Team Quadruple

Introduction

Our aim is to predict the browser node ID for Amazon products, so we used the NLP approach to solve the problem.

Libraries and Modules

Major libraries imported were **Pandas** for data manipulation, **NumPy** for mathematical manipulation, **Scikit-learn** for importing transformers and model structures, **nltk,** and **re** for cleaning and modifying text data.

Importing Data

Data was pretty huge with around ~3M rows in train and ~0.1 M in test data. The import was done using pandas and then the train data was distributed into batches

Functions

We used three custom functions in our notebook:

1. **create_product(DataFrame)**
   After analyzing the data, we concluded that details in Title, Description and Bullet Points are clashing so, we concluded merging the Title, Description, and Bullet points into one single article which can be tokenized and used for training, so this function creates a new column "Products" using the existing 3 columns
   Also, we replaced the Null values in Title, Description, and Bullet Points with the empty string("")

2. **clean_data(data)**
   This is the most important helper function in the notebook, it is used for cleaning the text data as it is important to send uniform data to our model for better training. This function lowercase all the text removes unnecessary punctuations and whitespaces, stems the words to their basic form, and removes stop words.

3. **create(index, model):**
   This is a testing function that takes an index as an input. It runs the model over the test data and predict browse_node_ids and creates a CSV file of labels and predictions

Stratified Data sampling

We had a HUGE!! Dataset (~ 3M rows). So, instead of training the whole Dataset, we used Stratified sampling of the given Data. It gave us a sample Dataset that best represents the entire Dataset under study. For this, we used Scikit-Learn's StratifiedShuffleSplit class.

Cleaning

From stratified sampling, we got the required representative training data, so before applying any Machine Learning algorithm on this data we need to first clean it. For cleaning the main text data containing column('Product') we used the clean_data function on training as well as testing data.

Models used

To make the classifier, we can create a pipeline of all these estimators and transformers we want to use. To apply any machine learning model to the text data we have to first convert it into numerical representation, for that we used **Count Vectorizer** and **TfidfTransformer.**
Now, the numeric vectors are given as input machine algorithms. We applied MultinomialNB, SGD, and LinearSVC, to the chunk of data and checked their accuracy.
We found that **LinearSVC** was having the highest accuracy among all three estimators.

Training and Validation

We'll fit our model over the training data's Products_clean column and training data Browse_node_ids column.

Testing and Final Result

On testing data, our model accuracy came out to be **64.14%**, which came to be increased by the increasing complexity of our transformers and estimators. Trying the model with a different set of hyperparameters and then comparing best among them. Using state-of-art Transformers models, deep learning models, but all took so much time and memory to train.