Question Tagging System

CS-T066

TA : Dr.Asmaa

Name	ID	Section
مينا عادل لويز متى جرجس	20201700894	9
مينا نبيل اسعد نجيب	20201700897	9
مارلين تاوضروس يعقوب تاوضروس	20201701119	6
مارتينا صبرى مسعد عجايبى	20201700626	6
محمد تامر محمد محمدى	20201700677	7

Introduction : -

This code is a data preprocessing and model training pipeline for a multi-label classification task on stackoverflow questions. The code involves importing necessary libraries such as pandas, BeautifulSoup, nltk, etc. It then filters the data by reading in two CSV files, one containing the questions and the other containing their respective tags. The questions are filtered to only include those with a score greater than 5.

The data is then further processed by grouping the tags by ID, filtering out the most common tags, cleaning the text data by removing stopwords and punctuation, and vectorizing the text using TF-IDF vectorization.

A neural network model is then trained on the preprocessed data. The model architecture consists of several dense layers with varying units and activation functions. The model is compiled with the Adam optimizer and binary cross-entropy loss function. Hamming loss and accuracy metrics are used to evaluate the model during training. Early stopping is also implemented to prevent overfitting.

The trained model is then pickled to avoid retraining in the future. The code evaluates the model on the test set and generates a confusion matrix heatmap to visualize the results.

The goal of this project is to develop a model that can classify questions to their tags based on some classification methods and algorithms which will be explained later in this document along with how the model is build using the provided datasets.

In summary, this code preprocesses and trains a neural network model on stackoverflow questions data for multi-label classification.

<u>Data Filtering : -</u>

When dealing with large datasets, data filtering can be a useful technique to extract the most relevant information.

- In the beginning , the data provided was too large and hard to process so we had to reduce the amount of it and we did it by filtering the questions based on their **score** , any question with **Score > 5** is included in the next stages
- One common approach is to identify the most common "tags" or words in the dataset, and then filter the data to only include rows that contain one or more of those tags.

  To do this, you can use a technique called frequency distribution (freqdist) analysis. Freqdist is a method of counting the frequency of words or phrases in a dataset, and is often used in natural language processing and text mining.

  Using freqdist, you can identify the top 50 most common words or phrases in the dataset, and then remove any rows that do not contain any of those words or phrases. This can help you to quickly identify the most relevant data for your analysis, and to filter out irrelevant or less important information.
- And finally, we took a sample of 30000 row and saved it as CSV file to use it in the model we're about to build.

<u>Preprocessing : -</u>

The purpose of this project was to preprocess a dataset of text data, in order to make it suitable for further analysis or modeling. The text cleaning process involved removing HTML tags, converting text to lowercase, tokenizing the text into individual words, removing stop words and punctuation, and stemming the words to their base form.

Method : We began by defining a function called **text_cleaning** that takes a string of text and an optional argument called Stemmer, which defaults to the **PorterStemmer** algorithm. The function uses the **BeautifulSoup** library to extract the text from the input string, and remove any HTML tags or markup. It then converts the text to lowercase, and tokenizes it into individual words using the **word_tokenize** function from the NLTK library.

Next, we created a set of stop words and punctuation marks to remove from the text, using the **stopwords** and string libraries. We created a filtered list of words by removing stop words and punctuation marks from the tokenized text. We then used a **stemming algorithm** (specified by the Stemmer argument) to reduce each word in the filtered list to its base form. Finally, the function returned the filtered and stemmed words as a single string.

After defining the text_cleaning function, we applied it to two columns ('**Body**' and '**Title**') in a pandas DataFrame called qustions_with_tagging. We used the apply method to apply the function to each row in the specified columns, and returned the cleaned text as a new value in the DataFrame. Finally, we used the head() method to display the first few rows of the cleaned DataFrame.

The text cleaning process was successful in removing HTML tags, converting text to lowercase, tokenizing the text into individual words, removing stop words and punctuation, and stemming the words to their base form. This process helped to standardize the text data and prepare it for further analysis or modeling.

In conclusion, text cleaning and preprocessing is an important step in preparing text data for analysis or modeling. By removing noise and **standardizing** the text data, we can ensure that our analysis or modeling results are accurate and reliable. The use of libraries like **BeautifulSoup**, **NLTK**, and **pandas** makes this process straightforward and efficient.

Feature Extraction : -

After performing the text cleaning process on the "Body" and "Title" columns of the dataset, we applied the **TF-IDF** (term frequency-inverse document frequency) algorithm to these columns. TF-IDF is a statistical method that is used to evaluate the importance of a word in a document based on how frequently it appears in the document, and how commonly it appears across all of the documents in the dataset. The application of TF-IDF allowed us to assign weights to each word in the "Body" and "Title" columns based on their importance in the dataset, which can be useful for further analysis and modeling.

Model Training : -

In our project, we used the TF-IDF algorithm to extract features from preprocessed text data, and then trained a neural network model to perform multi-label classification. Neural networks are a type of machine learning algorithm that are loosely based on the structure and function of the human brain. They consist of interconnected nodes that process and transmit information through a series of layers.

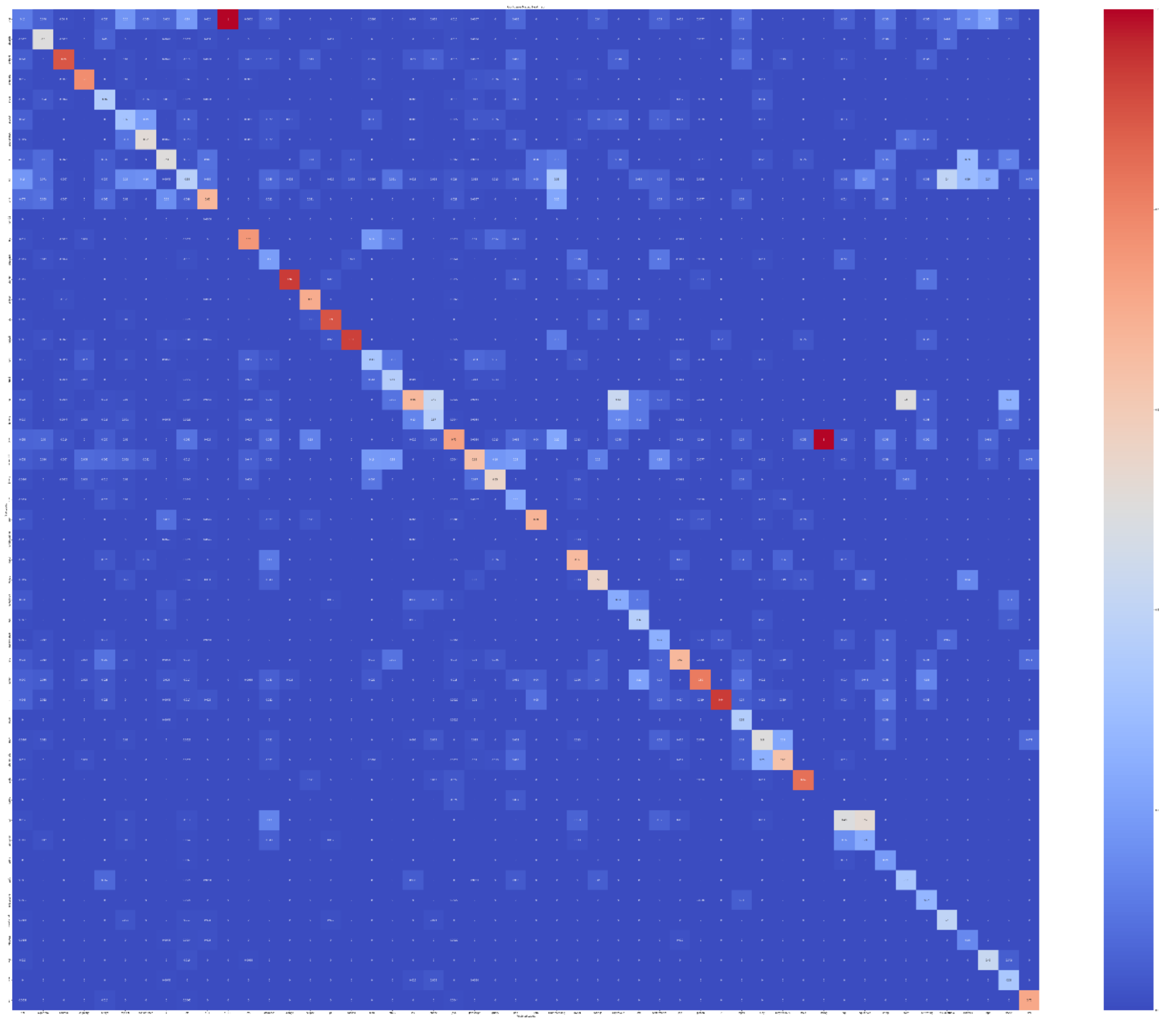
To improve the efficiency and speed of the code, we used the Adam optimization algorithm, which is well-suited for large datasets. We defined the neural network model using the Sequential class from the TensorFlow **Keras library**, and added several dense layers with varying numbers of units and activation functions. The final dense layer used the sigmoid activation function, which is well-suited for multi-label classification tasks like ours.

We compiled the model using the Adam optimizer with a learning rate of **1e-3**, and specified the binary **crossentropy** loss function and several metrics including hamming loss and accuracy to evaluate the performance of the model during training. To prevent overfitting, we used an early stopping callback that monitors the validation loss and stops training if the loss does not improve after a certain number of epochs.

Finally, we fit the model to the training data for 20 epochs and used the validation data to evaluate the performance of the model after each epoch. The resulting **trained model achieved a loss of 0.0071, hamming loss of 0.0024, and accuracy of 0.8084** and as for **the tested model achieved a loss of 0.1165 , hamming loss of 0.0206 and accuracy of 0.5658**. The use of neural networks is becoming increasingly important in the field of natural language processing, and can be a powerful tool for analyzing and modeling text data. Our project demonstrates the effectiveness of using a neural network to perform multi-label classification on preprocessed text data, and the importance of optimizing the model using appropriate algorithms and techniques like Adam optimization and early stopping callbacks. Overall, the combination of TF-IDF feature extraction and neural network modeling allowed us to achieve a high level of accuracy and performance in our text classification task.

Model Visualization : -

As part of our model visualization, we utilized a confusion matrix to visually represent the performance of our neural network model. The confusion matrix is a popular tool used to evaluate the accuracy of classification models, and it allowed us to compare the predicted and actual labels. We used the confusion matrix to create heatmaps,



which helped to highlight the areas where the model performed well and where it struggled. By analyzing this visualization, we gained valuable insights into the strengths and weaknesses of our model and were able to identify opportunities for further improvement.