***Questions Tagging System***

Members

|  |  |  |
| --- | --- | --- |
| Name | ID | Section |
| محمد سامي عبد الكريم احمد | 20201700691 | 7 |
| مينا اشرف ميخائيل صالح | 20201700888 | 9 |
| دعاء يحيي اسماعيل حسن | 20201700243 | 3 |
| يوسف عبد الحميد فرج | 20201701249 | 10 |
| كريم حسين عبده جويلي | 20201700605 | 6 |

**Introduction**

**Background**

Sites that are specifically designed to have questions and answers for their users like Quora and Stack Overflow often request their users to submit five words along with the question so that they can be categorized easily. But sometimes users provide wrong tags which makes it difficult for other users to navigate through. Thus, they require an automatic question tagging system that can automatically identify correct and relevant tags for a question submitted by the user.

**About the Data**

Data used: 10% sample from Stack Overflow Q&A on Kaggle

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

**Data Cleaning and Preparation**

The dataset contains thousands of questions with their tags and answers, in this project we are only concerned about the questions and their respective tags.

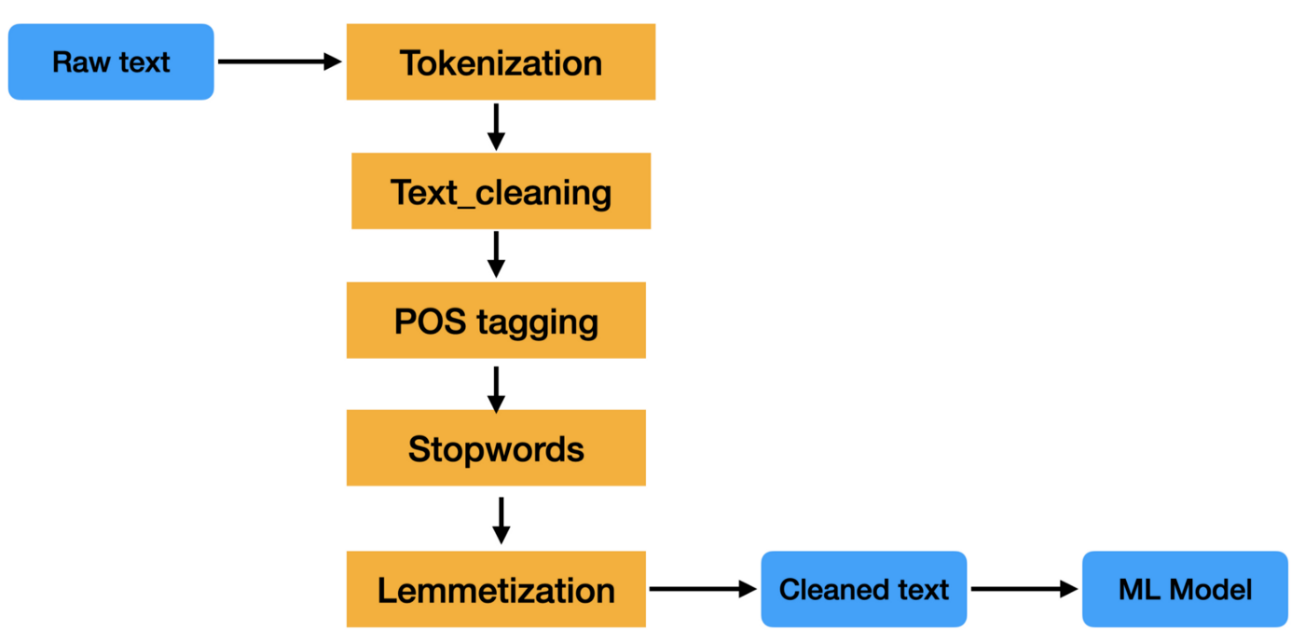
**Tags cleaning**

The tags dataset is cleaned by grouping up all tags of each single question in one row rather than many rows. Then a filtration process is done to filter the top 100 common tags to be used in this project.

**Questions cleaning**

Questions are first mapped to their respective tags in one dataframe. Then the new dataframe is filtered to contain only questions with a good rating on stack overflow (from the rating column), a threshold of 5 is used in this project. Finally HTML tags are removed from the questions bodies using beautiful soup library.

**Preprocessing and NLP pipeline**



The raw text data passes through a pipeline of NLP tasks to prepare it for the classifiers later on.

**NLP Pipeline Steps Used**

**Removal of Punctuation**

Punctuation characters are removed because they deliver no value for most NLP tasks.

**Tokenization**

Tokenization is the breaking down of text into smaller units called tokens, tokens are usually words, but they can be any meaningful unit of text. In this project word\_tokenize from NLTK was used, other tokenizers like TokTokTokenizer yielded similar results.

**Stop Words Removal**

Stop words are words that appear commonly in a language, they are considered to be of little value when it comes to the meaning of a sentence. In this project stop words were removed using English stop words provided by NLTK.

**Stemming /Lemmatization and POS tagging**

**Stemming**

It is a rule based heuristic approach that is used to reduce words to their base or root by removing affixes. The result of stemming is not always a valid word in the language. Stemming is usually faster than lemmatization as it is less computationally expensive.

**Lemmatization**

Similar to stemming, lemmatization is used to reduce words to their root, but the resulting words are always valid linguistically. Lemmatization takes more time to compute but it is more accurate.

**POS tagging**

POS tagging is a process of NLP that assigns a grammatical category for each word in a sentence, it can be used with lemmatization or stemming for better results.

In this project, different approaches were tried and it was seen that there is no marginable difference between using stemming or lemmatization, there is also not so much gain from using POS tagging.

**Models Creation, Evaluation and Visualization**

**Introduction**

To understand what kind of models need to be used in this project, it is important to understand what kind of problem we are trying to solve. This project requires a classification model to predict tags for questions. This problem is a multi -label classification problem and not a multi-class classification problem.

**Multi-label classification**

Classification in which each sample can be associated with multiple classes/labels, an example to that is classifying the topic of a book, it can be many things like “entertainment”, “science fiction” etc.. at the same time.

**Multi-class classification**

Classification where each sample must have only one class/label, an example to that would be classifying the picture of an animal whether it is a cat or a dog or a cow etc.. it can only be one of the three and no more than one.

A screenshot of a computer

Description automatically generated with low confidence

**Classifier Chain**

Classifier chain basically extends traditional binary classifiers’ functionality to handle multi-label tasks by considering label dependencies. A separate binary classifier is trained for each label, the classifier chain assumes that there is some correlation or dependency among the labels.

**Multi Output Classifier**

It extends simple binary classifiers similar to classifier chain, but it does not consider any dependencies between labels, instead it creates a classifier for each label and a classification task is done for each label independently of the others and the final result is the grouped results of all classifiers.

In This project both methods were tried and there was no significant difference between the two approaches.

**TF-IDF**Term Frequency-Inverse Document Frequency is a commonly used technique in natural language processing (NLP) for feature extraction from text data. It is used to quantify the importance of a term (word) in a document within a collection or corpus.

The TF-IDF score of a term is a product of two components: term frequency (TF) and inverse document frequency (IDF). It was used for feature extraction in this project.

**Metrics**

**Accuracy**

It is a common evaluation metric used to measure the performance of a classification model. It represents the proportion of correctly classified instances out of the total number of instances in a dataset.

**Hamming Loss**Hamming loss is an evaluation metric used in multi-label classification problems to measure the accuracy of predictions when dealing with multiple labels or classes. It calculates the fraction of labels that are incorrectly predicted compared to the total number of labels.

**Jaccard Score**

Jaccard score, also known as Jaccard similarity coefficient or Jaccard index, is an evaluation metric used to measure the similarity between two sets. In the context of classification tasks, the Jaccard score is commonly used to assess the similarity between the predicted and true sets of labels in multi-label classification problems.

The Jaccard score calculates the size of the intersection of two sets (the number of common elements) divided by the size of their union. It provides a measure of the overlap or similarity between the sets, ranging from 0 to 1.

Mathematically, the Jaccard score is defined as:

Jaccard score = (Size of Intersection) / (Size of Union)

For multi-label classification, the Jaccard score is often computed for each instance and then averaged across all instances to obtain an overall score.

The Jaccard score is particularly useful in multi-label classification problems where there can be multiple correct labels for each instance. It measures the proportion of correctly predicted labels compared to the union of predicted and true labels. A higher Jaccard score indicates a greater degree of similarity and better performance. All these metrics were used in the project.

**Visualization**

A concise confusion matrix that shows TP, TN, FP, FN is used in this report, another approach would be to make a confusion matrix that contains all 100 tags results, but the result is very unreadable.

**Models Report**

**A picture containing text, screenshot, rectangle, diagram

Description automatically generatedStochastic Gradient Descent Classifier**

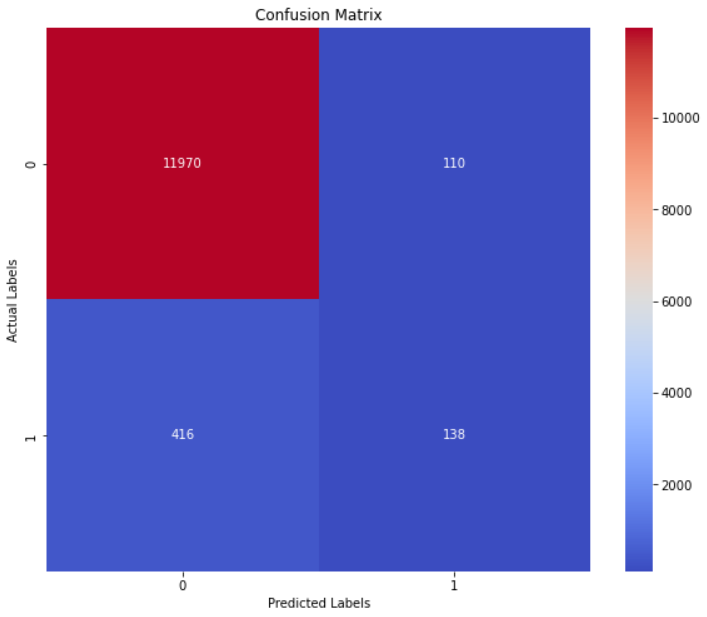
It is a linear classifier in scikit-learn that uses stochastic gradient descent optimization for efficient training. It supports various loss functions, regularization techniques, and adaptive learning rates. It is useful for binary or multi-class classification tasks, especially on large-scale datasets. It can handle imbalanced data and supports partial fit for incremental learning.

**Accuracy: ~**34%

**Jaccard Score: ~**41%

**Hamming Lose:** ~1%

**Linear SVC**

It is a linear support vector classifier in scikit-learn. It is a variant of the Support Vector Machine (SVM) algorithm specifically designed for linear classification tasks. It is well-suited for both binary and multi-class classification problems. It aims to find the optimal hyperplane that separates different classes by maximizing the margin between them.

**Accuracy: ~**40%

**Jaccard Score: ~**49%

**Hamming Lose:** ~1%

**Passive Aggressive Classifier**

**A picture containing text, screenshot, rectangle

Description automatically generated**It is used for binary classification tasks. It is particularly useful when dealing with large-scale and streaming data, as it can adapt and update the model incrementally as new instances arrive. The PA Classifier works by making updates to the model based on the misclassified instances it encounters during training. Instead of minimizing a regular loss function, it uses an aggressive update strategy to quickly correct misclassifications while maintaining stability.

**Accuracy: ~**38%

**Jaccard Score: ~**47%

**Hamming Lose:** ~1%

**t-distributed stochastic neighbor embedding (t-SNE):**

a statistical method for visualizing high-dimensional data by giving each datapoint a location in a two or three-dimensional map based on principle component analysis PCA.



**Conclusion**:

**The best model was the one that used the Linear SVC classifier. With the highest Accuracy and Jaccard score recorded among the others.**