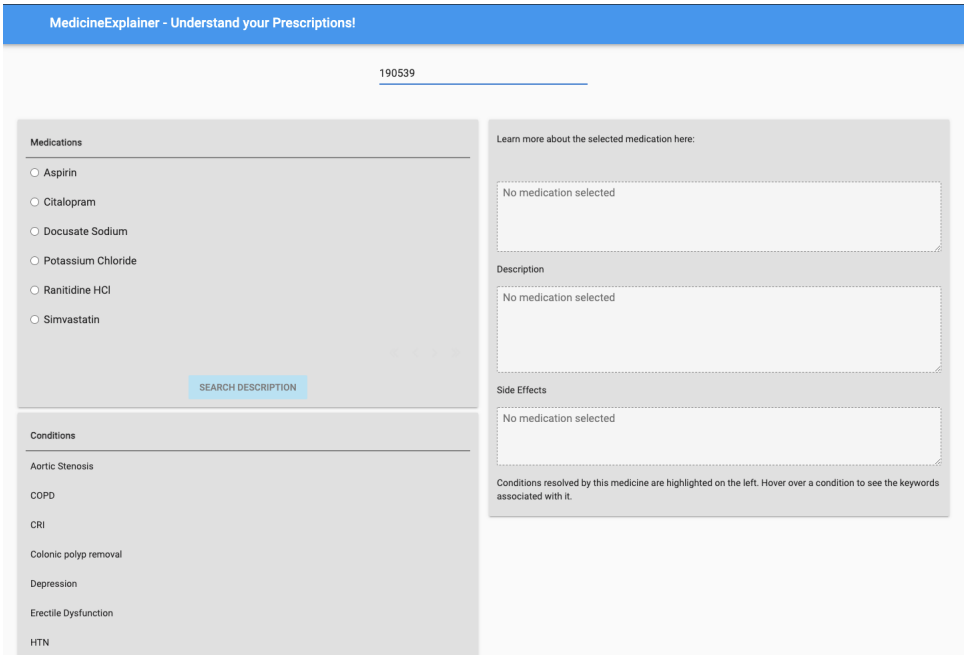
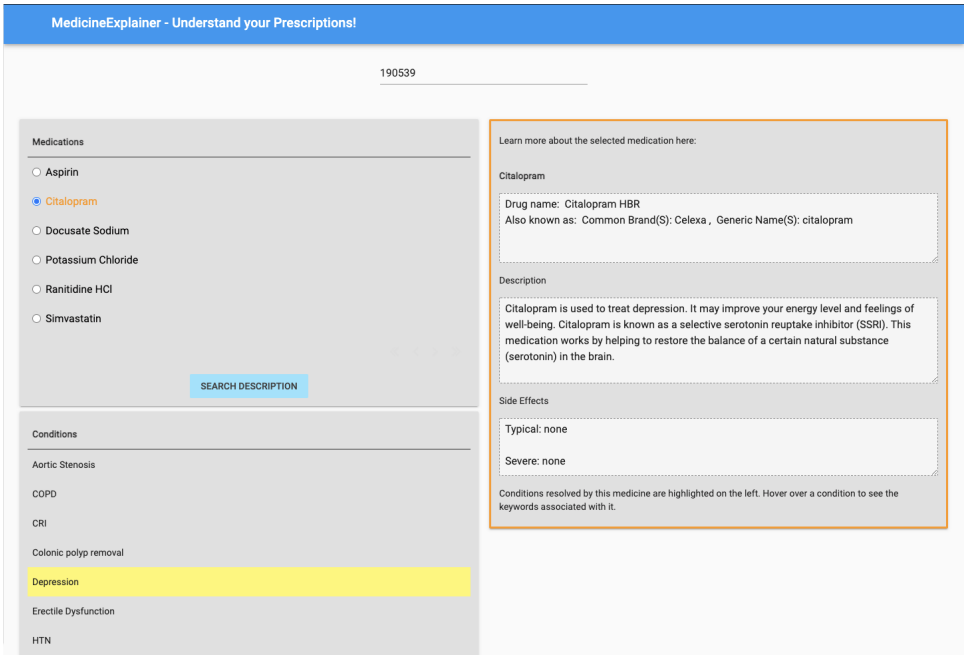


Figure 1: User opens application, must enter hospital admission ID.



((a)) After entering admission ID, patient's medications and conditions load. User can select a medication to learn more about, then press "SEARCH DESCRIPTION".



((b)) Information about the selected medication will be shown in the panel on the right, and the relevant conditions will be highlighted in the Conditions panel.

Figure 2: Overview of Front-end User Flow Screens 2 and 3

Once the user enters in the hospital admission ID, the discharge medications and conditions are rendered in their respective panel. The load time for this is approximately 40 seconds, depending on the number of medications and conditions a patient has. In order to query the use for each medication, the user can select the radio button next to a medication and press the "Search Description" button at the bottom of the Medications panel. Once the information for that medication has been pulled up, the selected medication will change to have orange text, and the Medication description panel on the right will have an orange border to demarcate that it is the description for the selected medication. At this point, the panel will be populated with useful information such as alternative drug or brand names for the medication, a description of what the medication is used for, and possible normal and severe side effects of the medication. Most importantly, when a condition from the Conditions panel is found to be related to the selected medication, it will be highlighted in yellow. The process of selecting and searching for a medication takes approximately 20 seconds, depending on the number conditions a patient has, as each condition could potentially be affected by the medication.

We find that the simplistic flow of the front-end makes it very user friendly, and that the orange highlighting of the medication and medication description panel border make it intuitive to understand what is happening It is significantly easier to read than an entire discharge summary, and the ability to simply click between medications for important and personalized information, rather than individually googling and reading numerous articles, will be faster and easier for patients, and especially those who are not skilled in using technology.

This flow was designed and tested by 2 of our group members, but to gain a better understanding of its usability we would have to also test it on actual patients, their caretakers, and clinicians. Furthermore, the performance of the system could be improved, as waiting 20 seconds for each medication can be frustrating for users accustomed to faster applications. We can do this by pre-loading the medication and condition descriptions in a hospital wide database,or caching condition-medication relationships as they are queried to make future queries faster.

5. Discussion

In this paper, we demonstrate a tool which can be utilised by patients to understand more about their medications and the conditions they treat without spending critical time and energy into web searching, with relevant results delivered directly to them on a click. We originally questioned if such information can accurately, or near accurately, be supplied to patients, and indeed, the sequence of algorithms we implemented yield a tolerable performance on the rather extensive cases tested. First, the algorithms reliably extracted the relevant medications and conditions for the patient in consideration, and next, they gave concise and real descriptions of the medications, and mostly the correct conditions treated by the medicine. In most cases, the keywords associated with each condition were as expected as well. It was interesting that the associations of medications with conditions and keywords was not as convoluted as usually conceived by a patient who has little knowledge about medications, and that the mapping, and hence understanding, was clean and

straightforward for most medications. Even when the medicine is such that it can treat a multitude of conditions, the conditions being chosen were not too off the correct ones.

Limitations There are a few limitations of the current implementation, one of them being the speed of extraction, which might hinder scalability when operated on larger datasets on the scale of a hospital. To counter that, we envision the use of caches, or a big database, where the results are either stored once queried for, or are pre-stored in bulk for common conditions and medications, respectively, using some form of dynamic programming. This is to avoid going through the web and invoking the ML model every time the same query is put in, and this way we can pre-load descriptions for medications and conditions. Secondly, we are using discharge summaries to obtain patient medications and conditions right now, which does not make extraction very consistent, but in future uses of the tool we can integrate it into real systems which normally have the lists of medications and conditions separately stored, as is the case in many common online patient portals nowadays. Additionally, as the major goal is to improve patient understanding, and hence treatment adherence, the use of keywords might prove particularly effective, although this needs to be tested more. Another simplicity of the model which can be exploited further is the use of unigrams in the NLP model; longer combinations might prove beneficial and improve the performance drastically because currently any word that is not a noun is ignored, so for example, the word "not" gets thrown away but if considered, it changes the meaning of the combination entirely. We might also be able to use recurrent neural networks to make the model more sophisticated and accurate, but we cannot ignore that the simplicity and transparency of the present system complements its power and usefulness.

As for taking the system to an actual clinical setting, there are several things we need to do. We need to consult and work with clinicians to get better accuracy on medication-condition matches, as well as account for conditions such as falling down which are not real diseases and do not have medications associated with them. Our front-end is near ready to be implemented as an actual web application, and the simplicity and the platform should make it easy to integrate with current hospital systems, although we do need to conduct alpha and beta tests on actual patients and their caregivers to test the user flow. But all in all, the underlying algorithms coupled with a working, intuitive and usable front-end make for a solid prototype of a simple concept, which is enabling the patients to understand and follow their treatments and conditions; it is a big leap from doctors' reports and official health records, which are almost incomprehensible for the common public.

Acknowledgments

Many thanks to our TA Wei-Hung Weng, our instructors, Dr. Cait, Dr. Salant, and all our mentors!

Contributions

Vibha attended project meetings with Dr. Cait and Dr. Salant, brainstormed project idea with Anil, created initial technical plan and wrote the corresponding proposal section, edited the proposal, wrote functions `get_mutual_words`, `get_corpus` and 110 testcases and tested `get_pt_medications` and `get_pt_conditions`, set up Anvil server for class, helped

integrate parts of the project pipeline and in documentation, designed user flow and helped test and debug Anvil, wrote results section in writeup, helped create user flow figure and confusion matrix.
Shinjini implemented, wrote testcases for, and tested the `get_pt_medications` and `get_pt_conditions` functions from discharge summaries, wrote the `test_patient_medications` and `test_patient_conditions` functions, helped debug the `get_med_info` function and in pushing patient conditions to Anvil app development servers, consolidation of the written code into the final project pipeline, final code documentation, running final testcases and obtained final results for writeup, utilised `test_patient_medications` and `test_patient_conditions` on `Data_Bigger.csv` to obtain improved results as well as test for robustness, wrote the abstract, introduction, related works, generalizable insights, acknowledgments, contributions, references and relevant code sections and some of the methods and results sections in the writeup, helped edit and reformat user flow figure for writeup, did final editing and formatting of writeup, edited the code README, and handled the code-GitHub repository setup and workflow.
Stuti learnt how Anvil app development platform works, connected the Google Colab code to Anvil and ported over functions, helped design and then redefined the user interface to make it more intuitive and easy to follow while also minimalistic, with instructions in tooltips, buttons that deactivate when unusable, appropriate highlighting and placeholders. She ensured the functions extracting patient data and getting the medication information were appropriate for handling by Anvil, implemented a user ID input system in the app, wrote the discussions section in the writeup and helped creating the user flow figure.
Anil wrote the query for discharge summary data, Helped to implement regex for `get_pt_medications` and `get_pt_conditions` functions, wrote functions to scrape descriptions, side effects, etc. from Medlineplus or WebMD, generated 3 corpora by web scraping, used said corpora for `isImportantWord` function to filter keywords, wrote the background section in the proposal and methods section in the writeup and README for the code.

References

[1] https://www.mayoclinic.org/diseases-conditions/high-blood-pressure/in-depth/high-blood-pressure-medication/art-20046280, 2004. [Online; accessed 13-May-2020].

[2] M. R. Bronsert, W. G. Henderson, R. Valuck, P. Hosokawa, and K. Hammermeister. Comparative effectiveness of antihypertensive therapeutic classes and treatment strategies in the initiation of therapy in primary care patients: A distributed ambulatory research in therapeutics network (DARTNet) study. *The Journal of the American Board of Family Medicine*, 26(5):529–538, September 2013. doi: 10.3122/jabfm.2013.05.130048. URL https://doi.org/10.3122/jabfm.2013.05.130048.

[3] Michel Burnier. Adherence to treatment in hypertension. In *Manual of Hypertension of the European Society of Hypertension*, pages 369–377. CRC Press, June 2019. doi: 10.1201/9780429199189-52. URL https://doi.org/10.1201/9780429199189-52.

[4] Michel Burnier and Brent M. Egan. Adherence in hypertension. *Circulation Research*, 124(7):1124–1140, March 2019. doi: 10.1161/circresaha.118.313220. URL https://doi.org/10.1161/circresaha.118.313220.

[5] Catherine M. DesRoches, Sigall K. Bell, Zhiyong Dong, Joann Elmore, Leonor Fernandez, Patricia Fitzgerald, Joshua M. Liao, Thomas H. Payne, Tom Delbanco, and Jan Walker. Patients managing medications and reading their visit notes: A survey of OpenNotes participants. *Annals of Internal Medicine*, 171(1):69, May 2019. doi: 10.7326/m18-3197. URL https://doi.org/10.7326/m18-3197.

[6] Ellen Kim, Samuel M. Rubinstein, Kevin T. Nead, Andrzej P. Wojcieszynski, Peter E. Gabriel, and Jeremy L. Warner. The evolving use of electronic health records (EHR) for research. *Seminars in Radiation Oncology*, 29(4):354–361, October 2019. doi: 10.1016/j.semradonc.2019.05.010. URL https://doi.org/10.1016/j.semradonc.2019.05.010.

[7] Lídia MVR Moura, M Brandon Westover, David Kwasnik, Andrew J Cole, and John Hsu. Causal inference as an emerging statistical approach in neurology: an example for epilepsy in the elderly. *Clinical Epidemiology*, Volume 9:9–18, December 2016. doi: 10.2147/clep.s121023. URL https://doi.org/10.2147/clep.s121023.

[8] Jalees Rahman. https://blogs.scientificamerican.com/guest-blog/accuracy-of-medical-information-on-the-internet/, 2012. [Online; accessed 13-May-2020].

[9] A Yazdani and E Boerwinkle. Causal inference in the age of decision medicine. *Journal of Data Mining in Genomics & Proteomics*, 06(01), 2015. doi: 10.4172/2153-0602.1000163. URL https://doi.org/10.4172/2153-0602.1000163.

Appendix A.

The codebase for the project can be found at https://github.com/shinjinighosh/6.871-Opennotes-Highlighter.