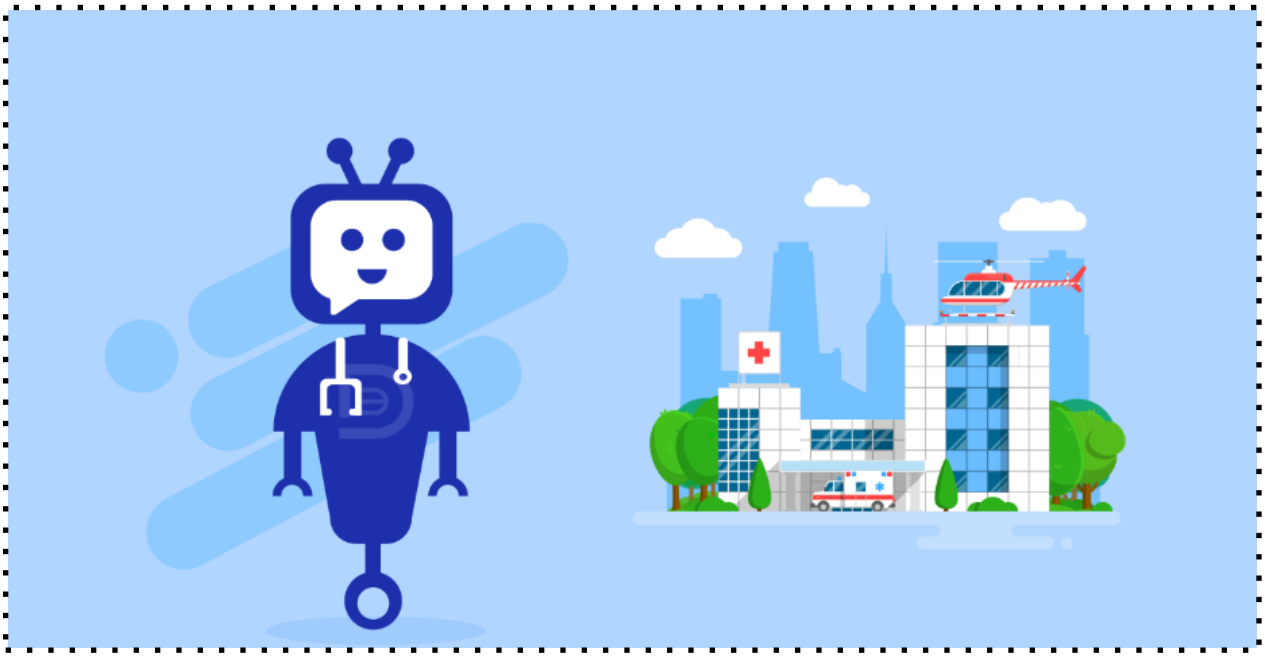
Final Paper on
Medibot
(Doctor in Pocket!)



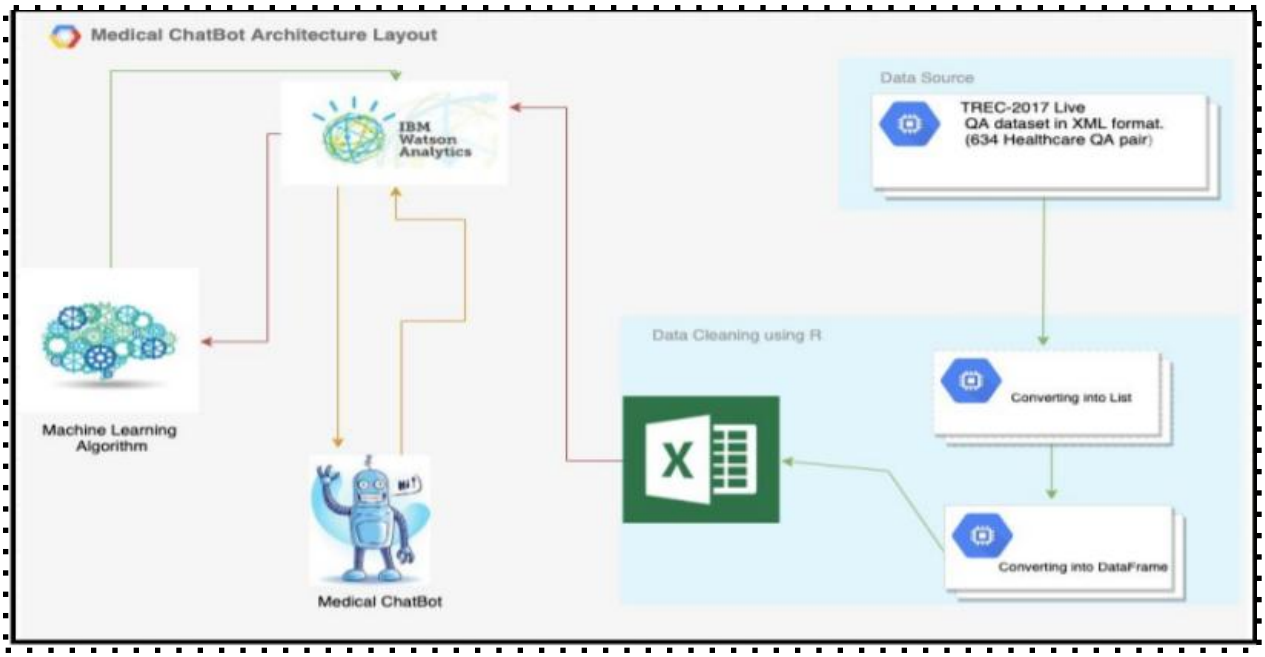
Submitted By
Alokita Garg

Introduction

A chatbot is a computer program or an artificial intelligence system which conducts a conversation via auditory or textual methods. Chatbots are becoming an integral part of customer experience and are allowing business to deliver services in a highly personalized manner where the message, operations, and human support can be combined in one experience. In this project, our focus is medical chatbots. Medical chatbots have the potential to reduce healthcare costs and improve access to medical information. Our product is a text-to-text diagnosis of medical bot that allows patients to talk about their medical problems. Our medical chatbot, Medibot will help patients understand their medical problems and will also provide an appropriate medical solution. The idea behind building this product is to reduce the dependency of doctors in minor medical problems and thereby minimizing the high medical costs. In United States of America, healthcare is a highly expensive field and most of the Americans have limited medical coverage. As Medibot is easily available and operates with minimal costs, it is an opportunity for effective and convenient automated medical help.

Architecture

Medibot will be trained by a set of 634 questions and answers data set. This data will be converted to lists and data frames using R statistical language. The transformed data will be stored in excel sheets. Excel sheets will be categorized as per IBM Watson analytics platform and will be deployed on platform as a service. We will leverage IBM's machine learning platform for to and fro communication with Medibot. All the input to chatbot will be processed by Natural Language Processing (NLP) technique. Through NLP input data will be into multiple categories of health problems like knee problems, heart problems, brain problems, etc.



Data Source and Collection

We have collected data in questions and answers pair required for Medibot from GitHub which are in XML format. The data is in question-answer pairs for training and testing, with additional annotations that can be used to develop question analysis and question answering systems.

Medical Training Data:

We have collected two XML files for training dataset which contains 634 question-answer pairs. The details regarding the XML files are mentioned below:

1) TREC-2017-LiveQA-Medical-Train-1.xml :
There are 388 question-pairs corresponding to 200 National Library of Medicine(NLM) questions. Each question is divided into one or more subquestions. Each subquestion has one or more answers.

2) REC-2017-LiveQA-Medical-Train-2.xml :
There are 246 question-pairs corresponding to 246 National Library of Medicine(NLM) Questions.

Simplified format of the Medical Training Data:

```
<NLM-QUESTION questionid="Q1" fRef="11373">
        <SUBJECT></SUBJECT>
        <MESSAGE>Literature on Cardiac amyloidosis.  Please let me know where I can
get literature on Cardiac amyloidosis.  My uncle died yesterday from this disorder.
Since this is such a rare disorder, and to honor his memory, I would like to distribute
literature at his funeral service.  I am a retired NIH employee, so I am familiar with the
campus in case you have literature at NIH that I can come and pick up.  Thank you
</MESSAGE>
        <SUB-QUESTIONS>
                <SUB-QUESTION subqid="Q1-S1">
                        <ANNOTATIONS>
                                <FOCUS>cardiac amyloidosis</FOCUS>
                                <TYPE>information</TYPE>
                        </ANNOTATIONS>
                        <ANSWERS>
                                <ANSWER answerid="Q1-S1-A1" pairid="1">Cardiac
amyloidosis is a disorder caused by deposits of an abnormal protein (amyloid) in the
heart tissue. These deposits make it hard for the heart to work properly.</ANSWER>
                                <ANSWER answerid="Q1-S1-A2" pairid="2">The
term "amyloidosis" refers not to a single disease but to a collection of diseases in which a
protein-based infiltrate deposits in tissues as beta-pleated sheets. The subtype of the
disease is determined by which protein is depositing; although dozens of subtypes have
been described, most are incredibly rare or of trivial importance. This analysis will focus
on the main systemic forms of amyloidosis, both of which frequently involve the
heart.</ANSWER>
                        </ANSWERS>
                </SUB-QUESTION>
        </SUB-QUESTIONS>
</NLM-QUESTION>
```

Medical Test Data:

There are 26 question-pairs in the medical test data which are associated with five categories. Each question contains one or more sub-questions and at least one focus and one question type.

Simplified format of the Medical Test Data:

<NLM-QUESTION qid="TQ1">
<Original-Question qfile="71.txt">
<SUBJECT>Noonan syndrome</SUBJECT>
<MESSAGE>What are the references with noonan syndrome and polycystic renal disease</MESSAGE>
</Original-Question>
<NIST-PARAPHRASE>What is the relationship between Noonan syndrome and polycystic renal disease?</NIST-PARAPHRASE>
<ANNOTATIONS>
<FOCUS fid="F1" fcategory="Problem">noonan syndrome</FOCUS>
<FOCUS fid="F2" fcategory="Problem">polycystic renal disease</FOCUS>
<TYPE tid="T1" hasFocus="F1,F2">EFFECT</TYPE>
</ANNOTATIONS>
<ReferenceAnswers>
<RefAnswer aid="TQ1A1">
<ANSWER> Noonan's syndrome is an eponymic designation that has been used during the last 8 years to describe a variable constellation of somatic and visceral congenital anomalies, which includes groups of patients previously referred to as male Turner's, female pseudo-Turner's and Bonnevie-Ullrich syndromes. It is now recognized that both sexes may show the stigmas of this condition and, unlike Turner's syndrome, there is no karyotype abnormality although there is often a familial pattern. The most commonly observed anomalies include webbing of the neck, hypertelorism, a shield-shaped chest and short stature. Congenital heart disease, principally pulmonary stenosis, and sexual infantilism often with cryptorchidism in the male subject are additional associated anomalies in this syndrome. Renal anomalies have been described rarely and usually consist of rotational errors, duplications and hydronephrosis. We report the first case of an infant who displayed many of the stigmas of Noonan's

syndrome and also showed early evidence of frank renal failure secondary to renal dysplasia with cystic disease. </ANSWER>
<AnswerURL>
https://www.ncbi.nlm.nih.gov/pubmed/765504 </AnswerURL>
<COMMENT>This article is from 1976, and is a case report.</COMMENT>
</RefAnswer>
<RefAnswer aid="TQ1A2">
<ANSWER>10% of patients with Noonan syndrome have renal abnormalities but most do not need treatment.</ANSWER>

<AnswerURL>http://www.thelancet.com/journals/lancet/article/PIIS0140-6736(12)61023-X/fulltext</AnswerURL>
<COMMENT>Often, the questions are about the associations that do not exist, as is in this case: there is one patient with both disorders. Answers about the lack of association or one of the conditions and more general disorders (as renal disease in this case) are relevant. </COMMENT>
</RefAnswer>
<RefAnswer aid="TQ1A3">
<ANSWER>Genitourinary. Renal abnormalities, generally mild, are present in 11% of individuals with NS. Dilatation of the renal pelvis is most common. Duplex collecting systems, minor rotational anomalies, distal ureteric stenosis, renal hypoplasia, unilateral renal agenesis, unilateral renal ectopia, and bilateral cysts with scarring are reported less commonly.</ANSWER>

<AnswerURL>https://www.ncbi.nlm.nih.gov/books/NBK1124/</AnswerURL>
<COMMENT>This answer does not specifically address polycystic renal disease but implicitly establishes the lack of association. </COMMENT>
</RefAnswer>
</ReferenceAnswers>
</NLM-QUESTION>

Data Cleaning

Since the data is in XML format which is quite unorganized for analysis will have to

parse the data through each and convert it into a readable data frame. For this process we will be using R. Libraries being used are ‘tidyverse’ and ‘xml2’. The data is basically a Q&A where the columns that we will be using are as follows:

Column Name	Description
Focus	The main topic of the Q&A example: Cervical cancer, Genetic mutation
Type	Category of the question example: cause, treatment
Message	A question related to medical problems
Subject	The subject matter of the question
Answer	The answer to the question

Later on, we will have to make sure data has no nulls or N/A.

Data Extraction

The data for the chatbot was extracted through the Github API. This API was accessed using the Git Token. The Git Token along with the secret access key was generated through the Github website by using Github account credentials.
The QA healthcare dataset is in XML format. This data is pulled from ‘abachha’ account using get request to Github API. The data from API was received in the JSON format.
Using the following code in R, the raw XML textual data is extracted using the Github API and stored in a list data structure.

```r
install.packages("jsonlite")
library(jsonlite)
install.packages("httpuv")
library(httpuv)
install.packages("httr")
library(httr)


# Can be github, linkedin etc depending on application

oauth_endpoints("github")

# Entering Github API Authentication credentials

myapp <- oauth_app(appname = "MediBot",
                   key = "0e360abadd830c271423",
                   secret = "5cfa19a9fce6d93b10ea1ad3faf182b26f8c09d3")


# Get OAuth credentials
github_token <- oauth2.0_token(oauth_endpoints("github"), myapp)



# Use API
gtoken <- config(token = github_token)
req <- GET("https://raw.githubusercontent.com/abachaa/LiveQA_MedicalTask_TREC2017/master/TrainingDatasets/TREC-2017


# Take action on http error
stop_for_status(req)

# Extract content from a request
json1 = content(req)

print(json1)

# Convert to a data.frame
gitDF = jsonlite::fromJSON(jsonlite::toJSON(json1))


```

Data Transformation:

The .XML data that was extracted from the above step we have to transform it to an easily analyzable format. We chose to convert it to a .CSV file so that we can import this easily into IBM Watson cloud. We used R code to do the transformation. Following are the steps followed:

1. Read data from XML to List

```
xmldata1 <- xmlParse("CIS_8395_Project/TrainingDatasets/Train1.xml")
xmlList1 <- xmlToList(xmldata1)

xmldata2 <- xmlParse("CIS_8395_Project/TrainingDatasets/Train2.xml")
xmlList2 <- xmlToList(xmldata2)
```

2. How the list looks like

```
xmlList <- xmlList1

xmlList$`NLM-QUESTION`$`SUB-QUESTIONS`$`SUB-QUESTION`$ANSWERS
```

```
## $ANSWER
## $ANSWER$text
## [1] "Cardiac amyloidosis is a disorder caused by deposits of an abnormal protein (amyloid) in the
heart tissue. These deposits make it hard for the heart to work properly."
##
## $ANSWER$.attrs
##   answerid     pairid
## "Q1-S1-A1"        "1"
##
##
## $ANSWER
## $ANSWER$text
## [1] "The term \"amyloidosis\" refers not to a single disease but to a collection of diseases in wh
ich a protein-based infiltrate deposits in tissues as beta-pleated sheets. The subtype of the disease
 is determined by which protein is depositing; although dozens of subtypes have been described, most
are incredibly rare or of trivial importance. This analysis will focus on the main systemic forms of
amyloidosis, both of which frequently involve the heart."
##
## $ANSWER$.attrs
##   answerid     pairid
## "Q1-S1-A2"        "2"
```

Here we can have a look at how to traverse through the nodes of the XML to use variables that we might need.

3. Choosing variables to use

```
xmldata <- read_xml("CIS_8395_Project/TrainingDatasets/Train1.xml")
subject <- xml_find_all(xmldata, "//SUBJECT")

subject <- trimws(xml_text(subject))
head(subject)
```

```
## [1] ""
## [2] "treatment options versus migraine types"
## [3] ""
## [4] "cramp"
## [5] "Retina"
## [6] "hypothyroidism"
```

```
focus <- xml_find_all(xmldata, "//FOCUS")
focus <- trimws(xml_text(focus))
head(focus)
```

```
## [1] "cardiac amyloidosis"  "migraine"             "pyridoxine"
## [4] "cramp"                "macular degeneration" "hypothyroidism"
```

As we can see in the above screenshot the 'SUBJECT' variable has null values. Hence we decided to use the 'FOCUS' variable to categorize the questions and answers.

4. Creating vectors for the data columns

```
focus <- vector()
question <- vector()
answer <- vector()
```

5. Extract data into vectors from the list of the first XML file into vectors

```
for (i in 1:length(xmlList)) {
  subquestionSize <- length(xmlList[i]$`NLM-QUESTION`$`SUB-QUESTIONS`)
  question1 <- xmlList[i]$`NLM-QUESTION`$MESSAGE
  for (j in 1:subquestionSize) {
    answersSize <- length(xmlList[i]$`NLM-QUESTION`$`SUB-QUESTIONS`[j]$`SUB-QUESTION`$ANSWERS)
    for (k in 1:answersSize) {
      focus1 <- xmlList[i]$`NLM-QUESTION`$`SUB-QUESTIONS`[j]$`SUB-QUESTION`$ANNOTATIONS$FOCUS
      type1 <- xmlList[i]$`NLM-QUESTION`$`SUB-QUESTIONS`[j]$`SUB-QUESTION`$ANNOTATIONS$TYPE
      focus[length(focus) + 1] <- focus1
      question[length(question) + 1] <- question1
      answer1 <- xmlList[i]$`NLM-QUESTION`$`SUB-QUESTIONS`[j]$`SUB-QUESTION`$ANSWERS[k]$ANSWER$text
      answer[length(answer) + 1] <- answer1
    }
  }
}
```

6. Write the extracted data to .CSV

```
data <- data.frame(focus,question,answer)
write.csv(data, "Watson1.csv")
```

7. Extracting data and writing it to .CSV from the second list of XML file, with a slight variation as data in the answer and question node was in different format.

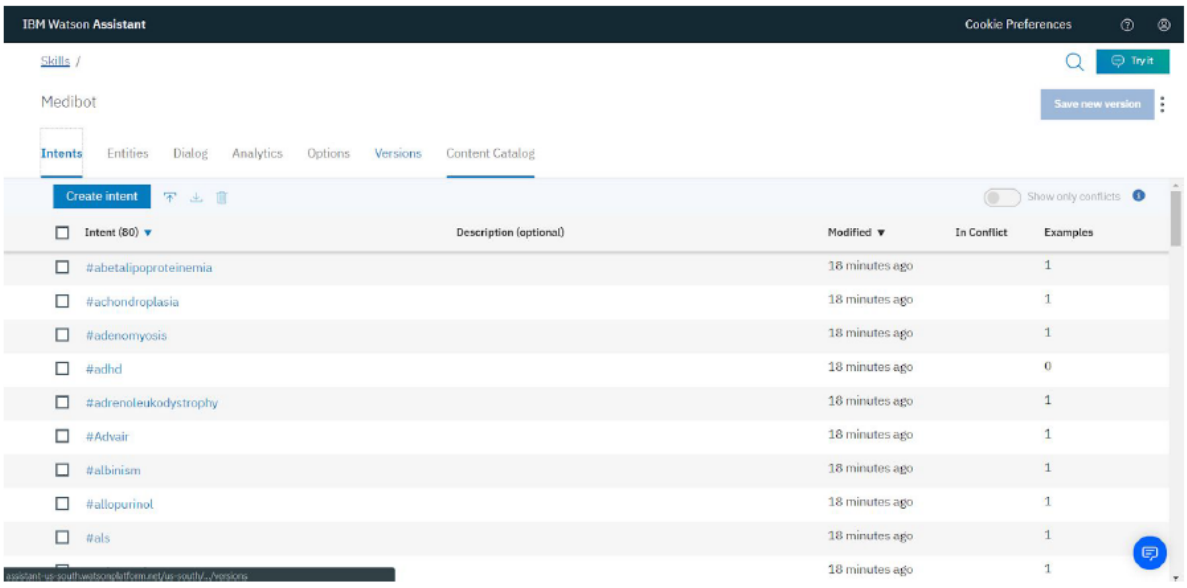
```
focus <- vector()
type <- vector()
question <- vector()
answer <- vector()
xmlList <- xmlList2
for (i in 1:length(xmlList)) {
  question1 <- xmlList[i]$`NLM-QUESTION`$MESSAGE
  focus1 <- xmlList[i]$`NLM-QUESTION`$`SUB-QUESTIONS`[1]$`SUB-QUESTION`$ANNOTATIONS$FOCUS
  type1 <- xmlList[i]$`NLM-QUESTION`$`SUB-QUESTIONS`[1]$`SUB-QUESTION`$ANNOTATIONS$TYPE
  focus[length(focus) + 1] <- focus1
  answer1 <- xmlList[i]$`NLM-QUESTION`$`SUB-QUESTIONS`$`SUB-QUESTION`$ANSWERS$ANSWER
  answer[length(answer) + 1] <- answer1
}
```

```
xmldata <- read_xml("CIS_8395_Project/TrainingDatasets/Train2.xml")
question <- xml_find_all(xmldata, "//MESSAGE")
question <- trimws(xml_text(question))
```

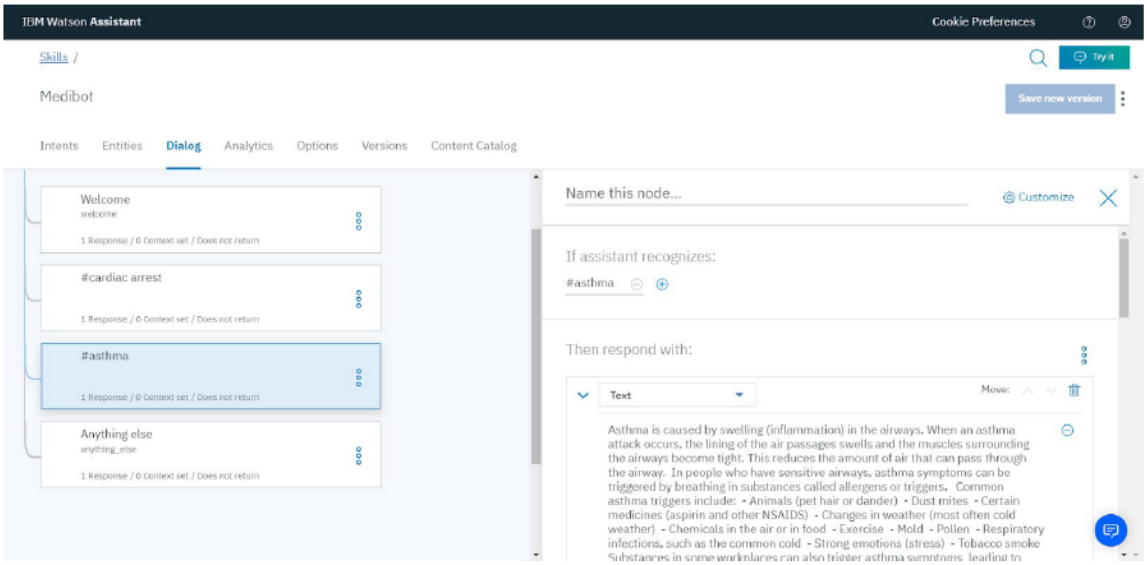
```
data <- data.frame(focus,question,answer)
write.csv(data, "Watson2.csv")
```

Data Loading

We have loaded the transformed CSV data file into IBM Watson required to train the dataset for Medibot. We have refined our data as per our requirement. Refining data consists of cleansing and shaping it. While cleansing the data, we have fixed or removed data that is incorrect, incomplete, improperly formatted, or duplicated. And by shaping data, we have customized it by filtering, sorting, combining or removing redundant questions, and performing operations.
As we manipulate our data, we built a customized Data Refinery flow that we can modify and save for future re-use. When we save the refined data set, we typically load it to a different location than where we have read it from. In this way, our source data can remain untouched by the refinement process.
We have created intents for respective questions to train the assistant which will recognize the user's questions/goals. We have also enhanced them by adding different ways people say what they are looking for.

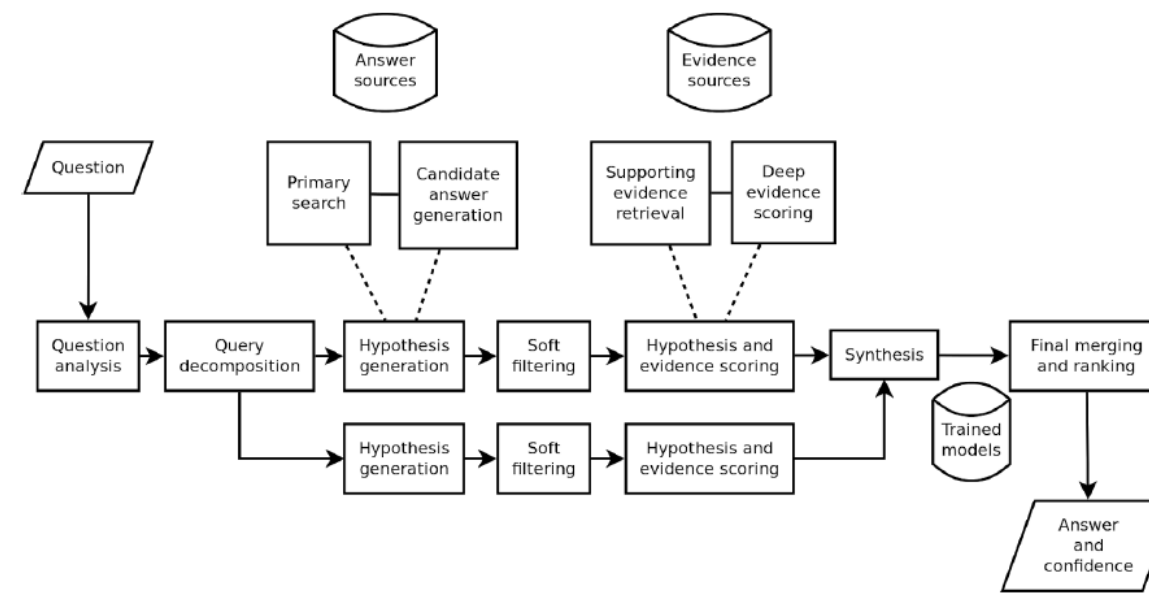


We have used IBM Watson Dialog service to automate branching conversations between a user and the Medibot. In Dialog service, various nodes are created in which the Medibot provides a response for a specific intent. The Dialog service can track and store user profile information to learn more about end users, guide them through processes based on their unique situation, or pass their information to a back-end system to help them take action and get the help they need . Medibot provides targeted responses to a wider variety of queries by building business terms in entities.



Data Storage

Our data will be stored on IBM Watson Cloud. The below picture depicts the high-level architecture and how interactions are handled.



IBM Watson Assistant mainly uses three components to get answers for the questions asked. They are:

1. Intents:

Intents are purposes or goals expressed in a customer's input, such as answering a question or processing a bill payment. By recognizing the intent expressed in a customer's input, the Watson Assistant service can choose the correct dialog flow for responding to it.

Intents Creation Overview:

a. Planning of Intents for Medibot:
   We first considered what Users might want to ask and would Medibot will be able to handle on their behalf. For example, if Medibot wants to help users to answer related asthma problems we have added #asthma intent.

b. Teaching IBM Watson regarding our Intents:

We have taught Watson the business intents that Medibot needs to handle for the Users.

For each intent, we have provided 10 examples of utterances that Users typically use to indicate their goal. We have used the transformed data to provide user examples which are tailored specifically. For example, a user example of Medibot is 'i am having a respiratory problem. what should i do?'.

The User examples provided are used by our assistant to build a machine learning model that recognizes the same and similar type of utterances and map them to the appropriate intent.

2. Entities:

Entities represent information in the user input that is relevant to the user's purpose.
If intents represent verbs (the action a user wants to do), entities represent nouns (the object of, or the context for, that action). For example, when the intent is to get a weather forecast, the relevant location and date entities are required before the application can return an accurate forecast.

Recognizing entities in the user's input helped us to craft more useful, targeted responses. For example, we have a **#asthma** intent. When a user makes a request that triggers the #asthma intent, the Medibot response reflects an understanding of what *something* is that the user wants to ask Medibot asthma-related. For example, we have added a **@air_disease** entity and then used it to extract information from the user input about the disease that the User wants to ask. Finally, we have added multiple responses to the dialog tree with wording that differs based on the **@air_disease** value that is detected in the user's request.

3. Dialogues:

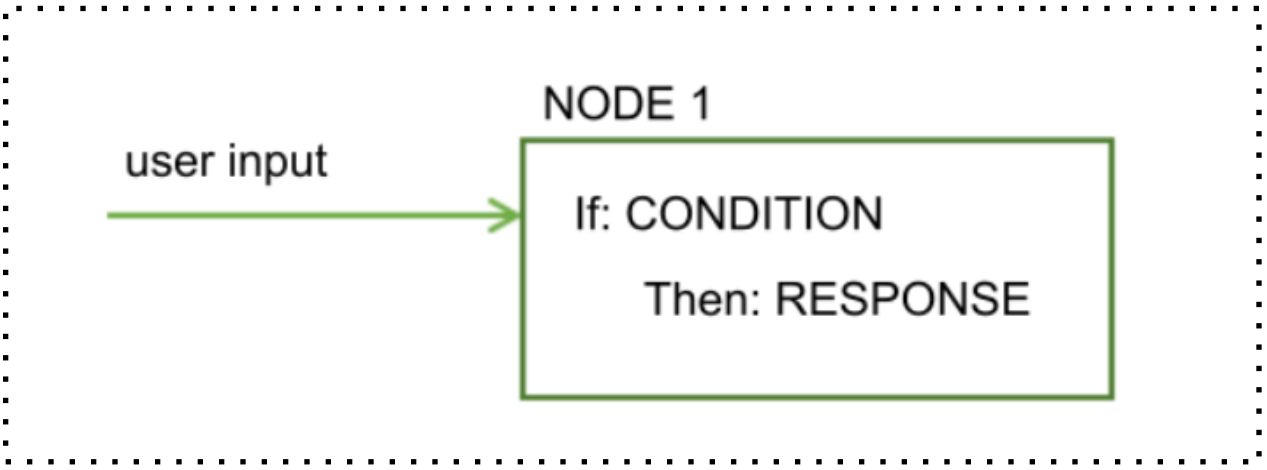
The dialog uses the intents that are identified in the user's input, plus context from the application, to interact with the user and ultimately provide a useful response.

The dialog matches intents (what users say) to responses (what the Medibot says back). The response might be the answer to a question such as ‘What is the consensus of medical doctors as to whether asthma can be cured?’. The intent and entity might be enough information to identify the correct response, or the dialog might ask the user for more input that is needed to respond correctly.

The dialog is represented graphically in the Watson Assistant tool as a tree. A branch is created to process each intent that you want your conversation to handle. A branch is composed of multiple nodes.

Dialog Nodes:

Each dialog node contains, at a minimum, a condition and a response.



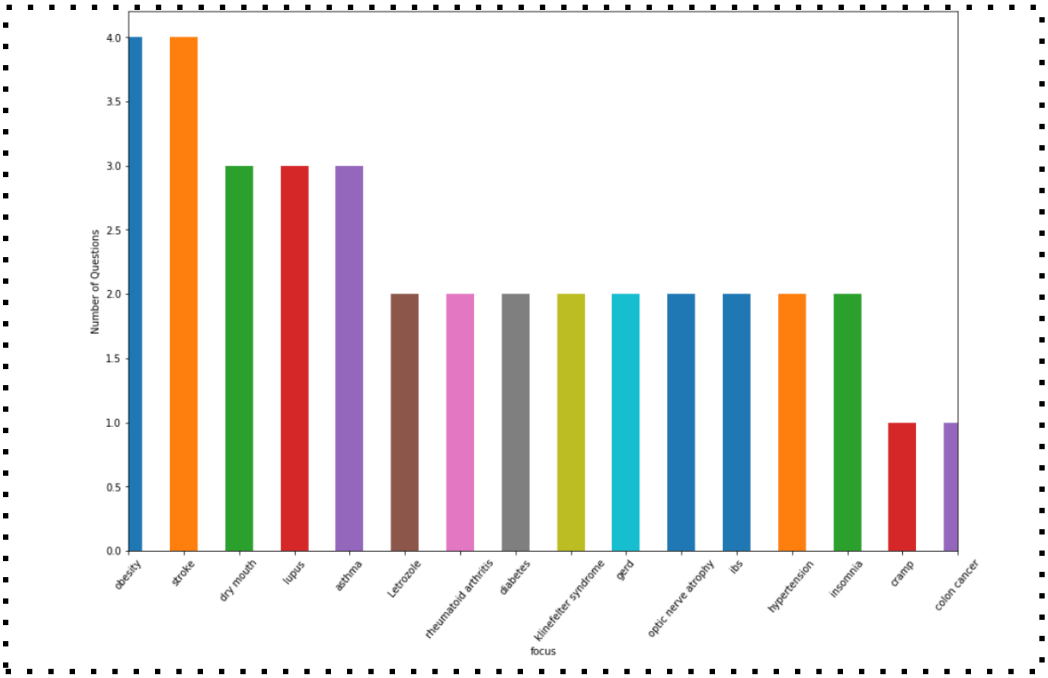
- **Condition**: It specifies the information that must be present in the user input for this node in the dialog to be triggered. The information is typically a specific intent. It might also be an entity type, an entity value, or a context variable value.
- **Response:** The utterance that Medibot uses to respond to the user. The response can also be configured to show an image or a list of options, or to trigger programmatic actions.

We can think of the node as having an if/then construction: if this condition is true, then return this response.
For example, a specific node is triggered if the natural language processing function of Medibot determines that the user input contains the #asthma intent. As a result of the node being triggered, Medibot responds with an appropriate answer.

Visualizations

Data visualizations are the presentation of data in a practical or graphical format. The way human brain processes information, using charts or graphs to visualize large amounts of complex data is easier than myriad lines of reports. In our product Medi Chatbot, we are showing visualizations of the input data. Primitive questions like how many topics can a chatbot answer, how many questions are there in each category etc are very well elucidated in the following visualizations. Visualizations not only helps the decision makers to understand the in and out of any product but also helps developers to get an insight into the gaps in the data.

Below bar chart is showing a number of focuses and their corresponding number of questions. We used a bar chart because we wanted to compare the different number of focuses and rank them according to the number of questions they hold. Although there are 120 focuses, we are showing only 16 focuses on the bar chart. Every focus has a different color for easy identification. Through this chart, we could anticipate the need to increase the questions is a particular focus.



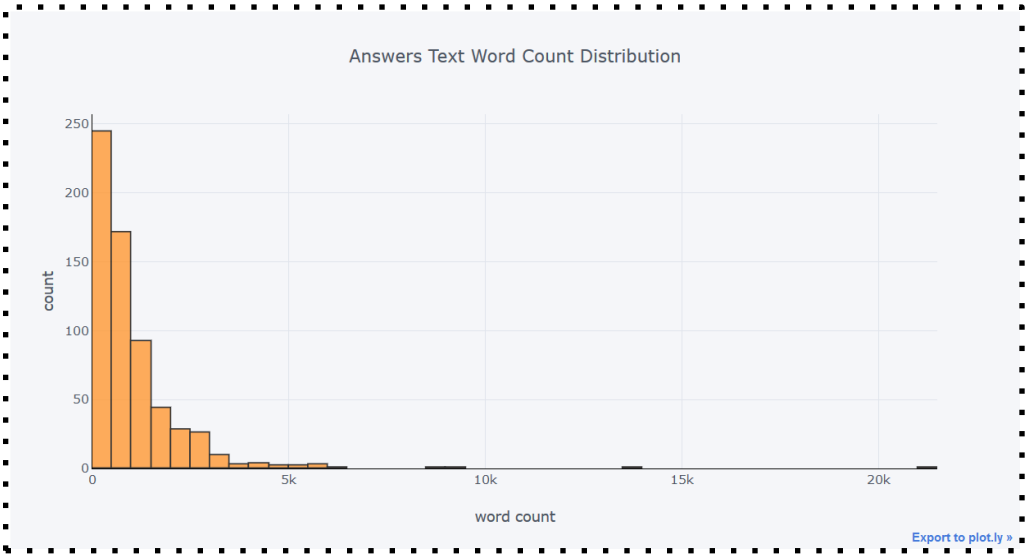
Word cloud is an image composed of words used in a particular text or subject, in which the size of each word indicates its frequency or importance. Having said that the idea behind generating the word cloud of all the questions gives an insight into topics that are being frequently asked. More focus can be emphasized on the topics per se - amyloidosis, cardiac, which we can see in the below word cloud. With the change in the frequency of topics in the word cloud, we can shift our focus the frequent topics as well.

Similarly, we generated a word cloud of all the focuses in our data set. Terms like congenital, syndrome, stroke are the frequently enquired disease. Since we know the common diseases people are suffering, we can be more cognizant about keeping our database update for these diseases.

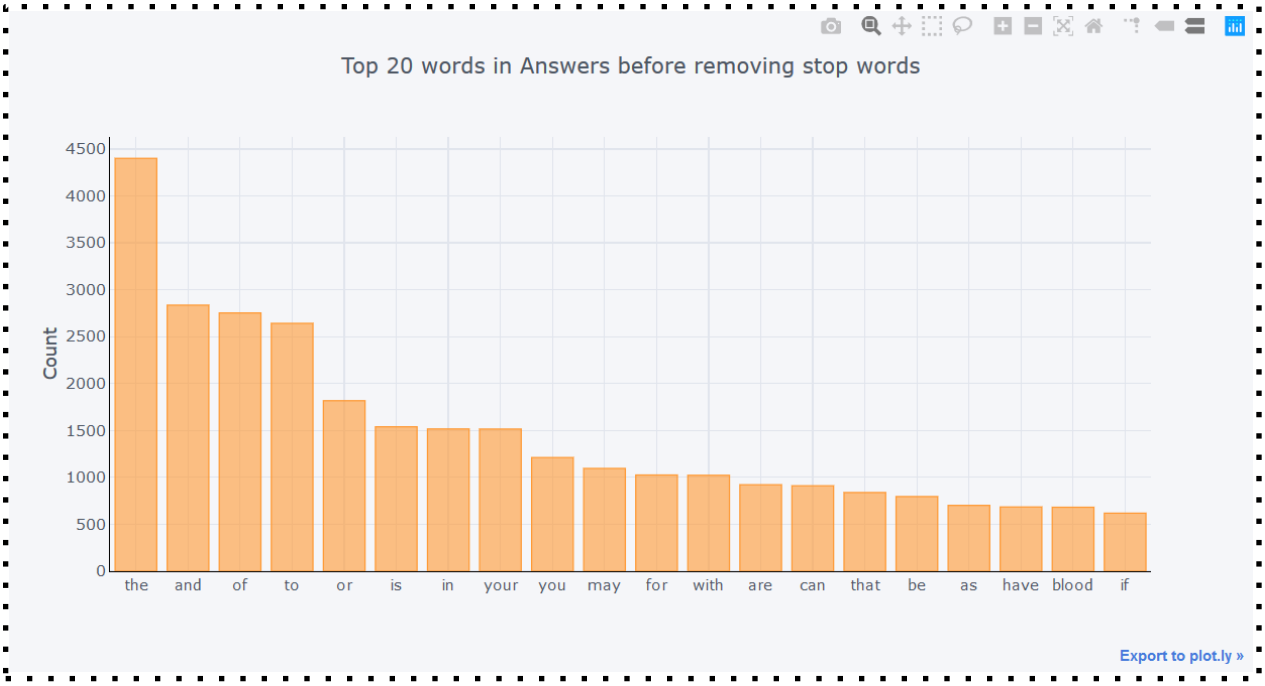
We have used Python to generate all the visualizations, below is the code snippet of same.

```
import numpy as np
import pandas as pd
from os import path
from PIL import Image
import matplotlib.pyplot as plt
% matplotlib inline
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
df = pd.read_csv("C:/Users/mpgvg/Documents/Watson1.csv",encoding = 'unicode_escape')
df.head()
focus= df.groupby("focus")
plt.figure(figsize=(15,10))
focus.size().sort_values(ascending=False).plot.bar()
plt.xlim(0,15)
plt.xticks(rotation=50)
plt.ylabel("Number of Questions")
plt.show()
questions= df.question[0]
wordcloud = WordCloud().generate(questions)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
wordcloud = WordCloud(max_font_size=50, max_words=100, background_color="white").generate(questions)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
focus= df.focus
focus = " ".join(review for review in focus)
wordcloud = WordCloud(max_font_size=50, max_words=100, background_color="white").generate(focus)
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

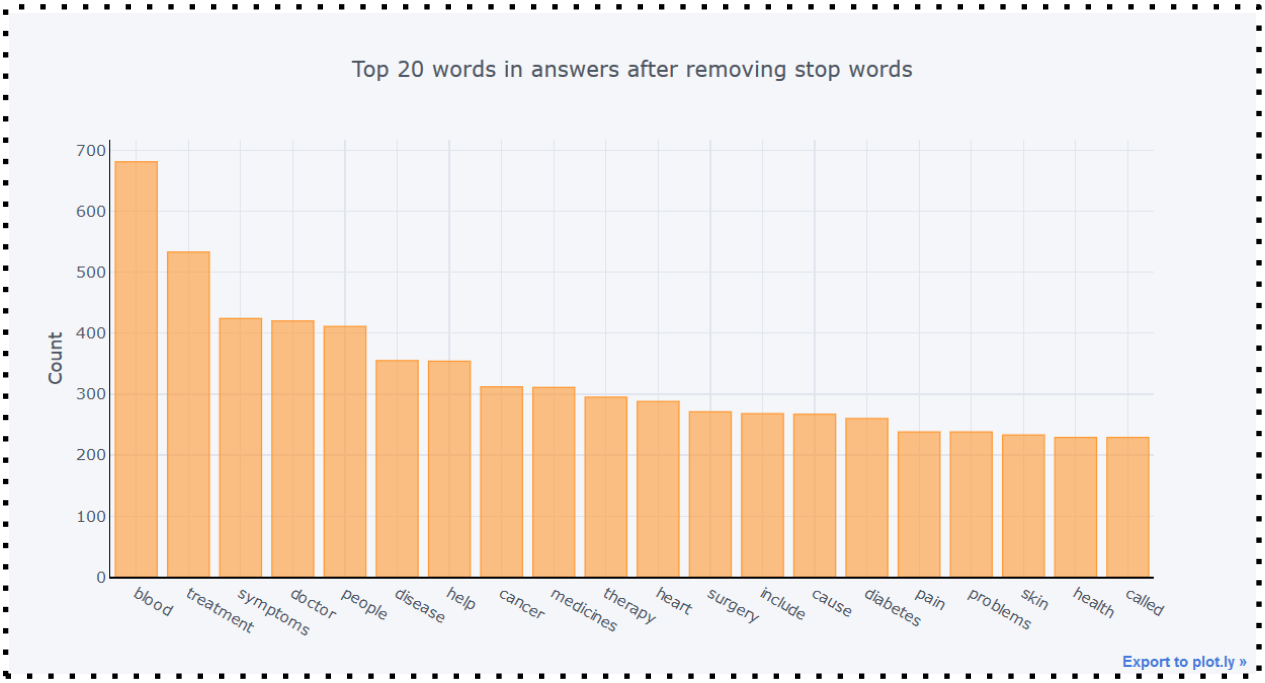
We did a distribution of the answers based on the word count in them. And found out the majority of the answers had around 250 words in them. The below plot shows the distribution helping us find the verbosity of the answers.



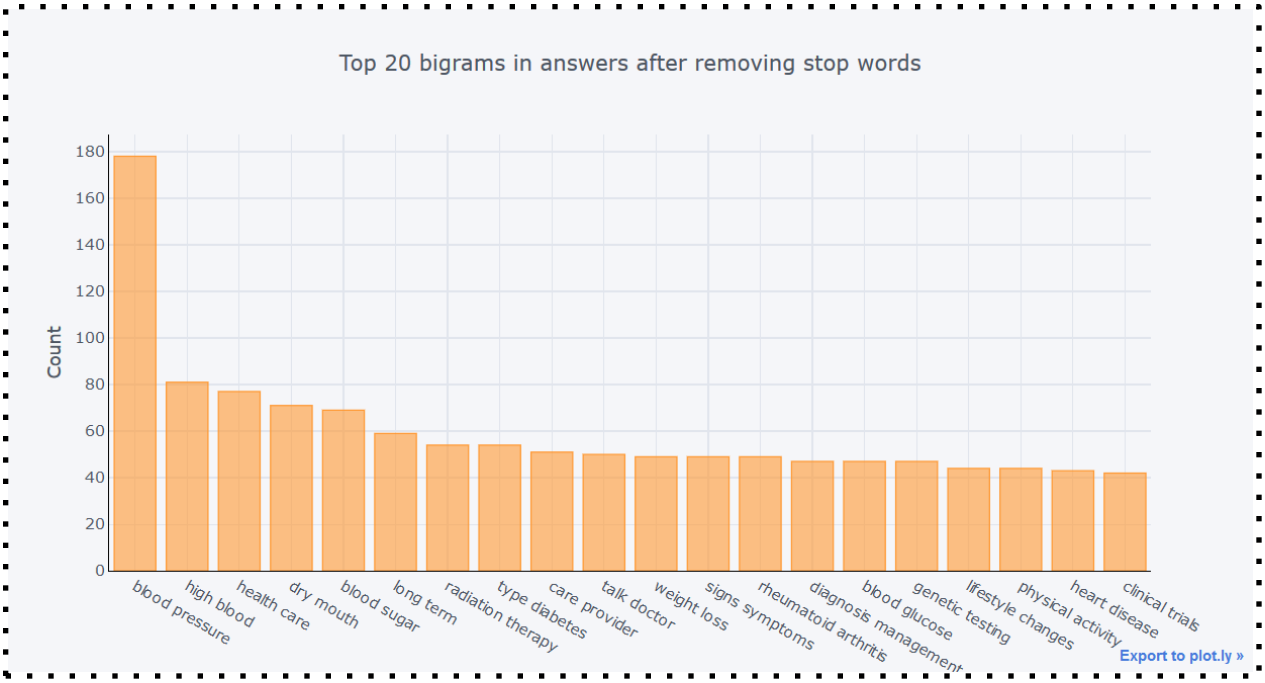
We then did extract N-grams for further analysis using the Scikit-Learn library's 'CountVectorizer function'. Below is the plot of Unigrams in the answers text before removing the stop words. Hence we can see that most of the common stop words have the upper hand here.



Then we did a plot of Unigrams after removing the stopwords. From the below plot we can see that now we have more words that are health related.



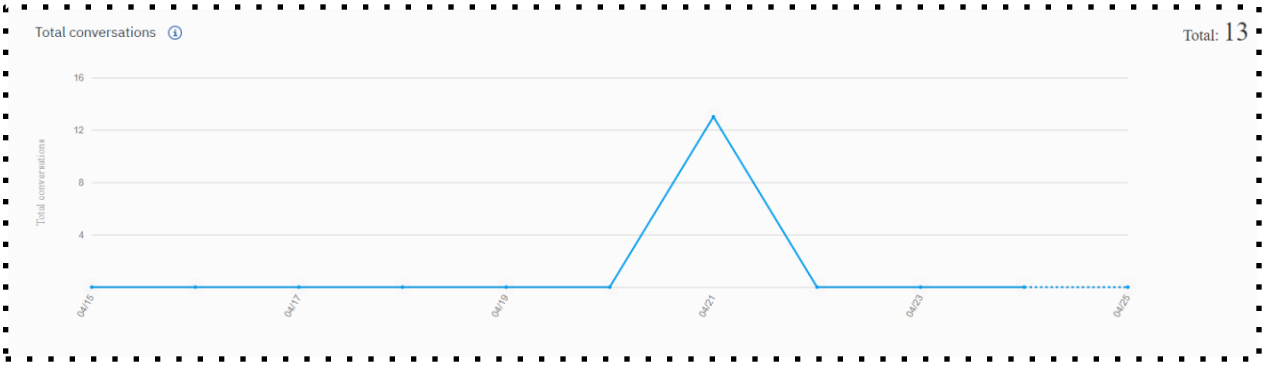
Lastly, we did a distribution of bigrams after removing the stop words. From the results below we can see that 'blood pressure' is the most common bigram in the answers.



Visualizations on IBM Watson Assistant:

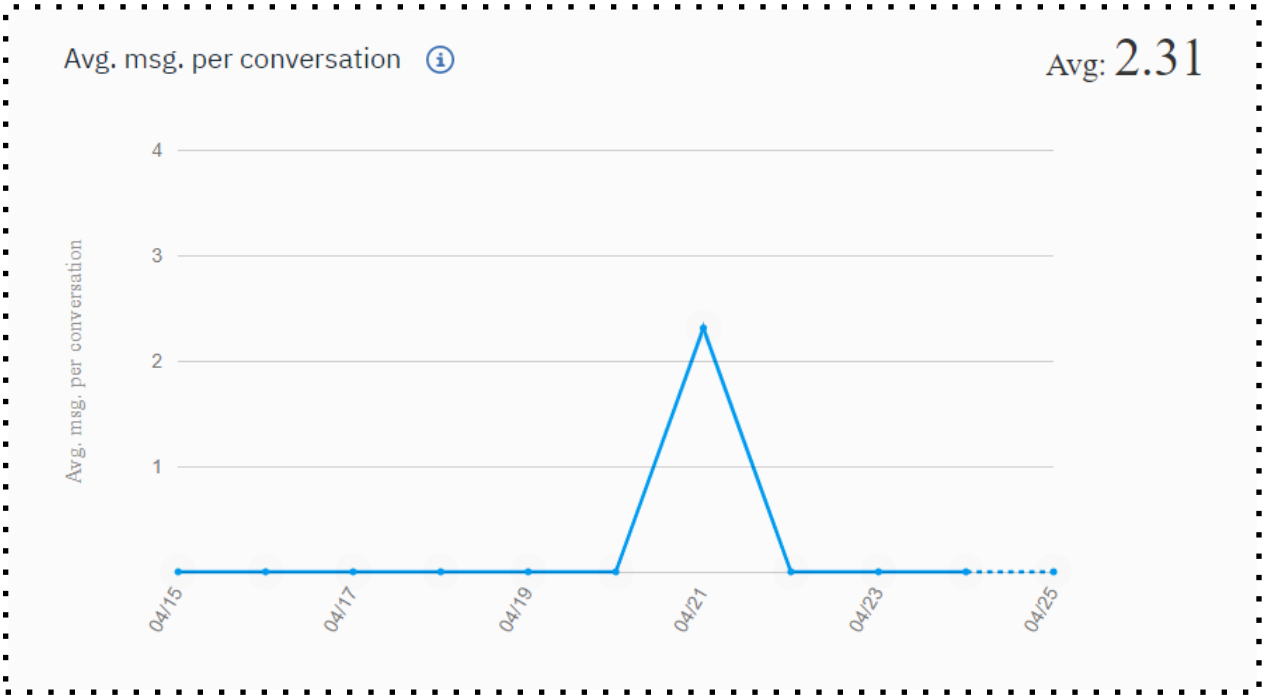
Once our bot consumed the data and after training it we ran a test of asking it some questions. During each of these conversations, the IBM Watson platform captures data relevant to these conversations which can be helpful insights on the bot's performance.

The below graph depicts the **total number of conversations** between active users and Medibot application, during the selected time period. From this, we can analyze how actively the users are conversing with the Medibot for a particular time period.

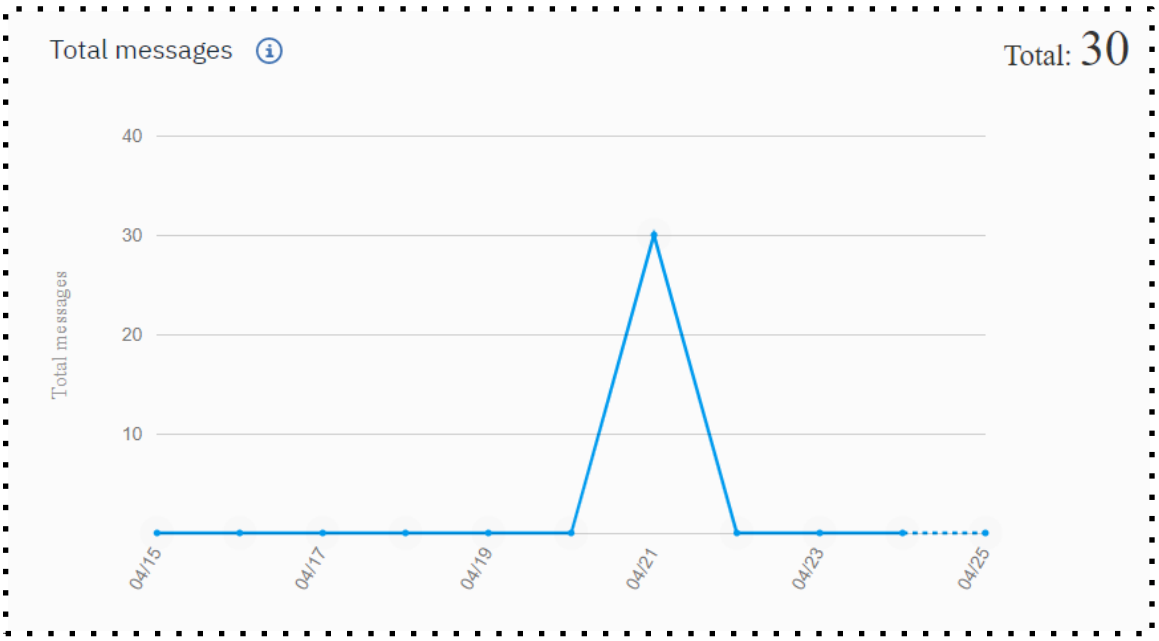


We have generated **average messages per conversation** visualization which depicts the total messages received during the selected date range divided by the total conversations during the

selected date range. From this, we can get to know what is the level of understanding of various topics for a particular time period.



Next visualization tells us about the **total messages** that Medibot application received during the selected date range. We can manage our analytics for various focuses as per the conversation in our Medibot by which we can get to know the popularity and usage pattern of the Medibot application.



NLP:

Computers are great at working with standardized and structured data like database tables and financial records. They are able to process that data much faster than we humans can. But for us humans don't communicate in "structured data" nor do we speak binary! We communicate using words, a form of unstructured data.

Unfortunately, computers suck at working with unstructured data because there are no standardized techniques to process it. When we program computers using something like C++, Java, or Python, we are essentially giving the computer a set of rules that it should operate by. With unstructured data, these rules are quite abstract and challenging to define concretely.

Human's have been writing things down for thousands of years. Over that time, our brain has gained a tremendous amount of experience in understanding natural language. When we read something written on a piece of paper or in a blog post on the internet, we understand what that thing really means in the real-world. We feel the emotions that reading that thing elicits and we often visualize how that thing would look in real life.

Natural Language Processing (NLP) is a subfield of Artificial Intelligence that is focused on enabling computers to understand and process human languages, to get computers closer to a human-level understanding of language. Computers don't yet have the same intuitive understanding of natural language that humans do. They can't really understand what the language is really trying to say. In a nutshell, a computer can't read between the lines.

That being said, recent advances in Machine Learning (ML) have enabled computers to do quite a lot of useful things with natural language! Deep Learning has enabled us to write programs to perform things like language translation, semantic understanding, and text summarization. All of these things add real-world value, making it easy for you to understand and perform computations on large blocks of text without the manual effort. Let's start with a quick primer on how NLP works conceptually.

The reason why NLP is hard:

The process of reading and understanding language is far more complex than it seems at first glance. There are many things that go in to truly understanding what a piece of text means in the real-world. For example, what do you think the following piece of text means?

"Steph Curry was on fire last nice. He totally destroyed the other team"

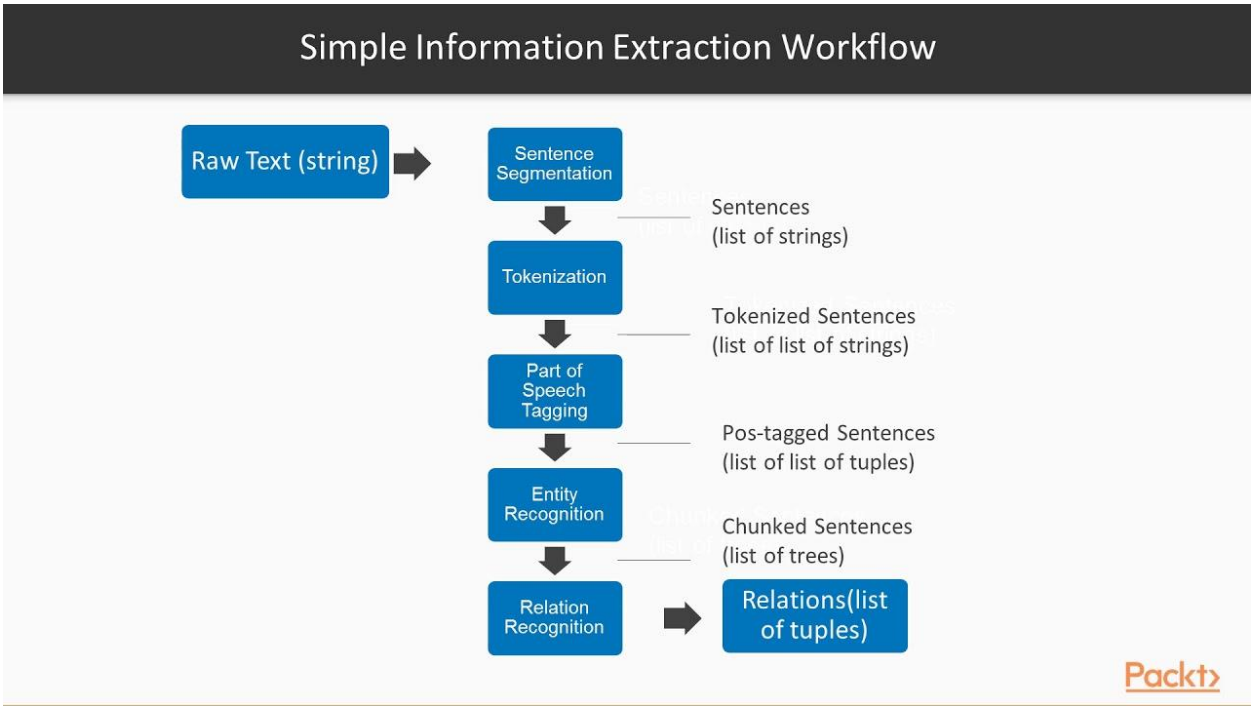
To a human, it's probably quite obvious what this sentence means. We know Steph Curry is a basketball player; or even if you don't we know that he plays on some kind of team, probably a

sports team. When we see “on fire” and “destroyed” we know that it means Steph Curry played really well last night and beat the other team.

Computers tend to take things a bit too literally. Viewing things literally like a computer, we would see “Steph Curry” and based on the capitalisation assume it’s a person, place, or otherwise important thing which is great! But then we see that Steph Curry “was on fire”…. A computer might tell you that someone literally lit Steph Curry on fire yesterday!

Thanks to Machine Learning we can actually do some really clever things to quickly extract and understand information from the natural language!

NLP Process:

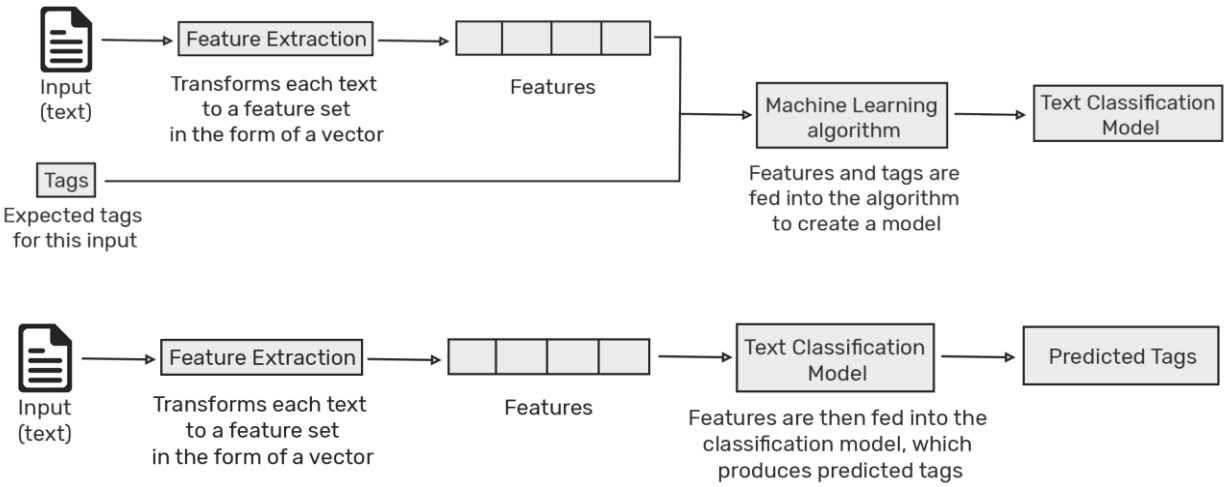


The above figure shows a common NLP pipeline. These are the most commonly used techniques to achieve good results in natural language processing.

1. **Input:** This is usually a raw text document or a collection of documents that we are using for the NLP. It can be data from twitter for sentiment analysis or customer reviews etc.
2. **Sentence Segmentation:** In this step we break down the entire document into individual sentences for easier processing.
3. **Tokenization:** Here we extract the individual words that we call tokens from the individual sentences. To further improve the quality we remove stop words like ‘it’,’they, ‘I’ etc which are usually not useful. Also, certain common words can be removed.

4. **Part of speech tagging:** In this step, we tag each token to its respective part of speech. Example: jump is tagged as a verb, gently is tagged as an adverb.
5. **Entity Recognition:** Then the entities of the token are recognised.. For example London is a City or a Place so the entity of London would be City or Place.
6. **Relation Recognition:** In this step the relation between the entities is recognized. For example in "London is the capital of England" the tokens London(entity: city) and England(entity: country) are related by capital.

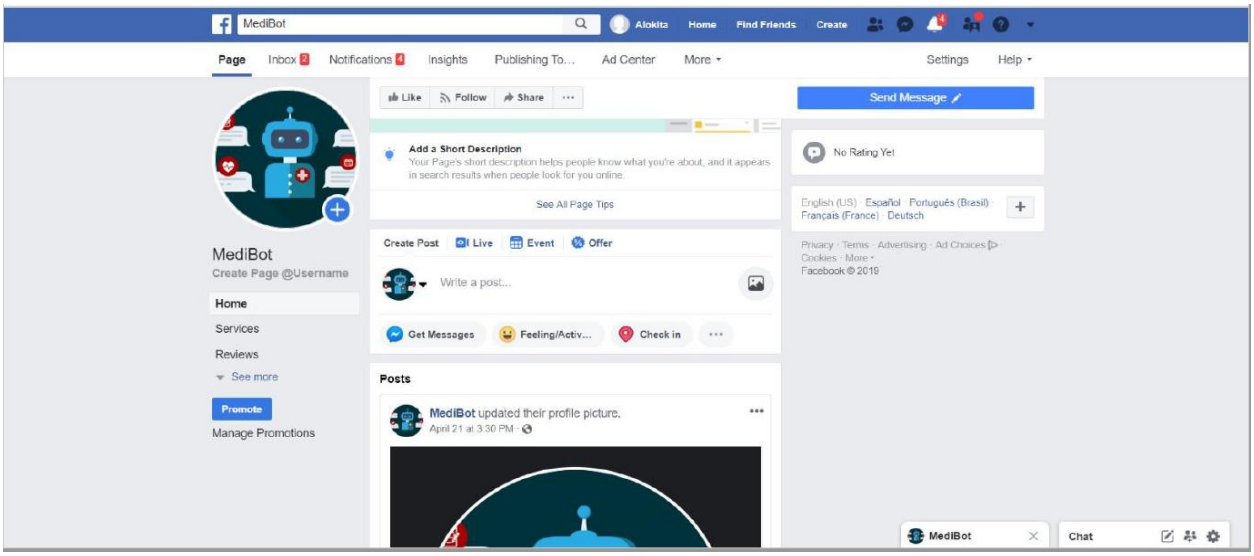
   Also certain machine learning algorithms like Naive Bayes and SVM can be used to classify texts to categories as shown in the below figure:

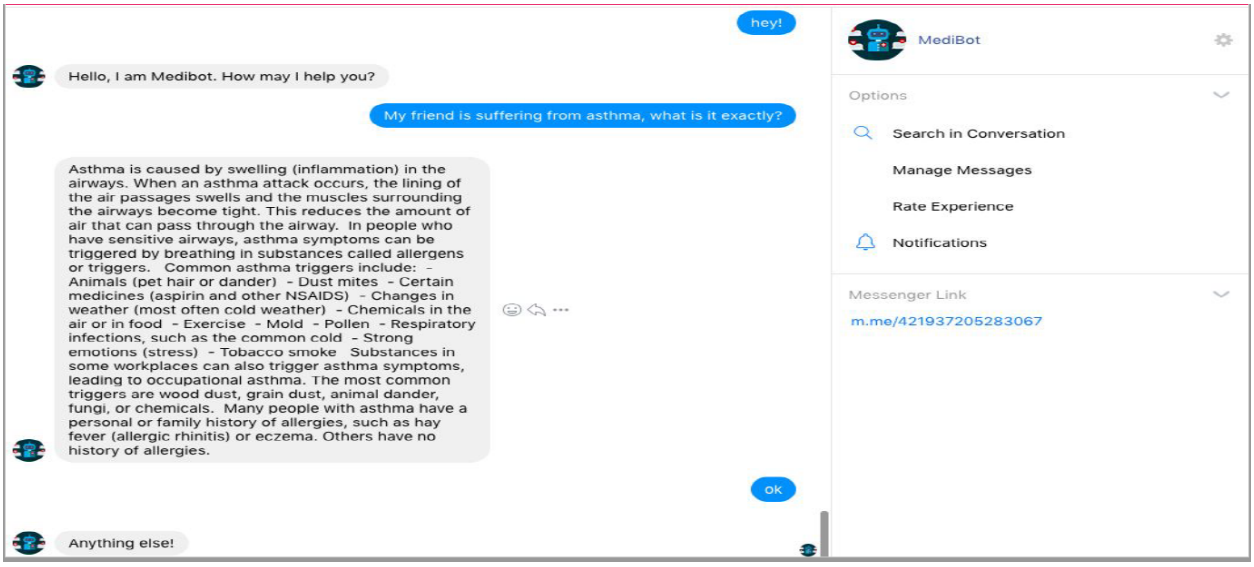


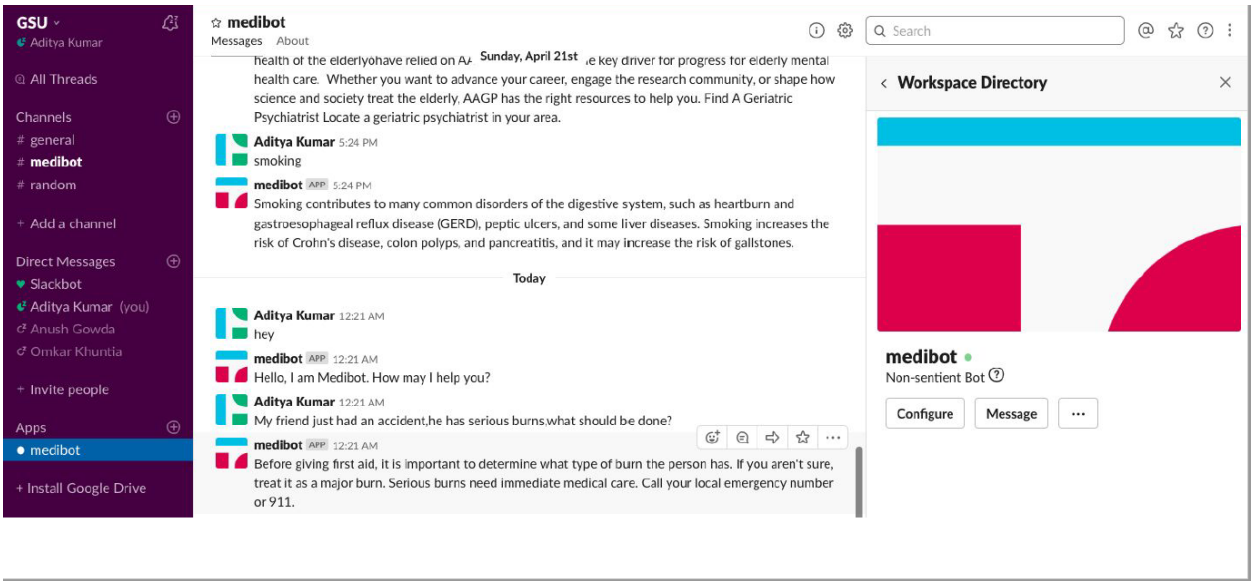
   Here the text is converted into feature sets in form of vectors. This feature set along with the labels or tags that the text belongs to is trained on a machine learning algorithm to create a model. Next the test data can be converted into feature sets as well and can be used to predict the tags or labels that they belong to using the previously created classification model.

User Interface

For the end user, there has to be an interface from which they can interact without MediBot. Instead of creating a new user interface leveraging latest technology like React, we have decided to explore the integration of IBM Watson with popular UIs facebook and Slack. These user interfaces are so popular and smooth to use that IBM Watson has provided configurations to communicate with them. We have integrated our Medibot with facebook messenger and slack.


MediBot
Alokita
Home
Find Friends
Create
Page
Inbox 2
Notifications 4
Insights
Publishing To...
Ad Center
More
Settings
Help
Like
Follow
Share
Send Message
Add a Short Description
Your Page's short description helps people know what you're about, and it appears in search results when people look for you online.
No Rating Yet
See All Page Tips
English (US) · Español · Português (Brasil) · Français (France) · Deutsch
Privacy · Terms · Advertising · Ad Choices · Cookies · More · Facebook © 2019
MediBot
Create Page @Username
Home
Services
Reviews
See more
Promote
Manage Promotions
Create Post
Live
Event
Offer
Write a post...
Get Messages
Feeling/Activ...
Check in
Posts
MediBot updated their profile picture.
April 21 at 3:30 PM
MediBot
Chat


hey!
Hello, I am Medibot. How may I help you?
My friend is suffering from asthma, what is it exactly?
Asthma is caused by swelling (inflammation) in the airways. When an asthma attack occurs, the lining of the air passages swells and the muscles surrounding the airways become tight. This reduces the amount of air that can pass through the airway. In people who have sensitive airways, asthma symptoms can be triggered by breathing in substances called allergens or triggers. Common asthma triggers include: - Animals (pet hair or dander) - Dust mites - Certain medicines (aspirin and other NSAIDS) - Changes in weather (most often cold weather) - Chemicals in the air or in food - Exercise - Mold - Pollen - Respiratory infections, such as the common cold - Strong emotions (stress) - Tobacco smoke Substances in some workplaces can also trigger asthma symptoms, leading to occupational asthma. The most common triggers are wood dust, grain dust, animal dander, fungi, or chemicals. Many people with asthma have a personal or family history of allergies, such as hay fever (allergic rhinitis) or eczema. Others have no history of allergies.
ok
Anything else!
MediBot
Options
Search in Conversation
Manage Messages
Rate Experience
Notifications
Messenger Link
m.me/421937205283067



Experience

At the end of this product development, I have gathered rich experience in multiple areas like data cleaning, data storage, architecture development, visualizing our requirements using the latest technology tools. We explored a new platform- IBM Watson. We understand how they are using Natural Language Processing to train medical chatbot. We received in-depth knowledge of information extraction workflow using NLP. We also got an idea of how IBM Watson using permutations and combinations of various machine learning algorithms to increase the accuracy of chatbot. At the end of this course, we wanted to explore a new cloud platform apart from Microsoft azure and Amazon AWS, so we choose IBM Watson. This course was very beneficial for me in my job hunt as well. Through this, I understand overall product development and all the latest technology buzz that we need in this tech-savvy industry. I specially want to thank Professor Gandapodi for showing me the path and enhance my envision for a competitive market.