AUTO TAGGING OF STACKOVERFLOW QUESTIONS

Introduction

We need to come up with a way for categorizing all of the labels we have. Based on the inquiry's Body and Title, we'll create two vectors and connect them. Create a classification model with the vectors created to propose the top 25 tags that suit the Stack Overflow query. To achieve the most exact model, we'll employ Label Powerset and Classifier Chains.

In today's computer world, Stack overflow is perhaps the most used method for debugging programming issues. If the customer asks a question on Stack Overflow, he or she should physically label and label the subject he or she is looking for in order to get the problem/issue resolved/diverted to the proper local region of people who have previously worked on[4] the technology. For example, if we're looking for any, AI point with a question associated to some SVM model preparation, we'll need to mark it as Machine learning after submitting the query, so it gets to the right group of people. Rather of identifying Machine Learning, we may label Accounting as a solution to the problem.

If we would be able to develop a model that automatically labels each inquiry with its own errand, for example, it labels an AI issue with an AI tag and bookkeeping inquiries with a bookkeeping tag, etc. As a result, we shall create solutions/arrangements for the enquiries as soon as the proper group of people is found.

Targets of our undertaking are:

- Select a Dataset with a variety of inquiry types and Tags.
- Labels Collection: A collection of the most important Tags from many fields.
- Preprocessing the body and title of the inquiry, such as deleting Stop Words, converting all text to lowercase, stemming, and lemmatization.
- Vectorization is the process of creating vectors that will be used to build and test the layout model.
- Preparing: Using the vectors we've produced, we'll train the model to predict the Top 25 labels that best fit the questions.
- In light of the test data, determining the optimum model.
- Using Web apps to communicate the code.

We'll eliminate three main highlights from the inquiries, which provide us an overview of the complete scenario, such as what the inquiry is managing, what tags should be assigned, and the inquiry's title. These are the three elements that determine which questions belong in which areas.

Background

A large number of website pages on the internet have been linked to the exchange of information and opinions. On the internet, people upload a lot of questions, film reviews, and other information. In any case, isolating this data with appropriate labels has become a difficult task. In manual labeling, a high percentage of clients are unlikely to choose important labels that would limit the data's range. Despite the fact that many investigations were directed to design a model that could order content, many investigations were directed to create a model that could order content. This multilabel sequence is used in a variety of situations, such as determining if a film poll is favorable or negative. The topic is addressed in three steps in Kartik Nooney's article "Profound leap into multi-name order...! (With a specific Case Study)" published on Towards Science. Initially, data cleaning is achieved using NLP procedures such as stemming, lemmatization, and the removal of HTML labels during the information preparation step. The cleansed data is then partitioned into Test and Train datasets in the next step. The Label Powerset Classifier is used to complete the last stage of the process. Regardless, extra execution of cleaning techniques and order models, which we'd apply in this project, may be required to achieve better precision.

Model

We have used Label Powerset Classifier to classify the tags which need to be assigned to that task.

Model Architecture

We will have a large variety of tags which need to be assigned to that particular task which makes the problem as multi label classification problem. We transformed the multi label classification to single class classification using Label powerset which considers a single label as a separate single classifier i.e., We convert the problem to a multi-class problem and train a single multi-class classifier on all unique label combinations identified in the training data.

Workflow Diagram

Consider the data we have is as below diagram for single value of x we have multiple y labels i.e., for a single Input variable we must find multiple labels. For example, we there is a question (X) is provided as an input to the model it should find the possible Tags (Y) labels.

Consider the below figure where X variables are x1, x2, x3, x4 and y labels for x1 are y1(1), y1(2), y1(3) and x2 the y labels would be y2(1), y2(2), y2(3) and so on

X	Y(1)	Y(2)	Y(3)
A	0	1	0
B	1	1	0
C	0	1	0
D	0	0	1

Fig 1: Raw Dataset with multi labels

In the above figure there are total 4 features each feature is having 3 labels. If we observe the data with respect to the labels, we can observe that feature A & C are having same labels. So, our model treats it as single value. Our main goal is to convert all the multi labels into single label as shown below.

X	Y
A	1
B	2
C	1
D	3

Fig 2: Modified dataset with single labels

After converting the data into problem from multi label classifier to the single label classifier the modified data looks like the data in the above data

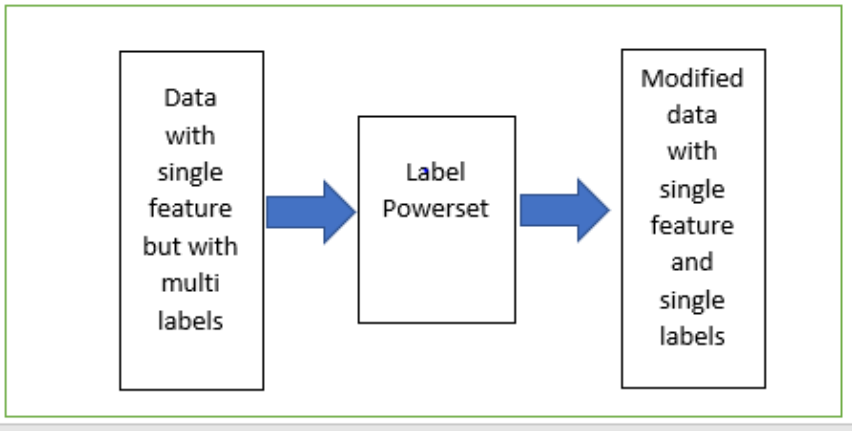

Fig 3: Working of Label powerset

Dataset

The dataset we're using has a total of six parts:

- **OwnerUserId:** This part contains information about the customer who reported the Stack flood problem. There are 875317 new lines in all. This segment provides no information regarding which labels should be distributed. As a result, we may ignore this portion while building our model.
- **CreationDate:** This field contains information about the date on which the Stackoverflow question was created. Even this portion does not provide information on which Tags should be designated. Rows are quite intriguing.
- **ClosedDate:** This will provide information once the inquiry has been resolved, which will aid our investigation.● Score: This segment gives the no of individuals who have upvoted this question i.e., all out no of individuals who have the connected inquiry.
- **Title:** This section contains information on the subject that we are in charge of. This is one of the most important aspects to consider. We have a total of 1263995 one-of-a-kind segments.
- **Body:** The body of the inquiry is where we acquire all of the information about the inquiry that has to be controlled nearby. Along these lines, this would be an important aspect of the job. The combination of Body and Title provides us with more exact information about the query, allowing us to assign the associated Tags to it.

Design of Features

We have identified three main highlights predicted to foresee the Tags expected to appoint to isolate enquiries from the plethora of highlights. TITLE, BODY, and TAGS are the three highlights that provide extra information about the enquiry.

This element gives us the short data about the inquiry i.e., it gives essential data that the inquiry has a place with which class.

For Example:

1. **Python error**

This suggests that the inquiry is about technology and the Python programming language in particular.

2. Journal ledger out of balance

This implies that the subject of the inquiry is accounting. As a result, the tags will be tied to the accounting field, using the tag journal ledger as an example.

The body of the question provides additional[3] details about the question with which we are dealing, as well as more background on the topic and the problem with which we should deal, so that we can give the appropriate tags to it.

For Example:

Despite the fact that numpy array has been imported, it is impossible to be imported.

This provides more information about the subject at hand. The above statement indicates that the issue is related to a library called numpy, which is unable to import.

Tags: The questions we're working on need to have tags given to them. We must apply one tag to each question linked to that certain issue, hence this feature is critical for us to specify the question's property.

Analysis of data

Data Pre – processing

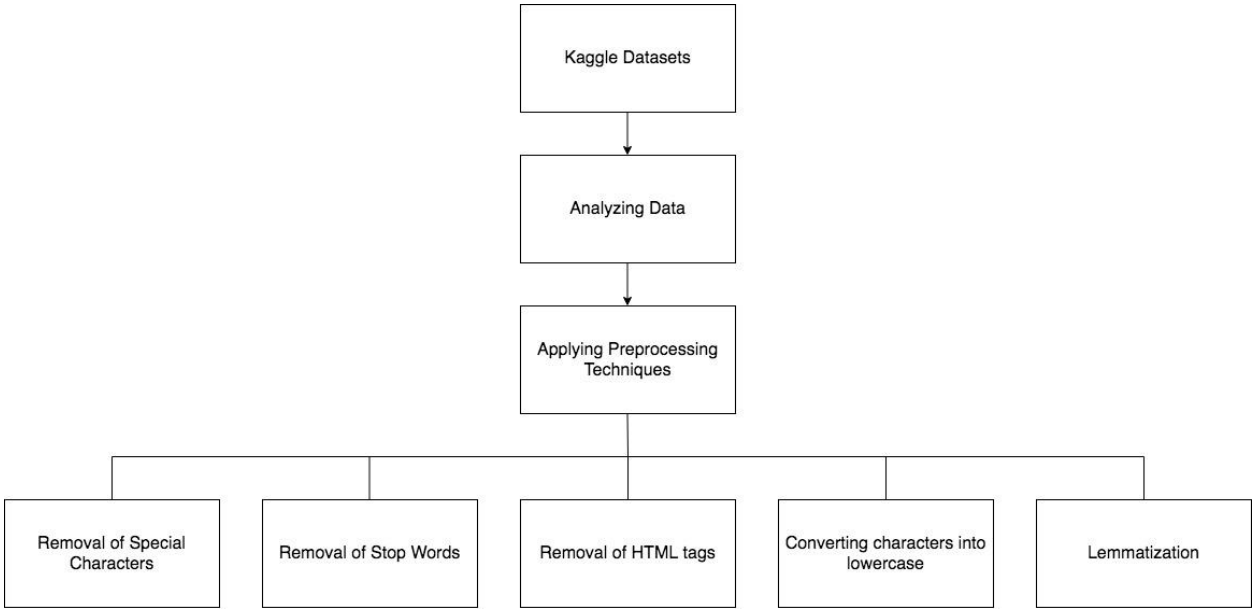

Fig 4: Detailed Data pre processing tasks

In a machine learning project, data preprocessing is a method which converts raw data into useful data or understandable data. There would be a large number of unwanted data in the raw data. With this unwanted data the efficiency of the model would decrease. Hence data preprocessing[1] techniques are applied on original data in order to remove irrelevant data. Here in this project we have used a few NLP text preprocessing techniques in order to clean our dataset which consists of stack overflow questions and their tags.

Firstly, we have removed special characters present in the title and body of the stack overflow questions. As the special characters in the data wouldn't have any meaning and would consume a certain amount of memory, we have removed these special characters. In the second step of data preprocessing, we have applied preprocessing techniques in order to remove stop words. The major reason behind removing stop words is that they don't have any meaning in the text and removing them would increase the model's time complexity. In the next step, we have removed HTML tags in the title and body as we won't be using them in training the models. Finally we converted all the words into lowercase and grouped them using lemmatization in order to make it easy to convert all the words into vectors using Tfidf vectorizer.

Implementation

Algorithm

The principal objective of this task is to plan a classifier which could auto label the inquiries in Stack overflow. We would utilize a Kaggle dataset which has a rundown of Stack overflow questions and their labels. At first, the dataset of inquiries what's more, labels would be removed into two dataframes one in each. Afterward, as a piece of information preprocessing, the undesirable data in questions is taken out utilizing NLP strategies, for example, Removal of unique characters from title and body, Removal of stop words, Removal of HTML labels, convert characters to lowercase and lemmatize the words. In the subsequent stage, we would change over the preprocessed information into vectors utilizing TFIDF Vectorizer. This vectorized information is additionally isolated into test and preparing information. Later utilizing the Label Powerset classifier we would order the Stack overflow inquiries with labels.

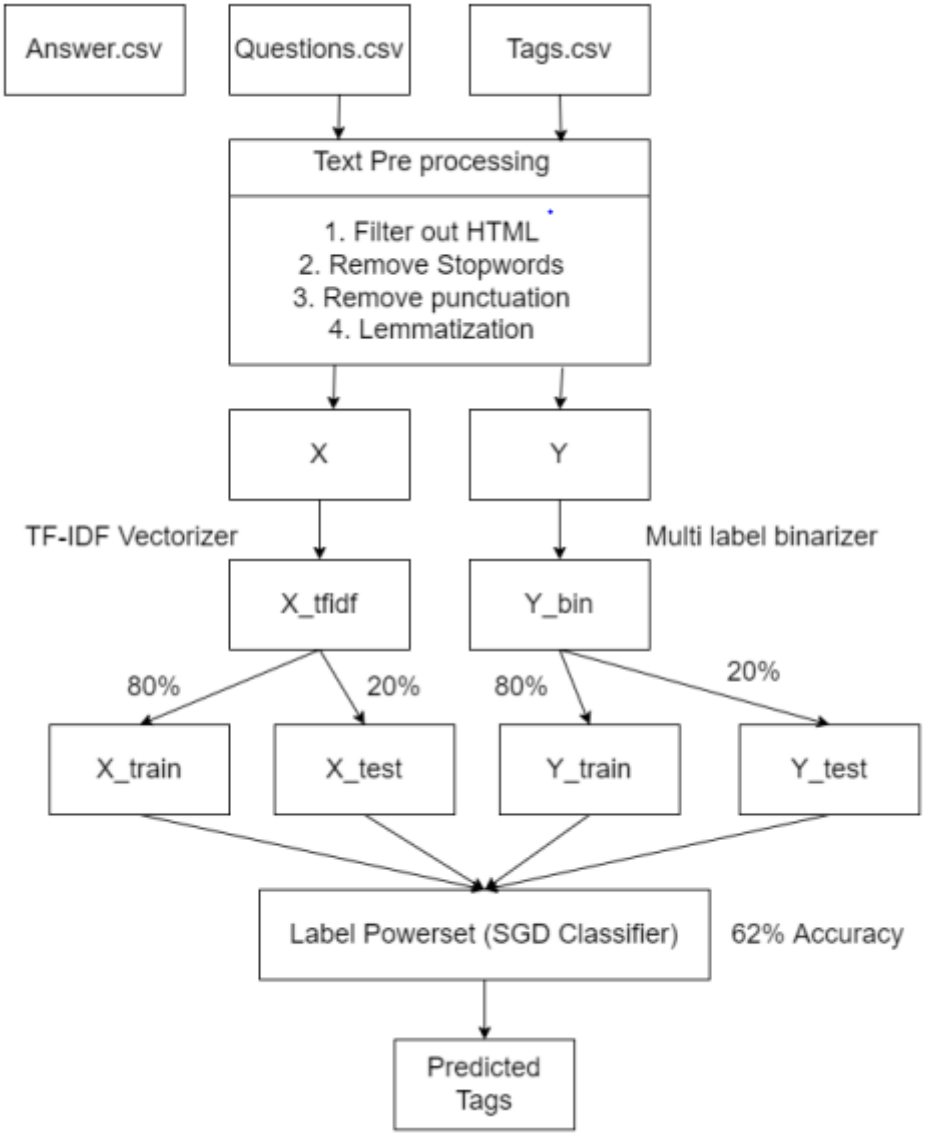

Fig 5: Workflow diagram for training the model

The first step is to merge all the data i.e., merge all the 3 dataframes answer.csv, questions.csv and Tags.csv into a one dataframe. The next step would be to preprocess the data like removing the stopwords, unnecessary punctuation marks and repeated words. We need to perform lemmatization to bring all the words to the root words because the word “book” and “books” have same meaning so there is not need to process the word “books”. The text received will contain the html tags because the dataset is extracted from web scrapping, so we have removed all the html tags.

After Text Preprocessing the text obtained is free from all the stopwords and unnecessary words. Then we split the data into X labels and Y labels. The data obtained is having multi labels, so we must convert the multi label into single

label, so we perform multi label binaries. Then we convert the data into vectors by performing the TF-IDF Vectorization. After creating the vectors, we are splitting the data into test and train data and passing it to the Label Powerset classifier which is also a SGB classifier. We train the model on training dataset and predict the tags for the unseen data. To Evaluate the model, we find evaluation metrics like finding precision, recall, accuracy and F1 score these are the factors tell us the how the model is performing.

GitHub Link :

https://github.com/Gagan-Achanta/CSCE_5290_NLP_Project

Dataset Link :

https://www.kaggle.com/datasets/stackoverflow/stacksample

Results

Evaluation metrics:

We use three evaluation metrics to evaluate our model: F1 score recall and precision. These evaluation indicators are used to determine the number of models available. Because a model may[2] perform well using one Evaluation metric, but not so well using another, evaluation metrics ensure that the model is working appropriately and optimally. If we don't utilize an alternative assessment metric, we may end up with a model that performs poorly on unknown data.

True positive (TP) occurs when Predicted value is positive with a positive actual value.

True Negative (TN) occurs when Predicted value is Negative with a Negative value.

False Positive (FP) occurs when a positive value is predicted but the actual value is negative.

False Negative (FN) occurs when a Predicted Negative with a Positive value.

Precision:

Precision indicates how many projected classes were correct and turned out to be positive. False positives are a greater issue than false negatives. False negative is a greater problem than false positive.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

Recall:

Recall shows how many actual positive cases are correct and truly turned out to be positive.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

F1 Score:

It is a matrix that combines concept precision and recall. Precision and Recall have a harmonic mean.

$$F1 = 2.\frac{Precision \times Recall}{Precision + Recall}$$

In this project we have used 3 different classifiers in order to classify stack overflow questions with their tags after training and validation of these models on the dataset we would be selecting suitable model by comparing F1 score and accuracy. The 3 machine learning model are Label Powerset, Classifier chains and binary relevance.

Label Powerset:

Initially using label powerset algorithm we have performed training and testing on the vectorized dataset. After performing training and testing we have evaluated the model using the evaluation metrics using F1 Score and accuracy.

We have obtained F1 score = 0.73, Accuracy = 0.62, Recall = 0.68 and Precision = 0.76

```
svc = LinearSVC()
sgd = SGDClassifier(n_jobs=-1)

clf = LabelPowerset(svc)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print_score(y_pred, clf)

kfold = KFold(n_splits=5)
X_sparse = X_tfidf.tocsr()

scores = []

for train_indices, test_indices in kfold.split(X_sparse, y_bin):
    clf.fit(X_sparse[train_indices], y_bin[train_indices])
    print(clf.score(X_sparse[test_indices], y_bin[test_indices]))
    scores.append(clf.score(X_sparse[test_indices], y_bin[test_indices]))

print(sum(scores)/len(scores))
```

```
Clf:  LabelPowerset
Accuracy score: 0.6237123235406333
Recall score: 0.6885059216519891
Precision score: 0.7651771625345002
Hamming loss: 2.5730637161388783
F1 score: 0.7374719315043108
```

Fig 6: Implementation of label PowerSet

Classifier Chains:

Believing that we would be getting better evaluation results when compared to label powerset model we have selected we have selected the classifier chains and performed training and validation on dataset. After testing the dataset using classifier chains model we have obtained a evaluation metrics as below

F1 score = 0.82, Accuracy = 0.59, Recall = 0.68 and Precision = 0.83

```
chains = [ClassifierChain(svc, order='random', random_state=i)
          for i in range(10)]

for chain in chains:
    chain.fit(X_train, y_train)

Y_pred_chains = np.array([chain.predict(X_test) for chain in
                          chains])

Y_pred_ensemble = Y_pred_chains.mean(axis=0)
ensemble_accuracy_score = accuracy_score(y_test, Y_pred_ensemble >= .5)
ensemble_recall_score = recall_score(y_test, Y_pred_ensemble >= .5, average='weighted')
ensemble_precision_score = precision_score(y_test, Y_pred_ensemble >= .5, average='weig
hted')
ensemble_f1_score = f1_score(y_pred, Y_pred_ensemble >= .5, average='weighted')
hamm = hamming_loss(Y_pred_ensemble >= .5, y_test)*100
print(ensemble_accuracy_score, ensemble_recall_score, ensemble_precision_score, ensembl
e_f1_score, hamm)
```

```
0.5946203739030904 0.6818250835104768 0.8311283015789716 0.820683699907695
2.2499046165585654
```

Fig 7: Implementation of Classifier Chains

Binary Relevance:

After comparing Label powerset and classifier chains we can observe that label powerset has given better results than classifier chains. Hopping to obtain the much more better results we have implemented binary relevance and perform testing and training on the dataset. After training and testing the dataset using binary relevance we have obtained the following results

F1 score = 0.74, Accuracy = 0.56, Recall = 0.64 and Precision = 0.84

```
# initialize binary relevance multi-label classifier
# with a gaussian naive bayes base classifier
classifier = BinaryRelevance(svc)

# train
classifier.fit(X_train, y_train)

# predict
predictions = classifier.predict(X_test)

print_score(predictions, classifier)
```

```
Clf:  BinaryRelevance
Accuracy score: 0.5628576879053796
Recall score: 0.6471302763437595
Precision score: 0.8452195939969764
Hamming loss: 2.3170545593285006
F1 score: 0.747106233552806
```

Fig 8: Implementation of Binary Relevance

Final Outcome:

As it can be clearly observed while comparing all the 3 models label powerset has provided the better results as compared to Classifier chains and binary relevance. The results can be clearly seen in the below table.

Model	Accuracy	Recall	Precision	F1 Score
Label Powerset	0.62	0.68	0.76	0.73
Classifier Chains	0.59	0.68	0.83	0.82
Binary Relevance	0.56	0.64	0.84	0.74

Fig 9: Final Results

Deployment

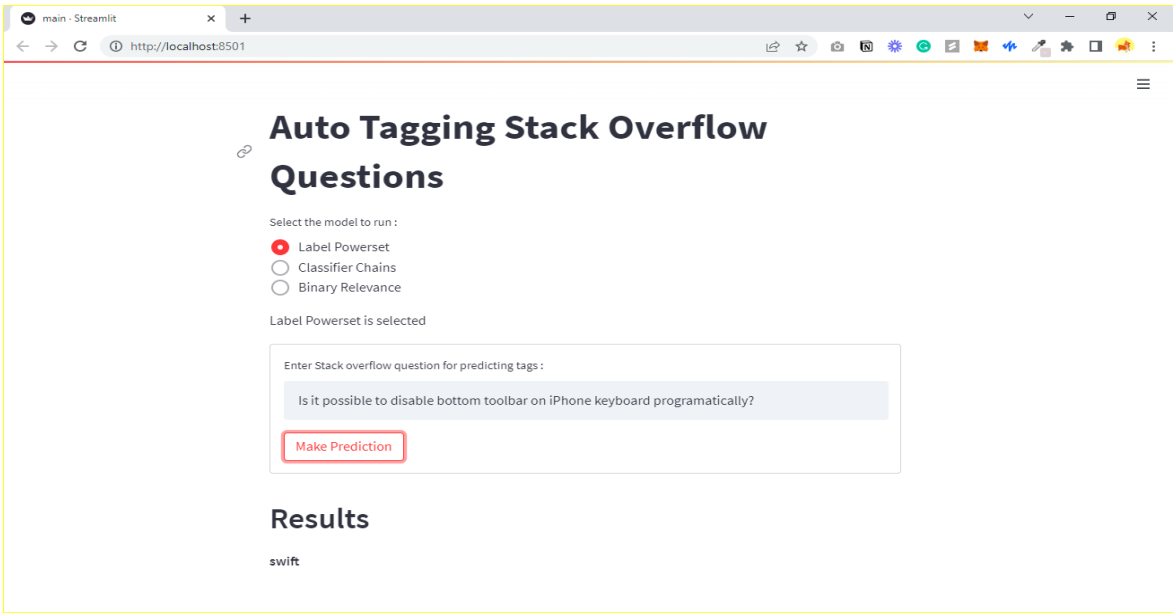

Fig 10: Running on Label Powerset and getting appropriate predictions

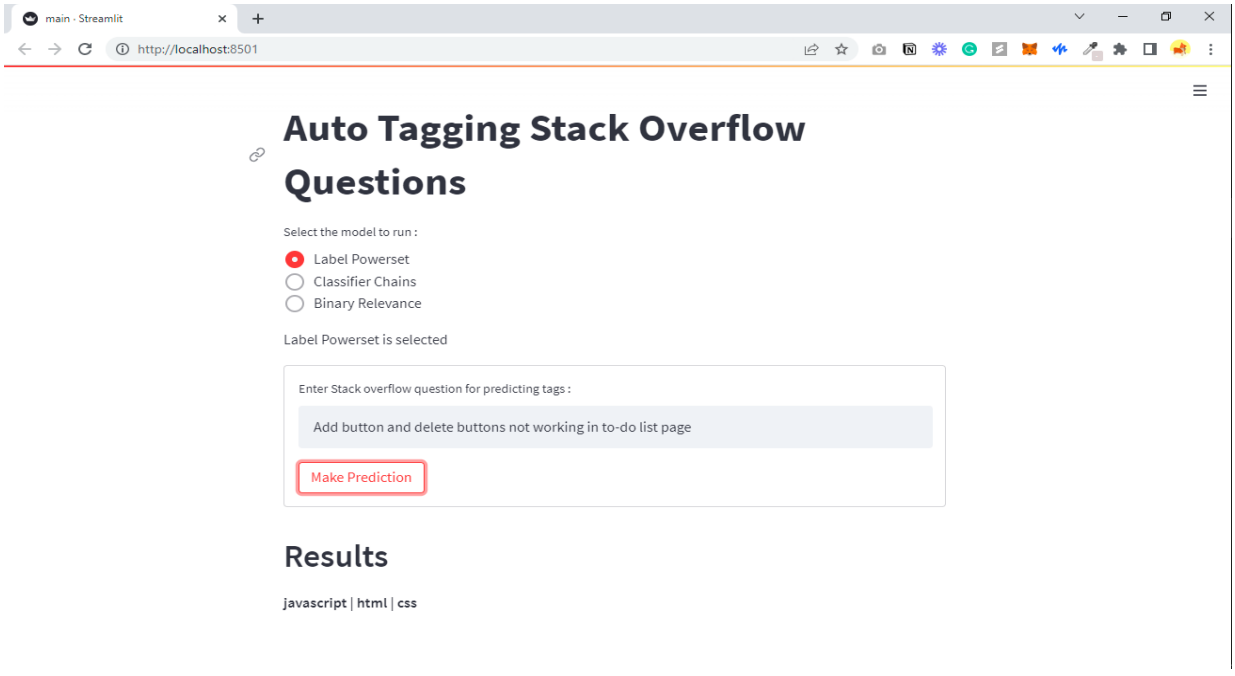

Fig 11: Trying different questions and predicting the tags

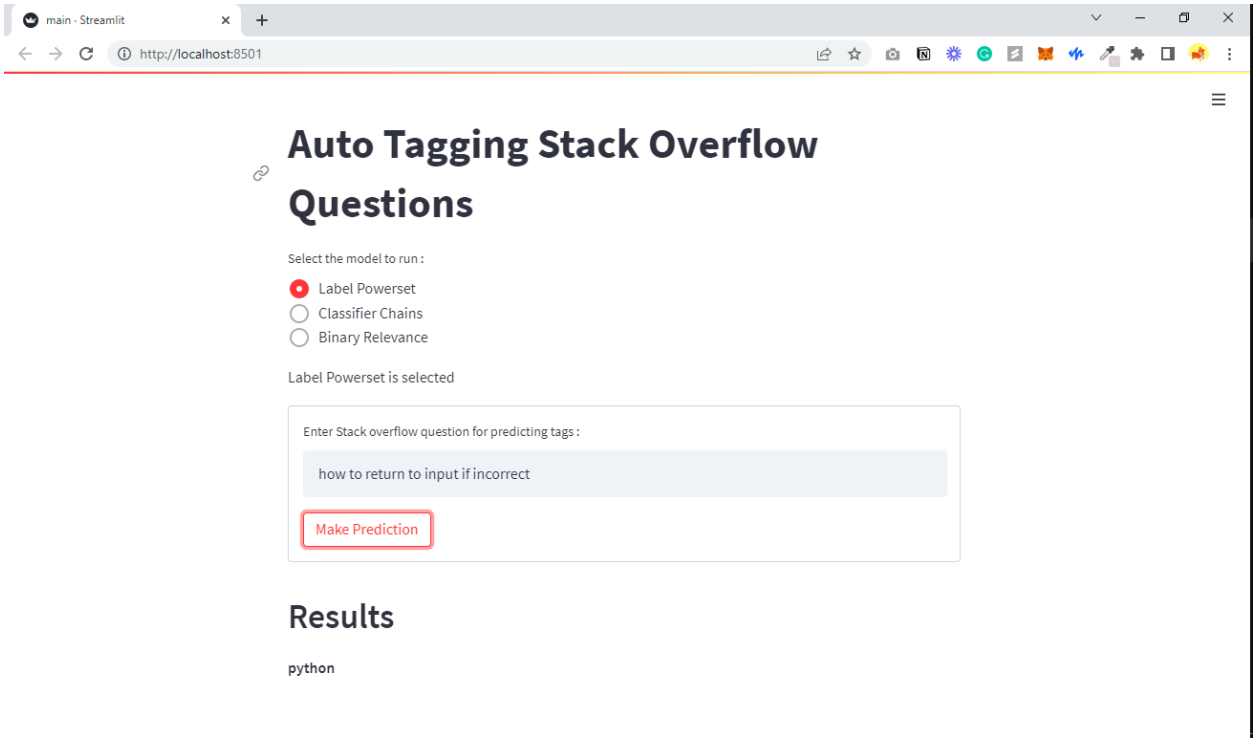

Fig 12: Trying another question using Label Powerset for prediction

Project Management

Implementation status report

Work completed

Description:

In the beginning, we looked for the datasets which includes a list of stack overflow questions and fortunately we found a similar dataset on Kaggle. We have downloaded that Kaggle dataset and have analyzed it to get a proper understanding of it. Once we understood about the dataset, in the next step we imported it using Python code. Later, we have created a couple of dataframes for a list of questions and for their tags. Once the dataframes has been created, we have performed preprocessing techniques on these dataframes in order to remove unwanted information in the data. The preprocessing techniques includes Removing of Special characters, removing stop words, removing HTML tags, converting all characters into lowercase and lemmatization. In the next step we have created vectors from data using TfIDF vectorizer. Followed by creation of vectors, we have split the data into train and test datasets 80% and 20% accordingly. Then we have trained Label Powerset, Classifier chains and Binary relevance classifiers in order to classify the stack overflow questions with respective tags. At the end, after comparing accuracy and F1 score of all classifiers we came to a conclusion that Label Powerset would be the best fit for the data in order to classify the Stack overflow questions.

Responsibility (Task, Person):

Aditya Pujari:

- Explored and selected a pepper dataset which consists of stack overflow questions and their tags.
- Examined the data if it has any null values and unwanted information before applying data preprocessing techniques.
- Scrutinized many machine learning and Natural Language processing algorithms and selected suitable algorithms to achieve the goals of the project.

Gagan Sai Ram Anvesh Achanta:

- Examined the csv files and observed most repeated tags with their questions of stack overflow.
- Observed the whole dataset and selected features which would help the model to classify the questions with their respective tags. Also, removed unwanted features like Upwords, identity.
- Removed HTML tags and other unfavoured information from features.

Sai Tarun Gunda:

- Applied cleaning techniques on the dataset such as Stop words removing, Special characters removing, lower case conversion and lemmatization.
- Divided the dataset into 80% of Training data and 20% of testing data.
- Created vectors from the data in train and testing datasets using Tfidf vectorizer.

Nithish Reddy Boddhi Reddy:

- Designed machine learning classifiers using Label Powerset, Classifier chains and Binary relevance in order to train and test the dataset.
- Implemented K fold cross validation for Label Powerset model with no.of splits 5.
- Calculated evaluation metrics accuracy and F1 Score for all the classifiers Label Powerset, Classifiers Chain and Binary Relevance.

Contributions (members/percentage)

Aditya Pujari	25%
Gagan Sai Ram Anvesh Achanta	25%
Sai Tarun Gunda	25%
Nithish Reddy Boddhi Reddy	25%

Issues/Concerns

- The accuracy we achieved in all the three models is less, since the dataset we used is inadequate to train and test all the three models.
- Few crucial features in the dataset have been lost while applying text preprocessing techniques on the dataset.
- The best accuracy out of all three models is only 62% for the Label Powerset Classifier. In a real world scenario the model would predict unrelated tags to the Stack overflow questions.

References

1. Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. CoRR, abs/1301.3781.
2. https://www.kaggle.com/competitions/multilabel-bird-species-classification-nips2013/data
3. https://towardsdatascience.com/journey-to-the-center-of-multi-label-classification-384c40229bf
4. https://www.analyticsvidhya.com/blog/2017/08/introduction-to-multi-label-classification/