702 Project - Prescription Recommendations Using Clinical Notes

Peter Shen

2018-04-15

Introduction

Electronic health record systems are primarily designed for hospitals to control their workflows and billing systems. Valuable information relating to health outcomes including medical adherence, effective therapy, and patient response is buried within clinical notes. Being able to extract information out of clinical text can lead to valuable insights and create both social and economic value in promoting data-driven changes to workflows and health management. Clinical texts often include symptomps and diagnostics, which lead to reasons behind changes to medication. However, being able to provide clinical decision support around prescription and therapies would be helpful in guiding physicians towards treatment as well as conducting studies on pharmacovigilence. In this exeperiment, I will be using deep learning to recommend medications based on the input of clinical text.

Data

Data Publication

The data is published by Dr. Ozlem Uzuner at Partners Healthcare with the main focus of creating a Research Patient Data Repository for a series of NLP Challenges hosted by i2b2. The dataset `2009 Medication Challenge` is released as the third i2b2 workshop by the 'the i2b2 medication challenge team'.

Motivations

The team at i2b2 are passionate about leveraging the vast amount of information existing in clinical data in order deliver insights that can lead to better applications that can impact and improve healthcare delivery. i2b2 is built on the use cases developed from Driving Biology Projects, and throughout the years, the information that has been unlocked from both unstructured text as well as other clinical information has been valuable in improving healthcare systems. The NLP project at i2b2 is specifically aimed at improving methods and technologies analyzing unstructured clinical text.

The medication challenge focuses on extracting the medications prescribed to the patient from discharge summaries, which can collectively provide comprehensive record of medications.

The extracted information can be useful for pharmacovigilance, comparative effectiveness studies, and for detecting adverse events.

Data Timelines

The last released data was on the 2014 challenge titled `De-identification and Heart Disease Risk Factors Challenge`[1]. The dataset that I am proposing to use in this project is the `2009 Medication Challenge` in order to get insights and recommendations from prescriptions based on clinical notes.

Data Structure

The clinical texts were annotated by a medical community, which extracted the following fields from the notes [2]: 1. Medications (m): including names, brand names, generics, and collective names of prescription substances, over the counter medications, and other biological substances for which the patient is the experiencer.

2. Dosages (do): indicating the amount of a medication used in each administration.
3. Modes (mo): indicating the route for administering the medication.
4. Frequencies (f): indicating how often each dose of the medication should be taken.
5. Durations (du): indicating how long the medication is to be administered.
6. Reasons (r): stating the medical reason for which the medication is given.
7. List/narrative (ln): indicating whether the medication information appears in a list structure or in narrative running text in the discharge summary.

Data Access

The data can be accessed at i2b2 NLP Research. However, you does need to register for an i2b2 account and have the research proposal reviewed. Once reviewed, you will sign a Data Use Agreement, and the data can be downloaded at the i2b2 Download Page under the `2009 Medication Challenge`.

For the purpose of this experiment, only the following data files are needed: * Community Ground Truth for the Test Data – This contains the annotations from the community * Test Files (released August 17) – These contain all of the discharge summaries

Usage Restrictions

Partners' policy explicitly forbids reproducing any portion of these notes in oral presentations and/or publications.

Ontologies

No ontologies are used in the dataset itself. However, as part of the preprocessing of the data, the USP Drug Classification (USPDC) 2018 guidelines for medication categorization is used. This is because the dataset is very limited, so in order to get any predictive power, the medications are categorized based on their therapeutic class that is mapped by the USPDC guidelines. From the USPDC website states that these classifications are only used for drug formulary development or review. From their website:

> The USP Drug Classification (USP DC) is an independent drug classification system developed in response to stakeholder input that it would be helpful to have a classification system beyond the Medicare Model Guidelines (MMG), to assist with formulary support outside of Medicare Part D. The Healthcare Quality & Safety Expert Committee goal is to create a comprehensive classification system for use in drug formulary development or review in non-acute or outpatient care settings. This tool has the potential to provide guidance towards the design and comparison of balanced formularies.

This mapping does not include all of the drugs used in clinical settings, but it is good enough to be used in the experiment for categorization purposes.

Related Works

The original challenge was to extract the medications and related metadata from clinical texts. Of the 20 teams, majority of the teams are using a combination of rule-based systems and SVM. Most of the workflows involved tokenization and then feature extraction. The system developed by Jon Patrick and Min Li [3] achieved the highest F-measure (based on precision and recall). This system involved a 3-stage processing: tokenization, feature generation based on conditional random fields (CRF), SVM for local contexts, then feeding these features into a rule-based system using entity and positions. The second highest achieving team from Vanderbilt (Doan et al. 2010)[4] build a system by simply splitting the setences from discharge summaries and using a semantic tagger for each feature. Since all of the teams were using methods involved in classical machine learning (SVM, rule-based systems), exploring the same dataset using deep learning would be a novel approach.

Modern based approaches to analyzing clinical text does include using deep learning. For example, CliNer [5] used data from i2b2/VA 2010 challenge, employing feature extraction using both linguistic features (e.g., ngrams and wordshapes) and domain knowledge. Another group [6] was able to predict HIV risk assessment using NLP and clinical notes. A comprehensive study [7] shows that comparing different NLP approaches, CNN methods was able to achieve the highest F-score compared to bag of words representation of a note with logistic regression and using an n-gram representation of a note with logistic regression.

References

1. Uzuner, Ozlem, Imre Solti, Fei Xia, and Eithon Cadag. 2010. "Community Annotation Experiment for Ground Truth Generation for the i2b2 Medication Challenge." Journal of the American Medical Informatics Association: JAMIA 17 (5): 519–23.
2. Uzuner, Ozlem, Imre Solti, and Eithon Cadag. 2010. "Extracting Medication Information from Clinical Text." Journal of the American Medical Informatics Association: JAMIA 17 (5): 514–18.
3. Patrick, Jon, and Min Li. 2010. "High Accuracy Information Extraction of Medication Information from Clinical Notes: 2009 i2b2 Medication Extraction Challenge." Journal of the American Medical Informatics Association: JAMIA 17 (5): 524–27.
4. Doan, Son, Lisa Bastarache, Sergio Klimkowski, Joshua C. Denny, and Hua Xu. 2010. "Integrating Existing Natural Language Processing Tools for Medication Extraction from Discharge Summaries." Journal of the American Medical Informatics Association: JAMIA 17 (5): 528–31.
5. Boag, Willie, Elena Sergeeva, Saurabh Kulshreshtha, Peter Szolovits, Anna Rumshisky, and Tristan Naumann. 2018. "CliNER 2.0: Accessible and Accurate Clinical Concept Extraction." arXiv [cs.CL]. arXiv. http://arxiv.org/abs/1803.02245.
6. Feller, Daniel J., Jason Zucker, Michael T. Yin, Peter Gordon, and Noémie Elhadad. 2018. "Using Clinical Notes and Natural Language Processing for Automated HIV Risk Assessment." Journal of Acquired Immune Deficiency Syndromes 77 (2): 160–66.
7. Gehrmann, Sebastian, Franck Dernoncourt, Yeran Li, Eric T. Carlson, Joy T. Wu, Jonathan Welt, John Foote Jr., et al. 2018. "Comparing Deep Learning and Concept Extraction Based Methods for Patient Phenotyping from Clinical Narratives." PloS One 13 (2): e0192360.

Methods

Preprocessing

Import annotations file and retrieve all medications that are given with a specific clinical reason

```
# annotation file this is the files contained in Community
# Ground Truth for the Test Data dataset
annot <- list.files(path = "data/annotations_ground_truth/converted.noduplicates.sorted/

cols <- c("m", "do", "mo", "f", "du", "r", "ln")

df <- data.frame(matrix(ncol = 7, nrow = 0))
```

```
colnames(df) <- cols

# extract all of the fields from the ground truth dataset
for (file in annot) {
    id <- strsplit(file, split = ".", fixed = T)[[1]][1]
    file <- paste0("data/annotations_ground_truth/converted.noduplicates.sorted/",
        file)
    df.this <- read.table(file, sep = "|", stringsAsFactors = F,
        quote = "")
    df.this <- df.this[, seq(1, 13, 2)]
    colnames(df.this) <- cols
    df.this$id <- id
    df.this <- df.this %>% filter(r != "r=\"nm\"")

    df <- df %>% bind_rows(df.this)
}

# select the medications and reasons fields
df <- df %>% select(id, m, r)
```

From the reasons, build the metadata for the lines in the discharge summaries that would contain the reason for the medication.

```
# function to parse the line numbers
parseLineNumber <- function(line) {
    return(strsplit(line, ":")[[1]][1])
}

# function to retrieve the line
getLines <- function(row) {
    lines <- unlist(strsplit(trimws(strsplit(row, "\"")[[1]][3],
        "both"), " "))
    lineNumbers <- sapply(lines, parseLineNumber)
    uniqueLines <- unique(lineNumbers)

    return(paste(uniqueLines, collapse = " "))
}

# function to tokenize the lines
getTokens <- function(row) {
    tokens <- unlist(strsplit(trimws(strsplit(row, "\"")[[1]][3],
        "both"), " "))
    tokens <- unlist(strsplit(tokens, " "))
    tokens <- unique(tokens)
    return(paste(tokens, collapse = " "))
```

```
}

# get the row numbers for the lines
getContent <- function(row) {
    return(strsplit(row, "\"")[[1]][2])
}

# get the medication metadata
df$mcontent <- sapply(df$m, getContent)
df$mtokens <- sapply(df$m, getTokens)
df$mlines <- sapply(df$m, getLines)

# get the reasons metadata
df$rcontent <- sapply(df$r, getContent)
df$rtokens <- sapply(df$r, getTokens)
df$rlines <- sapply(df$r, getLines)

# remove periods from strings
df$mcontent <- gsub("[.]", "", df$mcontent)
df$rcontent <- gsub("[.]", "", df$rcontent)

# filter out medications that do not have a reason
df <- df %>% filter(rcontent != "nm")

df <- df %>% select(id, mcontent, mtokens, mlines, rcontent,
    rtokens, rlines) %>% distinct()
```

Which drugs are present in the dataset?

```
mcontent.classes <- group_by(df, mcontent) %>% summarise(count = n()) %>%
    arrange(desc(count))
print(paste("Number of medication classes: ", nrow(mcontent.classes)))
```

```
## [1] "Number of medication classes:  512"
```

```
print("Output the top 10 classes.")
```

```
## [1] "Output the top 10 classes."
```

```
mcontent.classes[1:10, ]
```

```
## # A tibble: 10 x 2
##    mcontent      count
##    <chr>         <int>
##  1 coumadin         70
##  2 lasix            43
##  3 insulin          36
```

```
##  4 nitroglycerin    36
##  5 levofloxacin     34
##  6 oxycodone        33
##  7 heparin          31
##  8 tylenol          29
##  9 aspirin          27
## 10 antibiotics      26
```

Retrieve the sentences around the reasons from discharge summaries

```
# parameter to extract the number of lines above and below
# the reasons we can experiment with this parameter to test
# how well the model fits and can generalize
num.lines <- 3

# retrieve the sentences around the reason from the file
getReasonsFromFile <- function(row) {
    file <- row["id"]
    content <- read_file(paste0("data/test_files/", file))
    rline <- row["rlines"]
    rline <- as.numeric(unlist(strsplit(rline, " ")))
    content <- unlist(strsplit(content, "\n"))
    numLines <- length(content)

    content <- content[(max(min(rline), 0) - num.lines):(min(max(rline),
        numLines) + num.lines)]

    content <- tolower(content)
    content <- gsub("[.|,|!|:]", "", content)

    return(paste(content, collapse = " "))
}

df$rsentences <- apply(df, 1, getReasonsFromFile)
```

Map medications to USPDC drug classes. We are using the USP Category which is a category that contains a broad range of medications

```
# the mapping file
uspdc <- read.csv("uspdc-2018-alignment-file.csv", header = T,
    stringsAsFactors = F)
uspdc$Name <- tolower(uspdc$Name)

# match the drug class to the medication in the notes
getUSPClass <- function(drug) {
    matchedDrugs <- grep(drug, uspdc$Name, fixed = T)
```

```
    class <- uspdc[matchedDrugs[1], ]$USP.Category
    return(class)
}

df$USPClass <- sapply(df$mcontent, getUSPClass)
```

After mapping, how many drugs are unmappable?

```
df[is.na(df$USPClass), ] %>% group_by(mcontent) %>% summarise(count = n()) %>%
    arrange(desc(count)) %>% filter(count > 5)
```

```
## # A tibble: 23 x 2
##    mcontent                                count
##    <chr>                                   <int>
##  1 antibiotics                                26
##  2 percocet                                   23
##  3 nitroglycerin 1/150 ( 04 mg )              15
##  4 tylenol ( acetaminophen )                  13
##  5 dobutamine                                 11
##  6 nph                                        11
##  7 albuterol inhaler                          10
##  8 maalox-tablets quick dissolve/chewable     10
##  9 ntg                                        10
## 10 albuterol nebulizer                         9
## # ... with 13 more rows
```

Manually map the unmappable drugs to the USPDC categories.

```
df$USPClass[df$mcontent == "antibiotics"] <- "Antibacterials"
df$USPClass[df$mcontent == "percocet"] <- "Analgesics"
df$USPClass[df$mcontent == "nitroglycerin 1/150 ( 04 mg )"] <- "Cardiovascular Agents"
df$USPClass[df$mcontent == "tylenol ( acetaminophen )"] <- "Analgesics"
df$USPClass[df$mcontent == "dobutamine"] <- "Cardiovascular Agents"
df$USPClass[df$mcontent == "dopamine"] <- "Cardiovascular Agents"
df$USPClass[df$mcontent == "nph"] <- "Blood Glucose Regulators"
df$USPClass[df$mcontent == "albuterol inhaler"] <- "Respiratory Tract/Pulmonary Agents"
df$USPClass[df$mcontent == "albuterol nebulizer"] <- "Respiratory Tract/Pulmonary Agents
df$USPClass[df$mcontent == "ntg"] <- "Cardiovascular Agents"
df$USPClass[df$mcontent == "bb"] <- "Cardiovascular Agents"
df$USPClass[df$mcontent == "human insulin"] <- "Blood Glucose Regulators"
df$USPClass[df$mcontent == "argatroban"] <- "Blood Products/Modifiers/Volume Expanders"
df$USPClass[df$mcontent == "ativan ( lorazepam )"] <- "Anticonvulsants"
df$USPClass[df$mcontent == "dulcolax"] <- "Anti-Constipation Agents"
df$USPClass[df$mcontent == "insulin aspart"] <- "Blood Glucose Regulators"
df$USPClass[df$mcontent == "insulin regular human"] <- "Blood Glucose Regulators"
df$USPClass[df$mcontent == "kcl"] <- "Gastrointestinal Agents"
```

```
df$USPClass[df$mcontent == "nitroglycerins"] <- "Cardiovascular Agents"
df$USPClass[df$mcontent == "novolog ( insulin aspart )"] <- "Blood Glucose Regulators"
df$USPClass[df$mcontent == "maalox-tablets quick dissolve/chewable"] <- "Gastrointestina
df$USPClass[df$mcontent == "maalox"] <- "Gastrointestinal Agents"
df$USPClass[df$mcontent == "colace"] <- "Gastrointestinal Agents"
```

Construct the dataframe that is used to output the cleaned data

```
df.output <- df %>% select(USPClass, rsentences) %>% filter(!is.na(USPClass)) %>%
    distinct()
```

To deal with the issue of sparse matrixes and data, we only select drug classes with more than 100 data points so that we can actually train the model

```
multi.classes <- df.output %>% filter(!is.na(rsentences)) %>%
    group_by(USPClass) %>% summarise(count = n()) %>% arrange(desc(count)) %>%
    filter(count > 100)
df.output <- df.output[df.output$USPClass %in% multi.classes$USPClass,
    ]
```

Write out the training data and labels * `data_x.txt` is the training sentences * `data_y.txt` is the labels * `output.txt` is the output categories for classification mapping the class to numbers in `data_y.txt`

```
df.output$USPClass <- factor(df.output$USPClass)
write.table(df.output$rsentences, file = "data_x.txt", row.names = F,
    col.names = F, sep = "\t")
write.table(as.numeric(df.output$USPClass) - 1, file = "data_y.txt",
    row.names = F, col.names = F, sep = "\t")

lev <- levels(df.output$USPClass)
out <- c()
for (i in 1:length(lev)) {
    out <- append(out, paste0("\"", lev[i], "\":", i - 1))
}

write(paste(out, collapse = ",\n"), "output.txt")
```

Deep Learning with Keras

The section for deep learning was done using Keras framework in Python on O2 cluster.

Parameters:

Parameter	Value
Embedding	GloVe 100

Parameter	Value
Sequence Len	1000
Vocab Size	20000
Validation %	0.2
Epochs	150
Batch Size	64
Optimizer	Adagrad

There are four classes of drugs that are being recommended: * Analgesics * Antibacterials * Blood Products/Modifiers/Volume Expanders * Cardiovascular Agents

The neural network architecture is diagrammed below in figure 1.

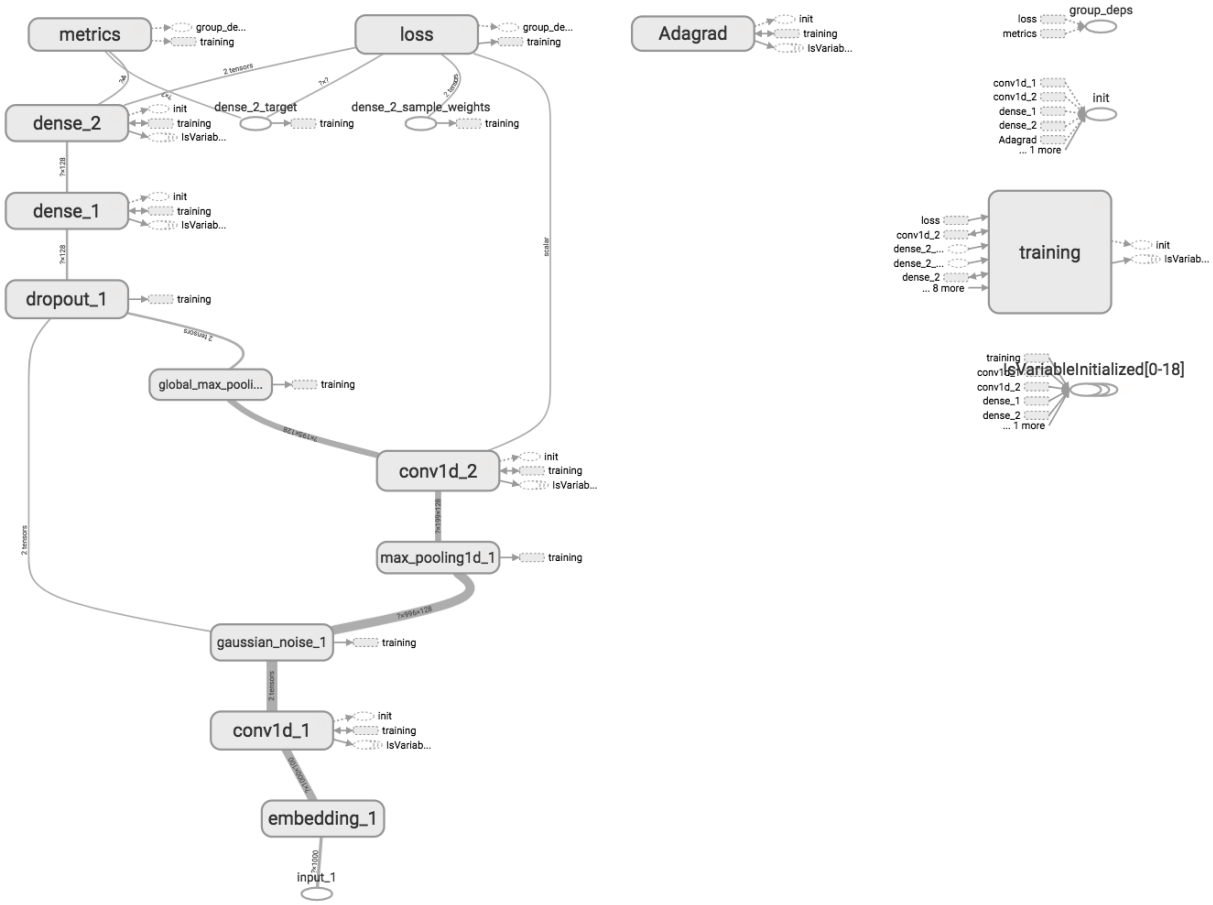

Figure 1: Network Diagram

In table form, the network is:

Layer	Description
Embedding	Input GloVe 100

Layer	Description
Conv 1D	128 ReLU hidden units
Noise	15% Gaussian
Max Pooling	5 units
Conv 1D	128 ReLU hidden units, L1 regularization 0.05
Glb Max Pool	Pool over all units
Dropout	40%
Dense	128 ReLU hidden units
Dense	Softmax Output layer

Results

With a very limited dataset of roughly 600 data records, the deep learning model ran into big issues with overfitting. However, with added dropout and gaussian nosise with L1 regularization, I was able to control the overtraining a little bit. The model accuracy and loss graphs are shown below:

Model Validation

Model accuracy between training set and validation set in figure 2.

Model loss between training and validation set is included in figure 3.

Deployment

The model is deployed using Flask (Python) to serve the model over a REST API-endpoint with a front-end written in Javascript and HTML to call the API.

A screenshot of the application is included in figure 4.

Code

Code is available at: https://github.com/p-shen/702-medication-project

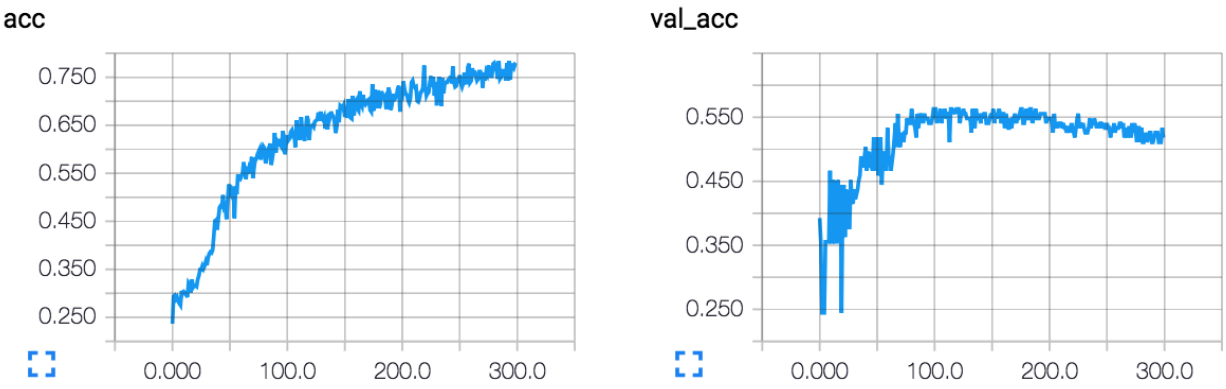

Figure 2: Model Accuracy

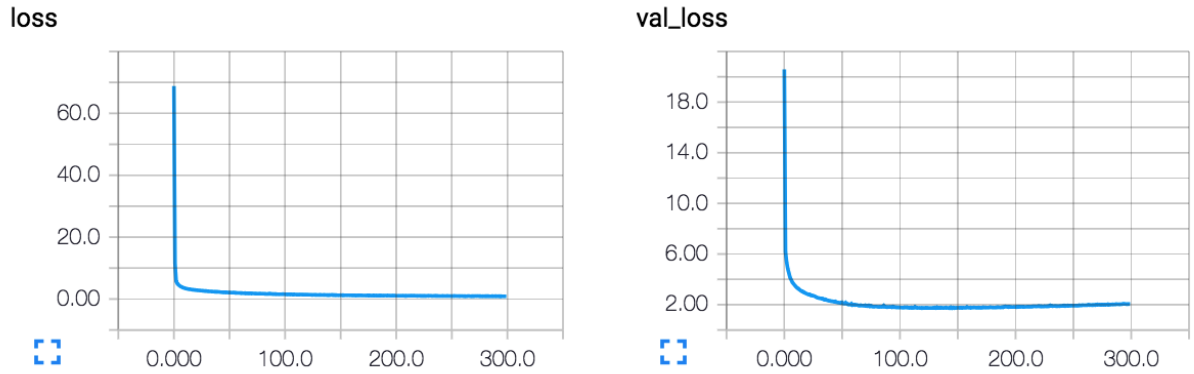

Figure 3: Model Loss

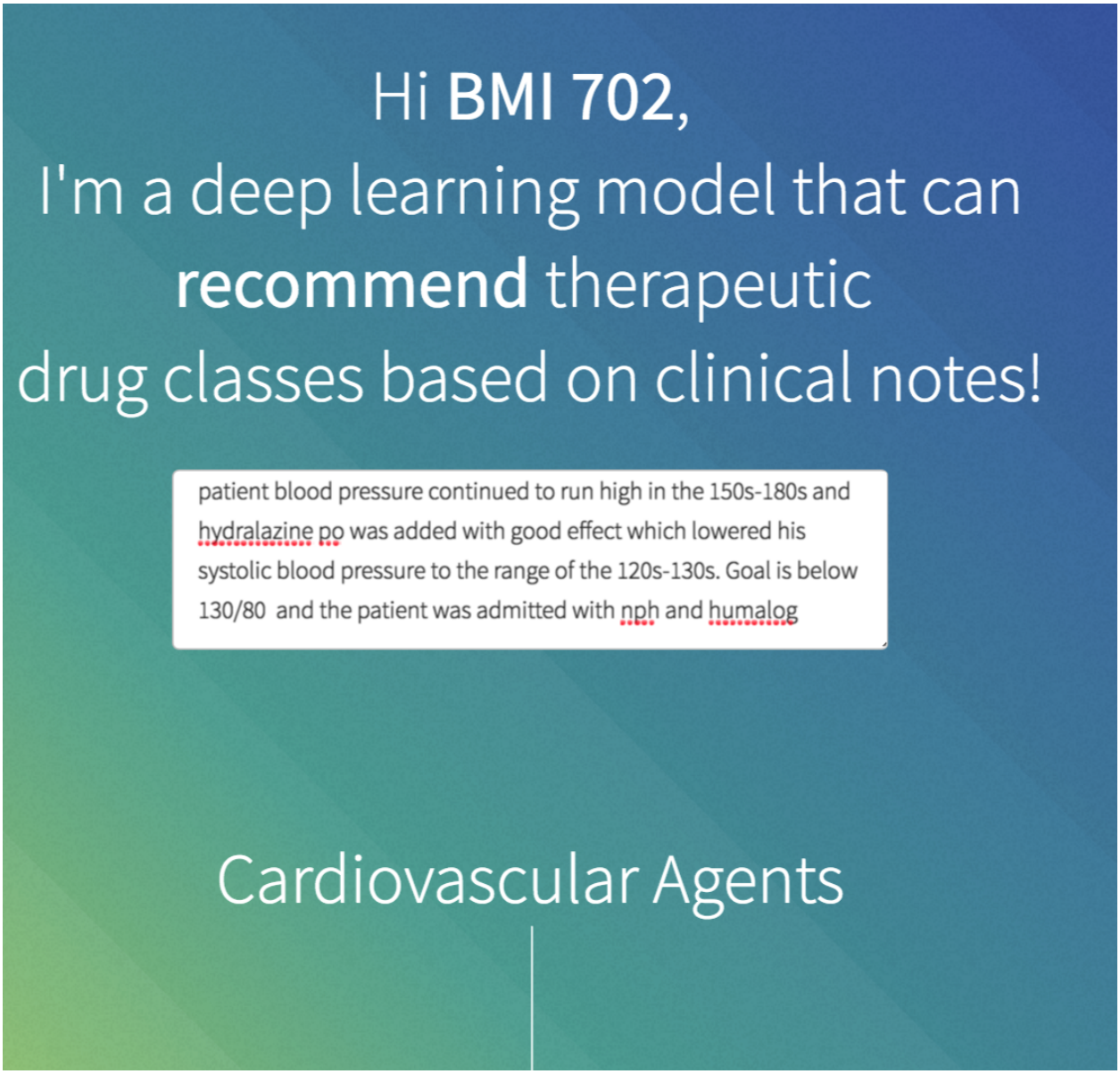

Figure 4: Application Screenshot