Automatic Questions Tagging System

Team ID: T006

TA. Asmaa Bahai ElDien

Name	Seat Number	Department
مينا لطفي فايز عبدالله	20201700895	CS
كيرلس عزيز جلال عزيز	20201701116	CS
مينا خليفة جندي خله	20201700892	CS
مارى سعد يوسف سعد سليمان	20201700631	CS
مينا انيس شكري معوض	20201700889	CS
ميشيل مجدي حلمي رزق	20201700886	CS

1- **Preprocessing:**

First we merged the two csv files (Questions.csv, Tags.csv) and grouped by the ID and merged the tags of each question in a list as a multilabel column for each question.

As the huge size of the dataset(10^6 row) we wanted to filter them to get the most useful data, so we filtered using the score of each question so the question that their score is less than (10) will be filtered.

we wanted to remove extra spaces in the tags list so we used the remove_spaces function to make this task, then we got the most frequent tags as their was large number of tags and we wanted to reduce them and use the most used tags in the dataset to predict more accurate, while we remove the unique tags , some empty lists were found so we replace in them NONE value to drop these rows afterwards.

Pre processing of the Title and Body:

First we prepared the data ,so we removed the html tags from the **BODY** using the regular expressions .

Then tokenize the text into words, removed the stop words ,removed punctuation characters, and at the end stemming and lemmatizing.

2- Feature Extraction:

TFIDF:

The reason we use TF-IDF is that it helps us to identify words that are significant within a given text or document. It does this by measuring the frequency of a word in a document, while also considering how often the word appears in other documents in the same corpus. Words that appear frequently in a document but do not appear frequently in other documents in the corpus are more important or unique to that document.

Hyperparameters:

a) **analyzer:** This parameter specifies whether the feature should be made up of individual words or character n-grams (sequences of n characters).

b) **min_df:** This parameter sets a minimum threshold for the number of documents that a word must appear in to be included in the vocabulary. It can take either an integer value (minimum number of documents) or a float value (minimum fraction of total documents).

c) **max_df:** This parameter sets a maximum threshold for the frequency of a word beyond which it is considered irrelevant and excluded from the vocabulary. It can take either an integer value (maximum number of documents) or a float value (maximum fraction of total documents).

d) **strip_accents:** This parameter specifies how to handle accents in the input text. If set to None, no action will be taken. If set to 'ascii', all accentuated characters are converted to their ASCII equivalent. If set to 'unicode', all Unicode accent characters are stripped.

e) **encoding:** This parameter specifies the character encoding scheme to be used when reading the input text.

f) **ngram_range:** This parameter specifies the range of n-gram sizes to use. For example, setting ngram_range=(1,3) means that the vectorizer will consider unigrams, bigrams, and trigrams.

g) **preprocessor:** This parameter specifies a callable function that can be applied to each document before tokenization.

h) **token_pattern:** This parameter specifies a regular expression used to match tokens. By default, the tokenizer splits tokens based on whitespace.

i) **max_features:** This parameter limits the maximum number of features (i.e., distinct terms) that the vectorizer will generate.

stack the title and body and split the data:

Then we joined the two columns to one column using hstack() so we can send them to the model and splitted into training and testing data

multilabel binarizer:

In multi-label classification tasks, each sample can belong to multiple classes or categories simultaneously. However, most machine learning algorithms, including Naive Bayes, are designed to handle binary classification tasks that have only two possible outcomes (e.g., positive, or negative).

To apply these algorithms to multi-label classification problems, we need to convert the multi-label targets into multiple binary classification problems - one for each possible label. This means representing each possible label as a separate binary column with values of 0 or 1 indicating whether a particular sample belongs to that label or not.

The MultiLabelBinarizer class provided by scikit-learn does exactly this. It takes a list of tags for each sample and converts it into a binary array where each column represents a unique tag. A value of 1 in a particular column indicates that the corresponding tag is present for that sample, while a value of 0 indicates that it is not.

3- A) Linear SVC model and accuracy:

Linear SVM is a binary classification algorithm that tries to find the optimal hyperplane that separates two classes in a high-dimensional space. The basic idea is to find the hyperplane with the largest margin between the two classes.

To do this, we first map the input data points to a higher-dimensional space using a transformation function (called a kernel function) which allows us to find a hyperplane that can separate the two classes.

The hyperplane that maximizes the margin is the one that has the largest distance to the closest data point of each class, also known as support vectors. This distance is called the margin.

The optimization problem for finding the optimal hyperplane can be formulated as a constrained optimization problem where we aim to maximize the margin subject to the constraint that all data points are correctly classified.

In practice, it's not always possible to find a hyperplane that perfectly separates the two classes, so we introduce a penalty parameter C that controls the trade-off between maximizing the margin and minimizing the classification error on the training data. A larger value of C leads to a narrower margin but fewer misclassifications on the training data.

Once the hyperplane is found, we can use it to predict the class of new data points based on which side of the hyperplane they fall.

B) SGDClassifier Model:

The SGDClassifier is a linear classification model that belongs to the family of gradient descent algorithms. It is a type of linear classifier that uses an iterative approach to optimize the loss function using stochastic gradient descent.

Specifically, the algorithm works by randomly selecting a subset of the training data (known as a minibatch) and updating the model's weights based on the errors made on these examples. This process is repeated for a number of iterations until the model converges or a stopping criterion is met.

Hamming loss:

The Hamming loss is a measure of the dissimilarity between two sets of binary data. It is defined as the fraction of labels that are incorrectly predicted by a classifier compared to the true labels.

Hamming-Loss is the fraction of labels that are incorrectly predicted, i.e., the fraction of the wrong labels to the total number of labels.

The Hamming loss is useful when dealing with multi-label classification problems where each instance can belong to more than one class simultaneously. Unlike other metrics such as accuracy, it considers the fact that multiple labels may be assigned to each instance.

F1 Score:

F1 score is a commonly used metric in machine learning and statistics to evaluate the performance of classification models. It balances the precision and recall of a model, giving a single number that summarizes its overall quality.

Precision and recall are two measures used to evaluate how well a model performs on a particular class. Precision is the fraction of true positives (i.e., correct predictions of the class) out of all positive predictions made by the model, while recall is the fraction of true positives out of all actual positive instances in the dataset.

The F1 score is the harmonic mean of precision and recall, and ranges from 0 to 1. It is calculated as follows:

F1 = 2 * (precision * recall) / (precision + recall)

A high F1 score indicates that a model has both high precision and high recall, meaning it can accurately identify positive instances while minimizing false positives and false negatives. Conversely, a low F1 score means that a model may be biased towards one of these metrics at the expense of the other, resulting in poor overall performance.