Leveraging Big Data Analytics for Disease Classification Using Chatbot-Generated Features

Aditya Balasubramanian

Boston University

December 7, 2023

Abstract

In the rapidly evolving landscape of healthcare, the influx of medical data has become unprecedented, paving the way for transformative advancements. This project proposal delves into the realm of leveraging big data analytics to enhance disease classification, facilitated by the burgeoning capabilities of chatbot technology. The core focus is on constructing a robust classification model that harnesses features generated through chatbot interactions. The medical dataset, accessible at HuggingFace. The healthcare ecosystem is immersed in an abundance of data, encompassing electronic health records and interactions with chatbots. Natural Language Processing (NLP) is crucial in this context due to its ability to empower machines to comprehend and respond to human language. In healthcare, where interactions often involve nuanced and context-dependent information, NLP becomes instrumental. It enables the model to discern the subtleties of medical conversations, facilitating accurate feature extraction and enhancing the overall performance of the disease classification model. The integration of NLP ensures that the model can effectively interpret, analyze, and classify diseases based on the intricate details embedded in chatbot-generated textual features, ultimately contributing to more informed and automated healthcare decisionmaking.

1 Introduction

The contemporary landscape of the healthcare industry is undergoing a transformative shift, primarily attributed to the burgeoning availability of medical data. This surge in data, ranging from electronic health records to real-time chatbot interactions, presents an unprecedented opportunity to revolutionize disease classification. In tandem with these advancements, chatbot technology has emerged as a pivotal player in healthcare, streamlining patient interactions, and generating valuable textual data in the process.

The integration of chatbots into healthcare not only facilitates efficient communication but also yields a wealth of textual features derived from patient queries, responses, and explanations. Leveraging the power of big data analytics in this context holds immense promise for enhancing disease classification methodologies. By harnessing the latent information embedded within chatbot-generated features, we aim to develop a robust disease classification model capable of automating and improving the accuracy of diagnostic processes.

The healthcare system generates an astronomical volume of data daily, encompassing diverse modalities such as electronic health records and textual exchanges from chatbot interactions. However, the challenge lies in effectively harnessing this data to automate disease classification. Manual classification processes are time-consuming, prone to errors, and may not keep pace with the growing volume of medical data.

The project’s core problem statement revolves around addressing the gap in automating disease classification by capitalizing on big data analytics and natural language processing (NLP) techniques. Despite the vast potential in healthcare data, there is a need for sophisticated models that can discern patterns and nuances in textual features obtained from chatbot interactions. This project seeks to bridge this gap by developing a model that can accurately classify diseases, thereby contributing to the efficiency and precision of diagnostic procedures in healthcare.

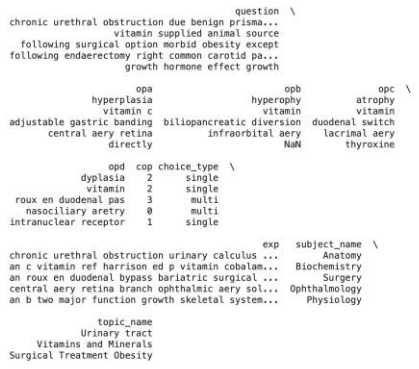
2 Dataset

The dataset used for this project is sourced from Hugging Face Datasets, a repository known for its diverse and comprehensive collection of datasets. The medical dataset at our disposal is tailored to the medical question-answering domain and aligns with the project's objectives.

2.1 Key Features

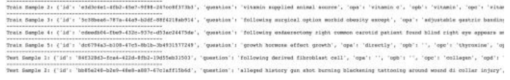
- Question: Represents queries related to patient symptoms or concerns.
- Options (opa, opb, opc, opd): Multiple-choice options associated with each question.
- Explanation (exp): Provides additional context or elaboration on the correct answer.
- Topic Name: Denotes the medical topic or category of the question.
- Choice Type: Indicates the type of choice, providing context on the format of answer options.
- COP (Target Variable): Stands for Correct Option Position, serving as the target variable for disease classification.

2.2 Snippets of the Dataset



```
                                            question  \
0  chronic urethral obstruction due benign prisma...
1                      vitamin supplied animal source
2    following surgical option morbid obesity except
3  following endaerectomy right common carotid pa...
4                         growth hormone effect growth

                        opa                       opb             opc  \
0               hyperplasia                 hyperophy         atrophy
1                 vitamin c                   vitamin         vitamin
2  adjustable gastric banding  biliopancreatic diversion  duodenal switch
3         central aery retina         infraorbital aery   lacrimal aery
4                   directly                       NaN       thyroxine

                    opd  cop choice_type  \
0              dyplasia    2      single
1               vitamin    2      single
2   roux en duodenal pas    3       multi
3     nasociliary aretry    0       multi
4  intranuclear receptor    1      single

                                               exp  subject_name  \
0  chronic urethral obstruction urinary calculus ...       Anatomy
1  an c vitamin ref harrison ed p vitamin cobalam...  Biochemistry
2  an roux en duodenal bypass bariatric surgical ...       Surgery
3  central aery retina branch ophthalmic aery sol...  Ophthalmology
4  an b two major function growth skeletal system...    Physiology

                topic_name
0            Urinary tract
1     Vitamins and Minerals
2  Surgical Treatment Obesity
```

Figure 1: Dataset



```
Map: 100% 182822/182822 [08:57<00:00, 378.72 examples/s]
Train Sample 1: {'id': 'e9ad821a-c438-4965-9f77-760819dfa155', 'question': 'chronic urethral obstruction due benign prismatic hyperplasia lead following change
-----------------------------------------------
Train Sample 2: {'id': 'e3d3c4e1-4fb2-45e7-9f88-247cc8f373b3', 'question': 'vitamin supplied animal source', 'opa': 'vitamin c', 'opb': 'vitamin', 'opc': 'vitam
-----------------------------------------------
Train Sample 3: {'id': '5c38bea6-787a-44a9-b2df-88f4218ab914', 'question': 'following surgical option morbid obesity except', 'opa': 'adjustable gastric bandin
-----------------------------------------------
Train Sample 4: {'id': 'cdeedb04-fbe9-432c-937c-d53ac24475de', 'question': 'following endaerectomy right common carotid patient found blind right eye appears m
-----------------------------------------------
Train Sample 5: {'id': 'dc6794a3-b108-47c5-8b1b-3b4931577249', 'question': 'growth hormone effect growth', 'opa': 'directly', 'opb': '', 'opc': 'thyroxine', 'o
-----------------------------------------------
Test Sample 1: {'id': '84f320d3-fca4-422d-8fb2-19d55eb31503', 'question': 'following derived fibroblast cell', 'opa': '', 'opb': '', 'opc': 'collagen', 'opd': '
-----------------------------------------------
Test Sample 2: {'id': 'bb85e248-b2e9-46e8-a887-67c1aff15b6d', 'question': 'alleged history gun shot burning blackening tattooing around wound di collar injury',
```

Figure 2: Dataset

3 Data Processing

Data preprocessing is a crucial step in the data analysis and machine learning pipeline that involves cleaning and transforming raw data into a format suitable for analysis or model training. It's often said that the success of a machine learning project is heavily dependent on the quality of the data, and data prepossessing plays a key role in enhancing this quality. We prepossessed the dataset using the following steps:

3.1 Tokenization, Lowercasing, and Stopword Removal

Tokenization is the process of breaking down text into individual tokens or words. This step is essential for converting raw text into meaningful units, allowing for further analysis. Lowercasing is employed to ensure uniformity by converting all text to lowercase, reducing the complexity of the dataset. Stopword removal involves eliminating common words (e.g., "the," "and," "is") that do not contribute significant meaning to the text, focusing on content-bearing words.

3.2 Lemmatization and Special Character Removal

Lemmatization involves reducing words to their base or root form, enhancing feature extraction by standardizing variations of a word. Special character removal is performed to ensure text cleanliness by eliminating non-alphanumeric characters.

3.3 Handling Missing Values

Addressing missing values is crucial for ensuring model robustness. Depending on the nature of the missing data, strategies include imputation, where missing values are replaced with estimated values based on the available data, or removal, where rows with missing values are excluded from the dataset.

4 Model Building

Model building involves creating a mathematical algorithm to make predictions or decisions based on input data in machine learning. The process includes defining the problem, collecting and preparing representative data, splitting it into training, validation, and test sets, selecting a suitable model architecture, tuning hyperparameters, training the model, and evaluating its performance on validation and test sets. Hyperparameters are fine-tuned to optimize model performance, and the final model is deployed in realworld settings for predictions on new data. Continuous monitoring and iteration are essential for improvement, considering factors like overfitting and underfitting. A solid understanding of the domain and problem is crucial throughout the process to make informed decisions and build a model that generalizes well to unseen data. For the context of this project we choose Random Forest, BERT model and RoBERTa model.

4.1 Traditional Machine Learning Model (Random Forest)

4.1.1 TF-IDF Vectorization TF-IDF vectorization is applied to convert the text data into numerical vectors for input to the Random Forest classifier. The TfidfVectorizer from scikit-learn is employed, with a maximum of 5000 features. This process transforms the text features into a format suitable for traditional machine learning models.

4.1.2 Model Training and Evaluation A Random Forest classifier is trained using the TF-IDF-transformed text data and the ’cop’ column as the target variable. The classifier is configured with 100 decision trees and a random seed of 42 for reproducibility. The model is evaluated on the validation set using the accuracy metric, which measures the percentage of correctly predicted ’cop’ values.

4.1.3 Model Saving

After training and evaluation, the Random Forest model is saved to Google Cloud Storage (GCS) for future use. This enables easy retrieval and deployment of the trained model without the need for retraining.

4.2 BERT Model

4.2.1 Tokenization and Encoding

The BERT model, based on the ’bert-baseuncased’ architecture, is employed for sequence classification. The text data is tokenized and encoded using the BERT tokenizer, and special tokens are added to mark the beginning and end of each sequence. The resulting tensors are then used as input to the BERT model.

4.2.2 Training Loop The model is trained over multiple epochs, with each epoch consisting of iterations through the training data. The training loop uses PyTorch DataLoaders to efficiently load batches of data. The AdamW optimizer is employed for optimization, and a learning rate scheduler is used to adjust the learning rate during training.

4.2.3 Monitoring Training and Validation Losses

During training, both training and validation losses are monitored. This information is crucial for understanding how well the model is learning from the data and whether there is overfitting or underfitting. The losses are printed after each epoch, providing insights into the model’s performance.

4.3 Support Vector Machine (SVM) Model

4.3.1 Feature Extraction and Vectorization

For the SVM model, the text data undergoes feature extraction and vectorization. The process involves converting the textual information in the ’question’ column into a format suitable for SVM training. In this case, TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is commonly employed. Each question is represented as a numerical vector based on the importance of words within that question.

4.3.2 Model Training The SVM model is trained using the vectorized features of the text data. The scikit-learn library provides the SVC (Support Vector Classification) class for SVM classification tasks. During training, the model learns the optimal hyperplane that separates different classes in the feature space. The ’cop’ column, representing the correct option, serves as the target variable during training.

4.3.3 Hyperparameter Tuning

SVM models have hyperparameters, such as the choice of kernel (e.g., RBF), regularization parameter (C), and kernel-specific parameters (e.g., gamma for RBF). Hyperparameter tuning may be performed to find the optimal combination that maximizes the model’s performance on the training data.

4.3.4 Evaluation on Validation Set After training, the SVM model is evaluated on the validation set to assess its performance on unseen data. Metrics such as accuracy are computed to quantify the model's ability to correctly classify instances in the validation set. This evaluation helps in understanding how well the SVM model generalizes to new, previously unseen questions.

5 Chatbot

We built two different kinds of chatbots. One as explained below, using the Replicate API and another using Llama 2. Llama gave a limited flexicbility for us to work on as it was a paid service and we ran out of free tokens.

5.1 Importing Modules

The script imports the following modules:

- os: Provides interaction with the operating system, primarily used to set environment variables.
- pickle: A module for serializing and deserializing Python objects, employed to load a pre-trained scikit-learn model.
- replicate: Presumably, a custom or thirdparty library interacting with an API (likely named "REPLICATE").

5.2 Loading the Scikit-Learn Model

The script loads a pre-trained scikit-learn model from a file named finalized_model_2.sav using the pickle.load() function. The loaded model is stored in the variable model.

5.3 Defining the Chatbot Response Function

A function named chatbot_response takes a symptom as input, utilizes the loaded model to predict a health disorder based on the symptom, and returns a response string.

5.4 Setting Environment Variable

The script sets the environment variable REPLICATE_API_TOKEN to a specific value, likely an authentication token required to access an API (possibly the REPLICATE API).

5.5 Chat Loop

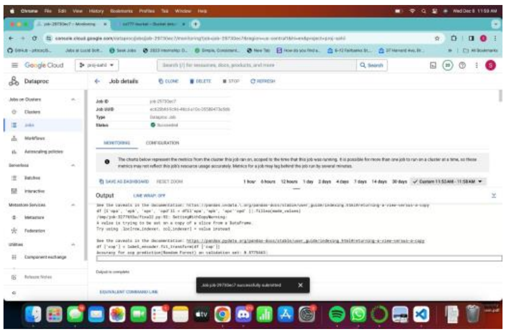
The script initiates a simple chat loop, continuously prompting the user to enter a symptom until the user types 'exit'. Inside the loop:

- User input is collected.
- If the user types 'exit', the loop breaks, and the program ends.
- Otherwise, the chatbot generates a response using the scikit-learn model and prints it.
- The script uses the REPLICATE API (presumably) by calling the replicate.run() function, passing parameters including the user input as a prompt. The output from this API call is then printed.

5.6 REPLICATE API Call

The replicate.run() function appears to make a request to the REPLICATE API using a specific model version. The API call includes parameters like top_k, top_p, temperature, max_new_tokens, and system_prompt, controlling the behavior of the language model used by the API.

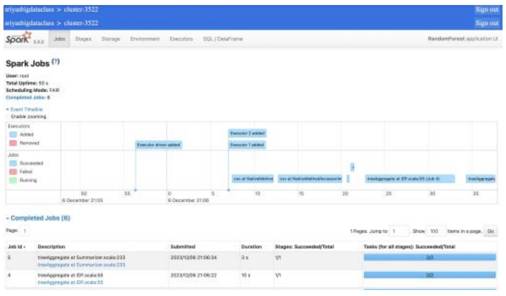
6 Results



Job ID: job-29730ec7
Type: Dataproc Job
Status: Succeeded

```
[illegible]
Accuracy for [illegible] prediction(Random Forest) on validation set: [illegible]
```

Figure 3: Accuracy for Random Forest is 57(percent)



Spark Jobs (?)

User: root
Total Uptime: 50 s
Scheduling Mode: FAIR
Completed Jobs: 6

Completed Jobs (6)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
5	treeAggregate at Summarizer.scala:233	2023/12/06 21:06:34	3 s	1/1	2/2
4	treeAggregate at IDF.scala:55	2023/12/06 21:06:22	10 s	1/1	2/2

Figure 4: Random Forest Log file

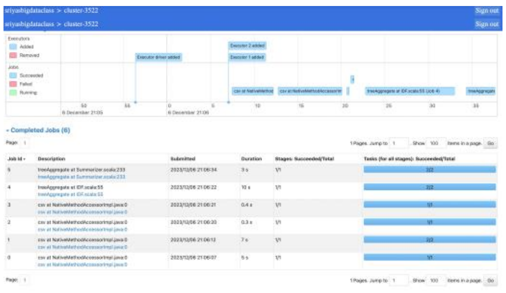
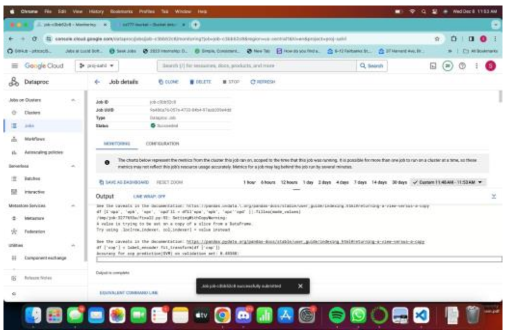


Figure 5: Random Forest Log File



```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
Try using .loc[row_indexer, col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df ['cop'] = label_encoder.fit_transform(df ['cop'])
Accuracy for cop prediction(SVM) on validation set: 0.48308
```

Figure 6: Accuracy for SVM is 48(percent)

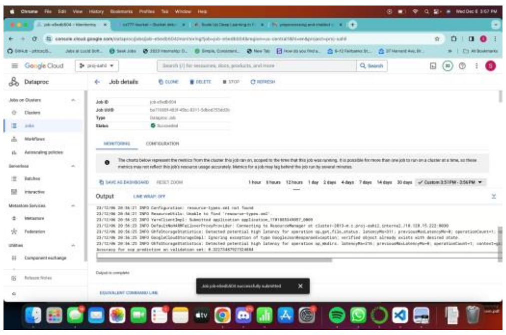


Figure 7: Log file SVM

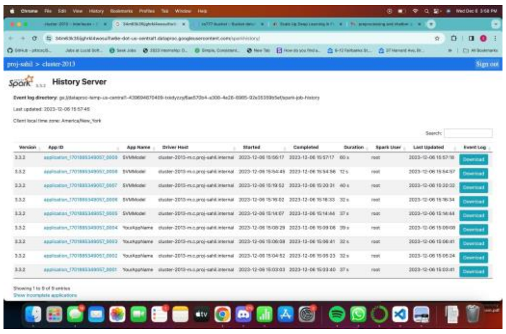
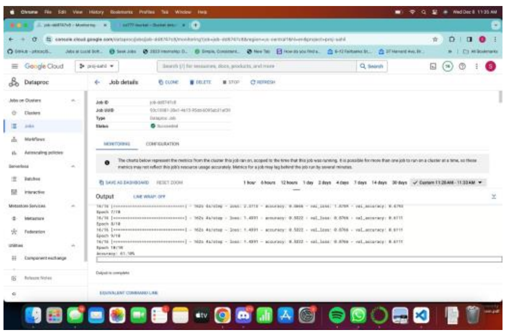
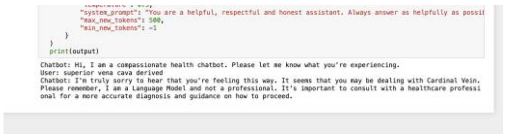


Figure 8: Log File SVM



```
Epoch 7/10
16/16 [==============================] - 162s 4s/step - loss: 1.4891 - accuracy: 0.5022 - val_loss: 0.8766 - val_accuracy: 0.6111
Epoch 8/10
16/16 [==============================] - 162s 4s/step - loss: 1.4891 - accuracy: 0.5022 - val_loss: 0.8766 - val_accuracy: 0.6111
Epoch 9/10
16/16 [==============================] - 162s 4s/step - loss: 1.4891 - accuracy: 0.5022 - val_loss: 0.8766 - val_accuracy: 0.6111
Epoch 10/10
Accuracy: 61.10%
```

Figure 9: The Accuracy for the BERT Model is 61(percent)



```
            "system_prompt": "You are a helpful, respectful and honest assistant. Always answer as helpfully as possi
            "max_new_tokens": 500,
            "min_new_tokens": -1
        }
    )
    print(output)
Chatbot: Hi, I am a compassionate health chatbot. Please let me know what you're experiencing.
User: superior vena cava derived
Chatbot: I'm truly sorry to hear that you're feeling this way. It seems that you may be dealing with Cardinal Vein.
Please remember, I am a Language Model and not a professional. It's important to consult with a healthcare professi
onal for a more accurate diagnosis and guidance on how to proceed.
```

Figure 10: Example of how the Chatbot Works

7 Conclusion and Future Scope

The envisioned goals and scope of our project encapsulate a forward-looking approach to technology and user engagement. Our primary objective is to achieve enhanced personalization, fostering stronger connections and satisfaction for users by tailoring experiences to individual preferences. Additionally, we aspire to integrate telehealth services, aiming to provide comprehensive and accessible healthcare solutions. The project also emphasizes the implementation of multimodal interaction and improved Natural Language Understanding (NLU) to create more intuitive and inclusive user interfaces. As we forge ahead, the overarching goal is to contribute to global health awareness and crisis response, utilizing technology as a catalyst for disseminating timely information and facilitating swift, impactful actions. In essence, our project's future goals align with harnessing technology to elevate user experiences, advance healthcare solutions, and contribute meaningfully to societal well-being.