Data 245 Team Project Report: Medical Symptoms Text Classification

Yijia Li
Department of Applied Data Science
San Jose State University
San Jose, USA
yijia.li@sjsu.edu

Haiyan An
Department of Applied Data Science
haiyan.an@sjsu.edu
San Jose, USA
haiyan.an@sjsu.edu

Wanyu Huang
Department of Applied Data Science
San Jose State University
San Jose, USA
wanyu.huang@sjsu.edu

Zhe Li
Department of Applied Data Science
San Jose State University
San Jose, USA
zhe.li01@sjsu.edu

Lilin Huang
Department of Applied Data Science
San Jose State University
San Jose, USA
lilin.huang@sjsu.edu

Abstract—**Since the Covid-19 pandemic, more and more people start seeking help from online medical services. Aiming to apply natural language processing and machine learning techniques in helping with the automation of the early identification of medical symptoms, we compared the performance of different combinations of machine learning algorithms and nature language process techniques. In the end, we proposed a tuned random forest classifier using the Word2Vec extracted features with 99.69% accuracy, 99.69% precision, and 99.71% recall score, which is implemented to build an interactive web application that could serve as a bridge between patients and real doctors.**

Keywords—natural language processing, machine learning, classification

I. Introduction

With more and more people seeking online medical service during the Covid-19 pandemic, the adoption of natural language processing and machine learning techniques have become a rising interest in the healthcare industry. Our project aims to help with the automation of the initial classification of medical symptoms, which will greatly save time in matching patients with the correct doctors. Thus, this is a text classification project that finds the best combination of textual feature extraction methods and multi-class classification algorithms that can be used to classify the ailments based on some phrases the patients entered.

II. Motivation

During the Covid-19 pandemic, online medical service platforms gained widespread attention. People often face the following problems: long waiting times for doctors in hospitals; worrying about cross-infection and not wanting to go to the hospital. Or patients only feel slightly unwell, but would like to get professional advice. As a result, online medical service consultation has great customer demand.

For some patients with periodic medical demands or simple medical needs, remote medical services can reduce he number of patient visits and help improve the efficiency of diagnosis and treatment. This also enables effective and efficient allocation of medical resources to patients with severe health issues, who need immediate or long-term medical attention. Patients with minor symptoms can consult online anytime anywhere without taking sick leaves to the hospital. It is particularly convenient for busy employees. At the same time, the convenient online medical service is especially suitable for some patients with limited mobility or transportation. All patients can access this online medical service anytime anywhere.

Online medical services can first help the hospital to further divert patients, and the patients can seek medical treatment offline more accurately. Automation of natural language processing in online medical services can interpret patients' long sentences and extract important textual features, identify symptoms and categorize them into associating medical fields. The application of natural language processing in online medical services can not only reduce time spent by doctors and patients to discuss, document, and diagnose basic symptoms, but also improve the accuracy of basic symptoms diagnosis by avoiding potential human errors.

III. Literature survey

We have done some research in the following three main categories: existing works and potential application in the scope of healthcare, overview elements in text classification and discover text mining technologies, and lastly, evaluate performance among different classification and feature extraction methods.

A. Selecting existing works and potential applications in the scope of healthcare

Chaitrashree, K.M. and his colleagues [1] in their paper "Unstructured Medical Text Classification using Machine

Learning and Deep Learning Approaches" have used machine learning techniques to build a classification system based on patients' disease descriptions. They use libraries and tools to handle original data cleaning, then feature extraction with the Bag-Of-Words technique. They train the model using 4 different algorithms, then compare the models' accuracy to choose the best model. They have also developed an app called MedAid for those who want to self-diagnose their illness. The resulting model is accurate but needs more performance metrics to further investigate the difference between models grouped with different feature extraction techniques. Pendyala, V. S. and his colleagues [2] illustrated how machine learning can help some of the healthcare functions such as "diagnosis for mass deployment." which is the exact purpose behind our project. In his paper, the author confirmed the feasibility of the text-mining approach to the medical diagnosis problem. Besides the solution of using machine learning-based feature extraction and classification, he had also applied first-order-logic-based systems for reasoning and pointed out that the future of this sort of study should be using the Hadoop ecosystem to improvise the results.

B. Overview elements in text classification and discover text mining technologies

Mirończuk, M. M. [3] claims in his paper that "there are various categories of text classification which are domain, classification purpose, classification task, general approach, and dedicated approach." . Mining techniques are important and each has its own advantages and drawbacks. We will make a table of techniques for comparison in the final report.

C. Performance among different classification and feature extraction methods

Khursheed, M. S. [4] in his paper used an automated process to examine the accuracy, precision, recall, and f-measure in all combinations of feature selection, term weighting, and classification algorithms on the large and diverse Arabic text dataset including poems and forums. The best model in his case turned out to be the SVM achieved using "TF as the base for term selection, GSS as the term selection method, and LTC as the representation scheme." Since our case has a much smaller size and simpler structure, the expected accuracy in our case should be higher than 95%. Gang Kou [5] claimed that the evaluation of methods for text classification with a small dataset should consider not only accuracy but also stability and efficiency. They applied MCDM (multiple criteria decision making) based methods to evaluate 10 feature selection methods and made recommendations to the ranked 5 methods.

IV. Methodology

Based on our literature survey of current machine learning applications on text classification in the healthcare industry, we learned many techniques, approaches, and algorithms that have been used to extract and classify information from text data. Our experiment will mainly focus on the two most important steps, feature extraction methods, and machine learning algorithms. We will experiment with different combinations of feature extraction methods and machine learning algorithms, compare model performance, and provide insight into which combination is the best suited for our project. The complete modeling process is shown in the flowchart below.

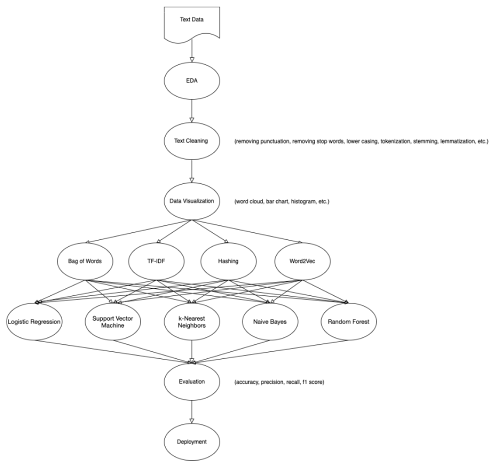

Text Data
EDA
Text Cleaning
(removing punctuation, removing stop words, lower casing, tokenization, stemming, lemmatization, etc.)
Data Visualization
(word cloud, bar chart, histogram, etc.)
Bag of Words
TF-IDF
Hashing
Word2Vec
Logistic Regression
Support Vector Machine
k-Nearest Neighbors
Naive Bayes
Random Forest
Evaluation
(accuracy, precision, recall, f1 score)
Deployment

Fig. 1. Flow chart for project

V. Data introduction

We decided to use a medical dataset which consists of 6,661 entries, 13 columns, and generated by different contributors describing their medical symptoms. The entries have a text phrase describing the symptom and the text phrases are labeled with ailments the text phrases correspond to. The text documents are labeled with 25 different ailments.

VI. EDA

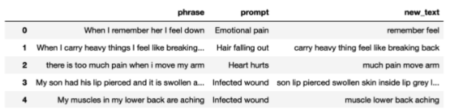
To better understand the data, we loaded the data (.csv file) into a pandas data frame and used pandas' shape(), and describe() methods to get the general information of the dataset. The dataset has 6661 entries, each entry has 13 columns. The 'phrase' column is the text phrase describing the symptoms, the 'prompt' column is the ailment the phrase is labeled to which includes 25 different ailments: Acne, Back pain, Blurry vision, Body feels weak, Cough, Earache, Emotional pain, Feeling cold, Feeling dizzy, Foot ache, Hair falling out, Hard to breathe, Headache, Heart hurts, Infected wound, Injury from sports, Internal pain, Joint pain, Knee pain, Muscle pain, Neck pain, Open wound, Shoulder pain, Skin issue, Stomach ache. There are also some other columns such as 'speaker-id', 'writer-id', 'background-noise-audiable' which are not useful for our project. We removed them from the data frame, so the final data frame only includes 2 columns: 'phrase' and 'prompt' and 6661 entries. We checked missing data and duplicated data, and none were found.

VII. Text cleaning

After the EDA with the original text data we collected, we cleaned the text by conducting tokenization, lowercasing, removing punctuation and stop words, and lemmatization.

To start the process, we tokenize the text by dividing the sentences into separated words using the word-tokenize function from Natural Language Toolkit (nltk) library. As upper cases and punctuations were also not needed in the NLP processing, all texts were converted to lower cases and punctuations were removed. English stop words such as between, them, this, an, are, and etc, did not add significant meaning to the sentence, and therefore were removed from the text as well. In English words, different tenses and forms of words essentially refer to the same meaning. As such, to finalize the text cleaning process, we have grouped different forms of words to represent the same word in the lemmatization process by using the WordNetLemmatizer function from nltk library.

The figure below shows the comparison of the original text in the ‘phrase’ column and the cleaned text in the ‘new-text’ column.



	phrase	prompt	new_text
0	When I remember her I feel down	Emotional pain	remember feel
1	When I carry heavy things I feel like breaking...	Hair falling out	carry heavy thing feel like breaking back
2	there is too much pain when i move my arm	Heart hurts	much pain move arm
3	My son had his lip pierced and it is swollen a...	Infected wound	son lip pierced swollen skin inside lip grey l...
4	My muscles in my lower back are aching	Infected wound	muscle lower back aching

Fig. 2. Example of text cleaning

VIII. Visualization

Based on the cleaned data, we used libraries like matplotlib and word cloud in python to make some visualizations and explore the data. We made a bar chart showing the count of documents for each symptom ‘prompt’ class (Figure 3). From the plot, we can see that, out of the 25 classes, ‘Acne’ has the highest label count of 328 and ‘Open Wound’ has the lowest label count of 208. A line chart showing the distribution of phrase length (Figure 4) is made, which shows that most phrases have a length around 40. Figure 5 shows a word cloud showing the word frequency in all the phrases with ‘pain’ being the most frequently appeared word. Figure 6 shows the distribution of the mean word length in the phrases, and we can see that 61.3% of the phrases have a mean word length of 4.

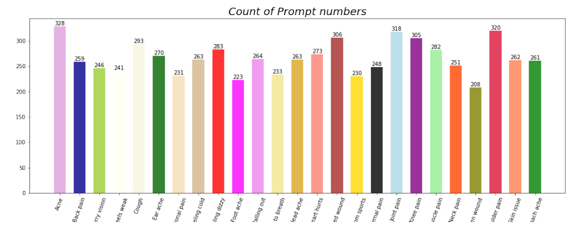


Fig. 3. Prompt number Count

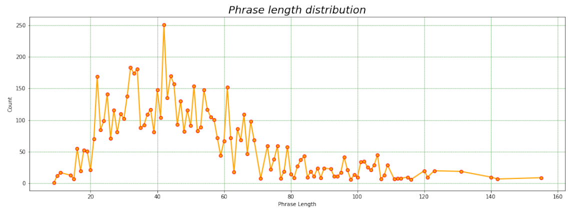


Fig. 4. Phrase Length Distribution

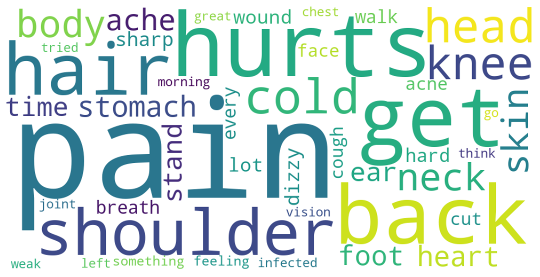


Fig. 5. Word Cloud

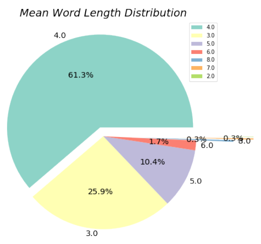
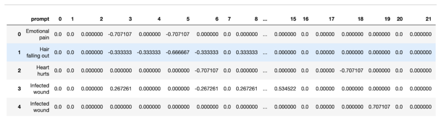


Fig. 6. Mean Word Length Distribution

IX. Feature extraction

After initial text cleaning, the text needs to be transformed in a format that can be understood by machine learning algorithms. This feature extraction process is also called vectorization. We tried four different methods, Bag of Words, TF-IDF, Word2Vec, and hashing. Specifically, Bag of Words [6] represents the text based on the count of words in the document. And TF-IDF [7] evaluates how important a word is to a document in a collection of documents. Hashing [8] converts a hash of words to a matrix of token occurrences. On the other hand, Word2Vec [9] represents each word as a vector in a multidimensional vector space where words with similar meanings are closer to each other. In Python, we used TfidfVectorizer, HashingVectorizer, CountVectorizer from sklearn library, and Word2Vec from gensim library. Each method returned a data frame of the same number of rows and a different number of columns based on the parameters we specified. Both Bag of Words and TF-IDF transform the original text into a feature space of 953 columns. Features extracted using hashing have 75 columns. For Word2Vec, since it returns a vector representation for each word. We need to extend these word vectors and generate vectors on document level, so we computed an average of all the words in the document, and returned a feature space of 100 columns. Below is an example of a converted data frame sample using the hashing vectorizer.



	prompt	0	1	2	3	4	5	6	7	8	...	15	16	17	18	19	20	21
0	Emotional pain	0.0	0.0	0.000000	-0.707107	0.000000	-0.707107	0.000000	0.0	0.000000	...	0.000000	0.0	0.00000	0.000000	0.000000	0.0	0.000000
1	Hair falling out	0.0	0.0	0.000000	-0.333333	-0.333333	-0.666667	-0.333333	0.0	0.333333	...	0.000000	0.0	0.00000	0.000000	0.000000	0.0	0.000000
2	Heart hurts	0.0	0.0	0.000000	0.000000	0.000000	0.000000	-0.707107	0.0	0.000000	...	0.000000	0.0	0.00000	-0.707107	0.000000	0.0	0.000000
3	Infected wound	0.0	0.0	0.000000	0.267261	0.000000	0.000000	-0.267261	0.0	0.267261	...	0.534522	0.0	0.00000	0.000000	0.000000	0.0	0.000000
4	Infected wound	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	...	0.000000	0.0	0.00000	0.000000	0.707107	0.0	0.000000

Fig. 7. Example of feature extraction

X. Feature transformation

We have 4 subsets with 4 types of feature extraction methods. The Bag of Words subset is one of the countvectorizer which contains only the binary features indicating whether the word exists. The TF-IDF, Hashing Vectorizer, and Word2Vec subsets are vectorized representations of the importance of words with respect to their frequency and counts. In addition, they are all in the range of -1 to +1. Therefore, there is no need to standardize those subsets. However, TF-IDF and Bag of Words subsets contain over 900 features and most of them are sparse. The sparse datasets will downgrade the performance of the models. We decided to use Principal Components Analysis to narrow down the number of features for training, making the result much more interpretable. Principal Component Analysis, or PCA, [10] is a dimensionality reduction process that reduces the less important and less correlated components or features, only keeping the most important and correlated features for modeling. In our project, we use the PCA function from the decomposition package in the scikit-learn library. To avoid missing the important information from feature reduction, we applied PCA methods to get reduced datasets with appropriate PCA components that include 99% variance. As a result, 489 features, 446 features, 6 features, and 92 features were disregarded in the Bag of Words subset, TF-IDF subset, Hash vectorizer subset, and Word2Vec subset respectively.

TABLE I
Table of Feature Transformation Summary

Subsets	# of Features	# of Reduced Features
Bag of Words	953	464
TF-IDF	953	507
Hash Vectorizer	75	69
Word2Vec	100	8

XI. Data split

Training machine learning models require learning from lots of input data, so the majority of the data will be used for training. The model evaluation must be performed on the test dataset, which has not been used for training. Therefore, we will keep the test dataset completely locked before models are ready to be evaluated to avoid the issue of data leakage. In our project, we used the train_test_split function from the model_selection package in the scikit-learn [11] library to split four transformed datasets into 8 training and test subsets. The proportion of them is 80% and 20% respectively. We stratify the split based on the label to ensure all classes are in the same distribution in each subset.

XII. Model

Random Forest [12] is an ensemble model that consists of a large number of individual decision trees. It combines the predictions of many decision trees and predicts the class with the most votes from those trees. The important key is that those individual trees are trained on random subsets of features and samples. Therefore the trees are independent of each other and protect each other from their individual errors. As a prerequisite, the data input should be preprocessed so that their errors can be least correlated. The diversity of decision trees as a group allowed it to outperform the single decision tree model. In our project, we use the Random Forest Classifier function from the ensemble package in the scikit-learn library to first build 4 classifiers with four sets of feature datasets. The initial setting is 100 estimators which means there are 100 individual decision trees in one Random Forest. After that, we further tune the hyperparameters including the impurity criteria choice of 'Gini' and 'Entropy', maximum depth of individual trees, minimum samples allowed in one node of the tree, and the number of individual trees in the Random Forest. We find out that the best Random Forest Classifier in our case is the setting that the impurity criteria is 'Gini', the maximum depth is 13, the minimum samples allowed is 1, and the number of individual trees is 50.

Logistic Regression [13] is a probabilistic model and discriminative classifier. It uses a predetermined conditional probability function which is based on the assumption of a linear relationship among the features and returns a probability after applying the sigmoid activation function to transform linear predictions($-\infty$, $+\infty$) to probabilities [0,1]. Therefore, we have a decision boundary to separate the class based on the threshold value we believe and normally it is

0.5. Since the sigmoid function is nonlinear, the model is more smooth and robust to outliers. In our project, we use the Logistic Regression function from the linear model package in the scikit-learn library to build 4 classifiers with four sets of feature datasets. Logistic Regression is not an inherently multi-class classifier. The default setting will use One Vs Rest methods to build 25 classifiers in which each classifier is trained to distinguish the specified class and all others.

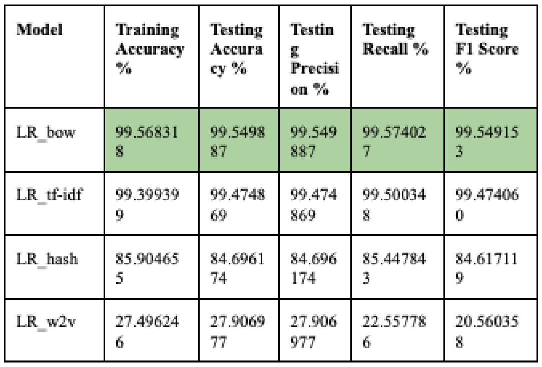
Support Vector Classifier [14] is a supervised learning model used for classification. The idea behind it is to search for an optimal hyperplane that maximizes the margins between data points from different classes. It is also called the margin maximization classifier. The hyperplane is determined by the points closest to the hyperplane as known as support vectors. Therefore, it can handle nonlinear data, mixed variables, and missing data. Furthermore, it is also efficient for high-dimensional data sets. Since some of our datasets after preprocessing contain hundreds of sparse data points, it can be a good model choice. In our project, we use the SVC function from the SVM package in the scikit-learn library to build 4 classifiers with four sets of feature datasets. Similar to Logistic Regression, Support Vector Classifier is also not an inherently multi-class classifier. The default setting will use One Vs Rest methods to build 25 classifiers in which each classifier is trained to distinguish the specified class and all others.

The Naive Bayes Classifier [15] is a generative model that finds the joint probability of distribution of data to derive the actual probability of class given data features. Since finding the real conditional probability is intractable for data in high dimensions, it simplifies the computing based on the assumption that the features are conditionally independent. It is faster than discriminative classifiers and inherently supports multi-class prediction compared to SVC and Logistic Regression. This Bayesian approach needs a significant amount of data and several strong assumptions to get accurate likelihood probabilities instead. In our project, we use the GaussianNB function from the naive_bayes package in the scikit-learn library to build 4 classifiers with four sets of feature datasets. However, GaussianNB assumes that features are continuous and follow the Gaussian distribution. Furthermore, the assumption of conditional independence of features is not realistic for text-vectorized datasets. Therefore, the real performance of this classifier may not be taken seriously.

K Nearest Neighbor [15] is a simple but lazy learning classifier that doesn't compute any model until test data arrives. It's memory-based learning as it stores feature vectors of training data points in the memory. When it receives unclassified data, it measures the distance such as Euclidean, Manhattan, and Minkowski from the new data to all other data that is already classified. After that, it generates K nearest neighbors and predicts the class of unseen data based on the class of neighbors that appears the most time. Since it is a distance-based method, all data should be normalized. The biggest advantage is it can handle highly nonlinear decision boundaries and doesn't assume the decision boundary's shape. In our project, we use the KNeighborsClassifier function from the neighbor's package in the scikit-learn library to first build 4 classifiers with four sets of feature datasets. The initial setting is k = 3 which means it will generate 3 nearest neighbors for the test data points. After that, we further tune the hyperparameters including the choice of distance metrics including 'euclidean' and 'manhattan', the number of K neighbors, and the choice of weight function for K neighbors including 'uniform' in which All points in each neighborhood are weighted equally and 'distance' in which weight points by the inverse of their distance. We find out the best K Nearest Neighbor Classifier in our case is the setting that distance metric is 'manhattan', K = 3, weight is 'uniform'.

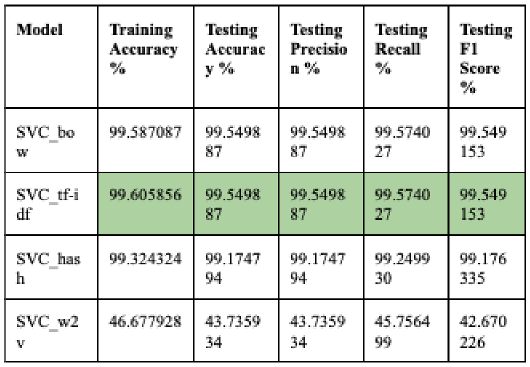
XIII. Evaluation

We have 5 machine learning algorithms with 4 different sets of features. In total, we have at least 20 models to evaluate which are hard to interpret in one step. Therefore, we first compare 4 models at once and find out which set of features is best suited for each machine learning algorithm in advance. Then we compare 5 combinations of machine learning algorithms with the best set of features and find out the winner. After that, we further tune the hyperparameter for the winner and find out our best model. The performance of the model is determined by 5 metrics: training accuracy, testing accuracy, testing precision score, testing recall score, and testing F1 score. In our project, we use accuracy_score, precision_score, recall_score, and f1_score function from the metrics package in scikit-learn library to generate those scores. Since our project is a multi-class problem and there is a precision, recall, and F1 score for each class, we use the parameter of 'weight' to calculate metrics for each label, and find their average weighted by the number of true instances for each label as their aggregated precision, recall, and F1 Scores. Training accuracy and testing accuracy are defined as the number of classes the model correctly predicts divided by the total number of the predictions made for the training dataset and testing dataset respectively. The testing precision score is the performance measurement for the model not to predict positive when the sample is true negative in the testing dataset. The testing recall score is the performance measurement for the model to correctly identify True Positives. F-beta score is the weighted harmonic mean of precision and recall. We choose beta to be 1 so that this score gives equal importance to recall and precision. As our prediction will not be used for real diagnostic purposes, we believe equal importance to recall and precision is the plausible choice. The best score of these four metrics will be highlighted in green color in the metric table.



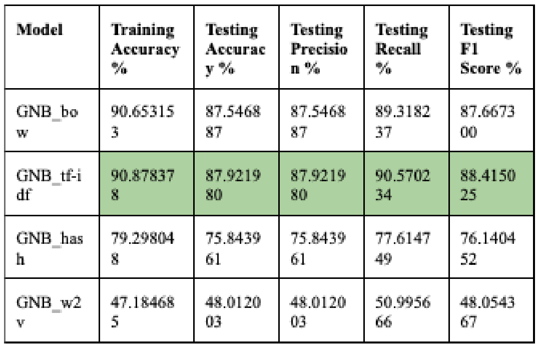
Model	Training Accuracy %	Testing Accuracy %	Testing Precision %	Testing Recall %	Testing F1 Score %
LR_bow	99.568318	99.549887	99.549887	99.574027	99.549153
LR_tf-idf	99.399399	99.474869	99.474869	99.500348	99.474060
LR_hash	85.904655	84.696174	84.696174	85.447843	84.617119
LR_w2v	27.496246	27.906977	27.906977	22.557786	20.560358

Fig. 8. Metric Table for Logistic Regression



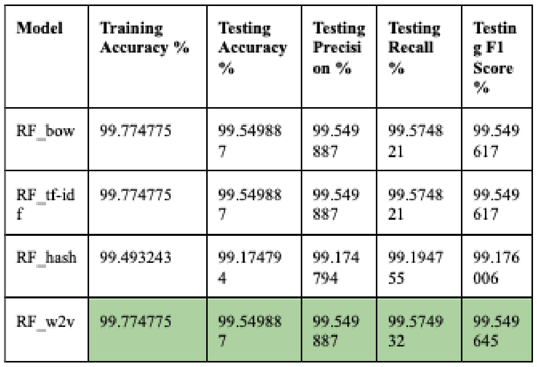
Model	Training Accuracy %	Testing Accuracy %	Testing Precision %	Testing Recall %	Testing F1 Score %
SVC_bow	99.587087	99.549887	99.549887	99.574027	99.549153
SVC_tf-idf	99.605856	99.549887	99.549887	99.574027	99.549153
SVC_hash	99.324324	99.174794	99.174794	99.249930	99.176335
SVC_w2v	46.677928	43.735934	43.735934	45.756499	42.670226

Fig. 9. Metric Table for Support Vector Classifier



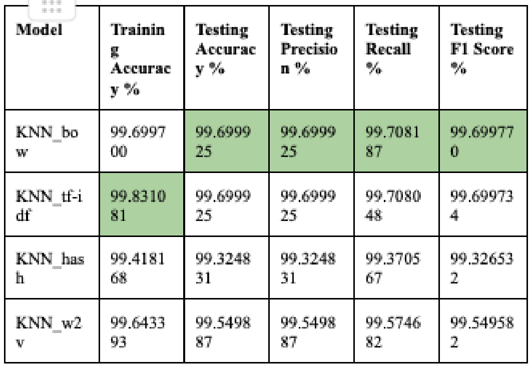
Model	Training Accuracy %	Testing Accuracy %	Testing Precision %	Testing Recall %	Testing F1 Score %
GNB_bow	90.653153	87.546887	87.546887	89.318237	87.667300
GNB_tf-idf	90.878378	87.921980	87.921980	90.570234	88.415025
GNB_hash	79.298048	75.843961	75.843961	77.614749	76.140452
GNB_w2v	47.184685	48.012003	48.012003	50.995666	48.054367

Fig. 10. Metric Table for Gauissian Naive Bayes



Model	Training Accuracy %	Testing Accuracy %	Testing Precision %	Testing Recall %	Testing F1 Score %
RF_bow	99.774775	99.549887	99.549887	99.574821	99.549617
RF_tf-idf	99.774775	99.549887	99.549887	99.574821	99.549617
RF_hash	99.493243	99.174794	99.174794	99.194755	99.176006
RF_w2v	99.774775	99.549887	99.549887	99.574932	99.549645

Fig. 11. Metric Table for Random Forest



Model	Training Accuracy %	Testing Accuracy %	Testing Precision %	Testing Recall %	Testing F1 Score %
KNN_bow	99.699700	99.699925	99.699925	99.708187	99.699770
KNN_tf-idf	99.831081	99.699925	99.699925	99.708048	99.699734
KNN_hash	99.418168	99.324831	99.324831	99.370567	99.326532
KNN_w2v	99.643393	99.549887	99.549887	99.574682	99.549582

Fig. 12. Metric Table for K Nearest Neighbors

As a result, Logistic Regression with bag-of-words extracted data is the best among all LR Classifiers. One of the surprising findings is that the Logistic Regression with Word2Vec extracted data is better than 4 % from random guessing since we have 25 classes but still drastically downgraded the performance. Support Vectors Classifier and Gaussian Naive Bayes with tf_idf extracted data are the best among all SVCs and all Naive Bayes Classifiers. However, they have the same issue as Logistic Regression with Word2Vec extracted data and perform significantly worse than those with other features data. All dataset has been preprocessed with the same step and extracted PCA components based on 99% variance. The only difference is that the Word2Vec dataset contains only 8 features. We assume the reason behind is that those models need a significant amount of data in high dimensions to get either plausible decision boundaries or accurate likelihood probabilities. Random Forest with word2vec extracted data is the best among all Random Forest Classifiers and K Nearest Neighbors with bag-of-words extracted data is the best among all KNNs. They both have a high score in every metric and are consistent with all kinds of feature datasets.

Next, we decide to further tune the hyperparameters for both classifiers and determine the final winner as our best model choice. The Tuning process has been discussed in the model section. After improvement, We find out the best Random Forest Classifier in our case is the setting that the impurity criteria is ‘Gini’, the maximum depth is 13, the minimum samples allowed is 1, and the number of individual trees is 50 and the best K Nearest Neighbor Classifier in our case is the setting that distance metric is ‘manhattan’, K = 3, weight is ‘uniform’. In conclusion, an improved Random Forest with Word2Vec feature dataset is the model of choice for our project since it gets the highest score on every metric. The receiver operating characteristic curve and Precision-Recall Curve are shown below. ROC curve is a graphical plot that illustrates the diagnostic ability and the precision-recall curve shows the tradeoff between precision and recall. Both curves present the performance of a binary classification for different thresholds. Therefore,there are 25 individual curves for each graph. Our model of choice performs extremely well for every label classification.

Model	Training Accuracy %	Testing Accuracy %	Testing Precision %	Testing Recall %	Testing F1 Score %
Tuned KNN	99.699700	99.699925	99.699925	99.708048	99.699734
Tuned RF	99.831081	99.699925	99.699925	99.708048	99.699734

Fig. 13. Metric Table to determine the best model

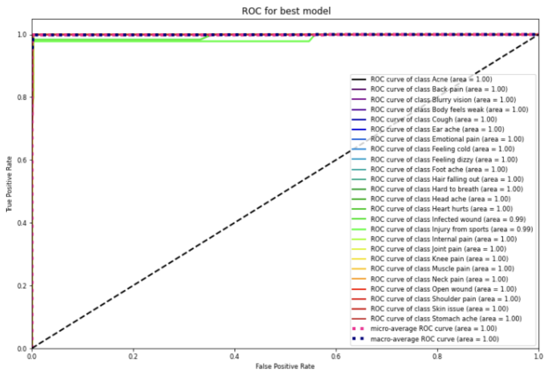


Fig. 14. ROC for best mode

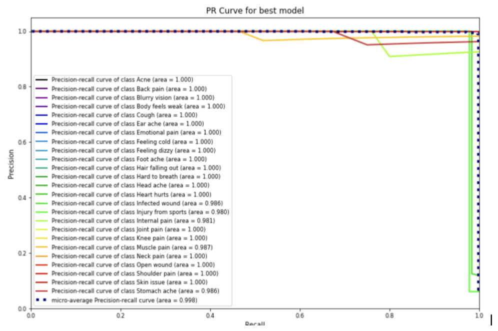


Fig. 15. PR Curve for best model

XIV. Final model deployment

The goal of deployment is to take a trained ML model and make its predictions available to regular users. In our use case, we don't expect regular users such as assistants from the hospital, or patients on the internet to know the process of cleansing and transfer the symptom description text into the design format of input data to feed in our model. We need to build an automated pipeline that preprocesses the text input and returns the prediction that can be easily read by users. Due to our limited capacity, our models for prediction and data preprocessing are static and immutable deployment. Pickle [16] in Python is an object serialization module that can be used for serializing and deserializing objects. We use pickles to save and reload the models without re-training. Three models have been serialized in our deployment. They are the Word2Vec model for tokenizing the text input, PCA model for transforming Word2Vec features, and machine learning model of our choice for predicting the class. The full workflow of deployment includes cleansing input text, tokenizing the text into smaller units as features, transforming features into PCA components to reduce correlation and sparsity, feeding into pre-trained model to get prediction, returning prediction to users visually. To better serve the regular users, we create a simple interface as a web portal to allow users to type in their symptom description text and to get response at the real time. The webpage is created by the Flask [17] function in Python with the help of css and html knowledge. The following diagrams illustrate our deployment workflow and example of web demo.

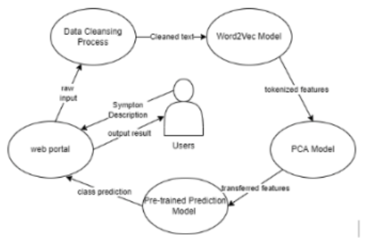


Fig. 16. Workflow diagram for deployment

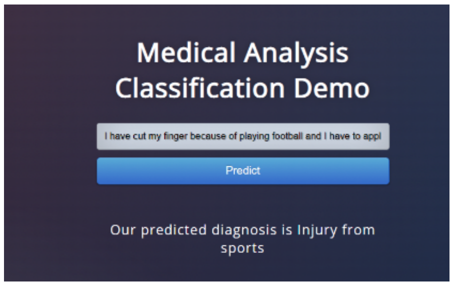

Medical Analysis Classification Demo
I have cut my finger because of playing football and I have to appl
Predict
Our predicted diagnosis is Injury from sports

Fig. 17. Workflow diagram for deployment

XV. Technical difficulty

First, we studied various methods to conduct text cleaning and decided to use functions from the nltk library to conduct tokenization, lowercasing, removing punctuation and stop words, and lemmatization. Second, in order to convert text to numerical format, we summarized different feature extraction methods that were covered in the literature survey, and decided to use four methods, Bag of Words, TF-IDF, Word2Vec, and Hashing. We also implemented a function that extends the word vectors to document level for Word2Vec. Third, since we have over 20 different combinations of machine learning models and feature extraction methods, we compare the model performance in two steps, comparing within each machine learning algorithm and comparing the winner across different machine learning algorithms. Fourth, in order to deploy our final model, we utilized pickle in Python to save and load the trained model.

XVI. Innovation

Most works in the similar medical symptoms text classification fields are conducting comparison between different machine learning models or comparison between different feature extraction methods. However, our project focused on finding the best combination of machine learning models and feature extraction methods. And we found that the best combination is the tuned random forest classifier using the Word2Vec extracted features. Also, our project introduced PCA to help reduce the feature dimensions, which was found to be very important in our project but is not covered in any other related works.

XVII. Significance to the real world

Despite online medical services starting to develop a few years ago, the Covid-19 pandemic significantly skyrockets the demand of online medical services and expedites the development process. The medical symptoms text classification we designed can be easily applied to today's clinics, hospitals, or anywhere that provides medical services, assisting the medical service institution to identify the patients' symptoms based on text messages, classifying accordingly and transfer to the medical departments with high accuracy and high efficiency. This medical symptoms text classification system is particularly useful and significant nowadays to reduce direct human contact which reduces the chance of spreading highly infectious diseases like Covid-19. It also helps reduce the administrative time and money spent on basic symptoms classification, releasing more medical resources to serve more patients. This classification system can not only be used on the online chat box, it can also be applied to auto-reply text messages sent to the phone.

In conclusion, the medical symptoms text classification system we designed is easily applicable in various medical institutions, adding significant and meaningful life saving and business values to the real world.

XVIII. Prospects of winning competition / publication

One of the strengths of our project is that we experimented with different combinations of machine learning models and feature extraction methods, which would offer a new point of view to the field of medical symptoms text classification. Moreover, our final model accuracy of 99.69% would be very impressive in competitions or publications.

XIX. Lessons learned

First, we learned that features engineering is extremely important to the performance of a model. One of the shocking findings is that the Logistic Regression, Support Vectors Classifier, and Gaussian Naive Bayes have a similar issue with Word2Vec extracted data. They perform significantly worse than the same machine learning algorithms with other feature data. However, tuned random forest classifiers using the Word2Vec extracted features are the best among all other models and become our model of choice for deployment. Therefore, we should tune the model with different hyperparameters and feature sets in the future work.

Second, accuracy is not an adequate metric when the performance of models is nearly perfect, especially when distribution of the label classes is extremely imbalanced. In our project, logistic regression classifiers and Support Vector Classifiers are using One VS Rest method to deal with multi-class classification and both have good accuracy but their precision scores are slightly smaller. There is a warning from jupyter notebook that shows there are certain classes that the classifiers never predict. The accuracy is still high as those classes contribute extremely small amounts in all data. But that's not the result we like to see in real life applications. In the future work, we should deal with data imbalance issues with data augmentation for minority classes or data reduction for majority classes. In addition, we should consider all metrics including accuracy, precision and recall scores to evaluate the model thoroughly.

Third, when we test our deployment, we find that some input text can result in an unexpected prediction. We analyze the problem and find out the issue exists because our dataset contains 6661 rows for 25 classes of disease which means support rows for each disease is not sufficient. Combining the issue of data imbalance, the model may not fully understand the dependency of features and disease. In the future work, we should either collect more data or aggregate some classes together to make the prediction more realistic.

Last, we learn how to deploy our model into an interactive website with the help of a pickle function to save and load pre-trained models. In the future work, we should consider the dynamic model deployment in which the model can be further improved by training with incoming text so that we can solve the issue of lack of data we discovered before.

XX. Conclusion

In this project, we investigate the performance of 22 models that comprise 5 machine learning algorithms with 4 nature language process techniques to find the best set of combinations. The best model of choice is used to build an interactive web application to solve the rising demand for online medical services. Under the fact that there are 5.4 billion internet users and a pandemic is raging over the world, more and more people are seeking help from online medical services. Our application not only enables patients to early identify the ailment and take preventative actions in advance but also serves as a bridge between patients and real doctors by saving time in matching patients. Our model of choice is the tuned random forest classifier using the Word2Vec extracted features. It gives the 99.69% accuracy, 99.69% precision, and 99.71% recall scores in the task of diagnosis classification. It is the model with the best performance compared to the other 21 models.

XXI. Team members and corresponding roles

Yijia Li: Conceptualization, Methodology, Investigation, Writing- Original draft preparation. Lilin Huang: Data curation, Methodology, Investigation, Writing- Original draft preparation. Wanyu Huang: Visualization, Methodology, Investigation, Writing- Original draft preparation. Haiyan An: Supervision, Methodology, Writing- Original draft preparation, Writing- Reviewing and Editing. Zhe Li: Software, Validation, Methodology, Writing- Original draft preparation.

References

[1] K. M. Chaitrashree, T. N. Sneha, S. R. Tanushree, G. R. Usha, and T. C. Pramod, "Unstructured medical text classification using machine learning and Deep Learning Approaches," 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), 2021.

[2] Mirończuk, M. M., & Protasiewicz, J. (2018). A recent overview of the state-of-the-art elements of text classification. Expert Systems with Applications, 106, 36–54. https://doi.org/10.1016/j.eswa.2018.03.058

[3] Pendyala, V. S., Yi Fang, Holliday, J., & Zalzala, A. (2014). A text mining approach to automated healthcare for the masses. IEEE Global Humanitarian Technology Conference (GHTC 2014). https://doi.org/10.1109/ghtc.2014.6970257

[4] Khorsheed, M. S., & Al-Thubaity, A. O. (2013). Comparative evaluation of text classification techniques using a large diverse Arabic dataset. Language Resources and Evaluation, 47(2), 513–538. https://doi.org/10.1007/s10579-013-9221-8

[5] Kou, G., Yang, P., Peng, Y., Xiao, F., Chen, Y., & Alsaadi, F. E. (2020). Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods. Applied Soft Computing, 86, 105836. https://doi.org/10.1016/j.asoc.2019.105836

[6] Jiang, H., Yang, D., Xiao, Y., & Wang, W. (2020). Understanding a bag of words by conceptual labeling with prior weights. World Wide Web, 23(4), 2429–2447. https://doi.org/10.1007/s11280-020-00806-x

[7] G., K., & A., S. (2020). Survey on Research Paper Classification based on TF-IDF and Stemming Technique using Classification Algorithm. International Journal of Computer Applications, 176(25), 23–27. https://doi.org/10.5120/ijca2020920248

[8] Y. Kanada, "A vectorization technique of hashing and its application to several sorting algorithms," Proceedings. PARBASE-90: International Conference on Databases, Parallel Architectures, and Their Applications.

[9] Ant, K., Sogukpinar, U., & Amasyali, M. F. (2018). Comparison of Templates with Word2Vec in Finding Semantic Relations Between Words. Journal of Intelligent Systems with Applications, 13–17. https://doi.org/10.54856/jiswa.201805007

[10] "Learn: Machine learning in python - scikit-learn 0.16.1 documentation," scikit. [Online]. Available: https://scikit-learn.org/. [Accessed: 07-May-2022].

[11] M. Kaur and M. Bansal, "Text classification using clustering techniques and P.C.A.," 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), 2016.

[12] Y. Sun, Y. Li, Q. Zeng, and Y. Bian, "Application research of text classification based on Random Forest algorithm," 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), 2020.

[13] T. Pranckevicius and V. Marcinkevicius, "Application of logistic regression with part-of-the-speech tagging for multi-class text classification," 2016 IEEE 4th Workshop on Advances in Information, Electronic and Electrical Engineering (AIEEE), 2016.

[14] M. Goudjil, M. Koudil, M. Bedda, and N. Ghoggali, "A novel active learning method using SVM for text classification," International Journal of Automation and Computing, vol. 15, no. 3, pp. 290–298, 2016.

[15] B. Y. Pratama and R. Sarno, "Personality classification based on Twitter text using naive Bayes, KNN and SVM," 2015 International Conference on Data and Software Engineering (ICoDSE), 2015.

[16] "Quick hacks to save machine learning model using pickle," Analytics Vidhya, 18-Aug-2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/08/quick-hacks-to-save-machine-learning-model-using-pickle-and-joblib/. [Accessed: 07-May-2022].

[17] "What is Flask Python," What is Flask Python - Python Tutorial. [Online]. Available: https://pythonbasics.org/what-is-flask-python/. [Accessed: 07-May-2022].

Appendices

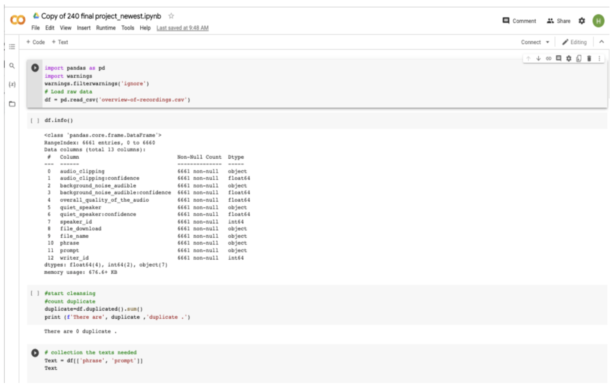
IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.

A. Medical Symptoms Text Classification Github:

https://github.com/sherryahy/Data-245-Medical-Symptoms-Text-Classification

B. Pair programming:

https://colab.research.google.com/drive/1QEAF6n2gc7E_pTcF5ot4-JMEYUMJG_58


Copy of 240 final project_newest.ipynb
File Edit View Insert Runtime Tools Help Last saved at 9:48 AM
Comment Share
+ Code + Text
Connect
Editing

```python
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
# Load raw data
df = pd.read_csv('overview-of-recordings.csv')
```

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6661 entries, 0 to 6660
Data columns (total 13 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   audio_clipping                    6661 non-null   object
 1   audio_clipping:confidence         6661 non-null   float64
 2   background_noise_audible          6661 non-null   object
 3   background_noise_audible:confidence 6661 non-null   float64
 4   overall_quality_of_the_audio      6661 non-null   float64
 5   quiet_speaker                     6661 non-null   object
 6   quiet_speaker:confidence          6661 non-null   float64
 7   speaker_id                        6661 non-null   int64
 8   file_download                     6661 non-null   object
 9   file_name                         6661 non-null   object
 10  phrase                            6661 non-null   object
 11  prompt                            6661 non-null   object
 12  writer_id                         6661 non-null   int64
dtypes: float64(4), int64(2), object(7)
memory usage: 676.6+ KB
```

```python
#start cleansing
#count duplicate
duplicate=df.duplicated().sum()
print (f'There are', duplicate ,'duplicate .')
```

```
There are 0 duplicate .
```

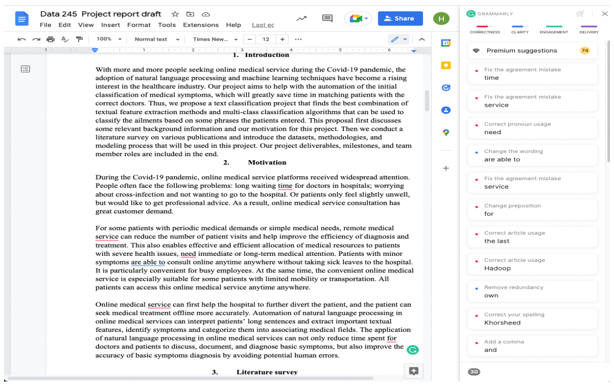
```python
# collection the texts needed
Text = df[['phrase', 'prompt']]
Text
```

Fig. 18. Used Google Colab for Pair programming

C. Trello project management:

https://trello.com/b/4UgCAmSN/data-245-team-project-management

D. Grammarly


Data 245 Project report draft
File Edit View Insert Format Tools Extensions Help Last ec
Share
100% Normal text Times New... 12
GRAMMARLY
CORRECTNESS CLARITY ENGAGEMENT DELIVERY
Premium suggestions 74
30

1. Introduction

With more and more people seeking online medical service during the Covid-19 pandemic, the adoption of natural language processing and machine learning techniques have become a rising interest in the healthcare industry. Our project aims to help with the automation of the initial classification of medical symptoms, which will greatly save time in matching patients with the correct doctors. Thus, we propose a text classification project that finds the best combination of textual feature extraction methods and multi-class classification algorithms that can be used to classify the ailments based on some phrases the patients entered. This proposal first discusses some relevant background information and our motivation for this project. Then we conduct a literature survey on various publications and introduce the datasets, methodologies, and modeling process that will be used in this project. Our project deliverables, milestones, and team member roles are included in the end.

2. Motivation

During the Covid-19 pandemic, online medical service platforms received widespread attention. People often face the following problems: long waiting time for doctors in hospitals; worrying about cross-infection and not wanting to go to the hospital. Or patients only feel slightly unwell, but would like to get professional advice. As a result, online medical service consultation has great customer demand.

For some patients with periodic medical demands or simple medical needs, remote medical service can reduce the number of patient visits and help improve the efficiency of diagnosis and treatment. This also enables effective and efficient allocation of medical resources to patients with severe health issues, need immediate or long-term medical attention. Patients with minor symptoms are able to consult online anytime anywhere without taking sick leaves to the hospital. It is particularly convenient for busy employees. At the same time, the convenient online medical service is especially suitable for some patients with limited mobility or transportation. All patients can access this online medical service anytime anywhere.

Online medical service can first help the hospital to further divert the patient, and the patient can seek medical treatment offline more accurately. Automation of natural language processing in online medical services can interpret patients' long sentences and extract important textual features, identify symptoms and categorize them into associating medical fields. The application of natural language processing in online medical services can not only reduce time spent for doctors and patients to discuss, document, and diagnose basic symptoms, but also improve the accuracy of basic symptoms diagnosis by avoiding potential human errors.

3. Literature survey

Fix the agreement mistake	time
Fix the agreement mistake	service
Correct pronoun usage	need
Change the wording	are able to
Fix the agreement mistake	service
Change preposition	for
Correct article usage	the last
Correct article usage	Hadoop
Remove redundancy	own
Correct your spelling	Khorsheed
Add a comma	and

Fig. 19. Using Grammarly tool check the report

Figure Labels: Use Grammarly tool to check our report grammers

E. Medical Symptoms Text Classification Model:

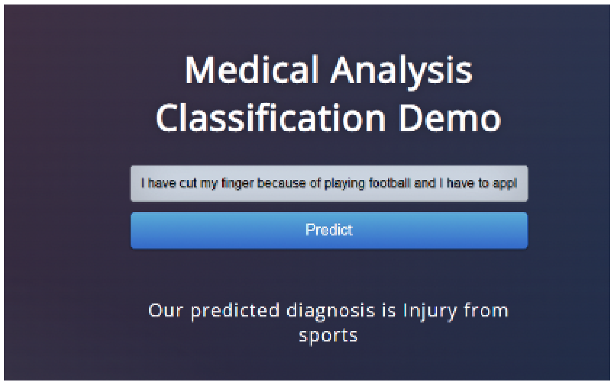
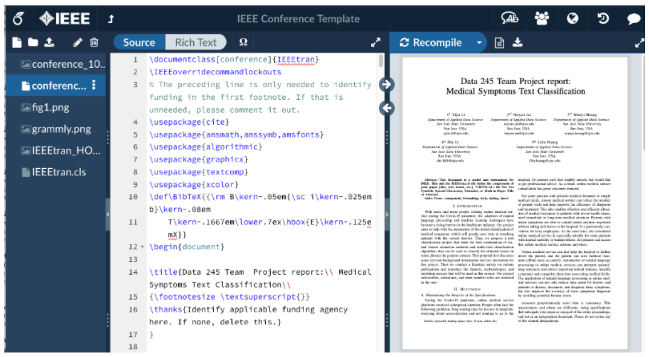

Medical Analysis Classification Demo
I have cut my finger because of playing football and I have to appl
Predict
Our predicted diagnosis is Injury from sports

Fig. 20. Medical Symptoms Text Classification Model

Figure Labels:Medical Symptoms Text Classification Model

F. Used LaTeX:

https://www.overleaf.com/read/kjtvxjqspyyp



```
\documentclass[conference]{IEEEtran}
\IEEEoverridecommandlockouts
% The preceding line is only needed to identify funding in the first footnote. If that is unneeded, please comment it out.
\usepackage{cite}
\usepackage{amsmath,amssymb,amsfonts}
\usepackage{algorithmic}
\usepackage{graphicx}
\usepackage{textcomp}
\usepackage{xcolor}
\def\BibTeX{{\rm B\kern-.05em{\sc i\kern-.025em b}\kern-.08em
    T\kern-.1667em\lower.7ex\hbox{E}\kern-.125emX}}
\begin{document}

\title{Data 245 Team  Project report:\\ Medical Symptoms Text Classification\\
{\footnotesize \textsuperscript{}}
\thanks{Identify applicable funding agency here. If none, delete this.}
}

```

Fig. 21. Used LaTex screenshot

G. Used Prezi to present

https://prezi.com/p/edit/fcedyp8cyfz0/