REAL TIME SPAM EMAIL CLASSIFICATION USING SPARK STREAMING

Team: BD_027_270_278_505

1. Design details

- **Reading and parsing of streaming data**
  Read the incoming stream of data and parse the json data to create a dataframe. Several command line arguments are accepted for deciding if the data is for training or testing, hash size, type of model to be trained and type of preprocessing to be used etc.
- **Preprocessing**
  We combine the subject and the body of mails into a single column and also remove punctuations, stop words and convert the words into vectors of fixed length using a hashing vectorizer. Convert the values of the target column to numeric.
- **Model building**
  Using the preprocessed data we build the model incrementally. We split the data into **features and labels** and train the model specified in the command line arguments incrementally using models imported from **sklearn library**. For clustering, models such as Kmeans and Birch are used. Also, a custom K Means model is used for clustering in an endless stream.
  After the training is done we save the model in a pickle file.
- **Testing the model**
  We read the testing data as we did for the training data and then preprocess the data, the same way as we did for the training data. We then evaluate our saved model on the training data, calculate the metrics for each batch of testing data and also keep track of the metrics globally for the entire dataset. Finally at the end of the stream we print the metrics for the entire test data.

2. Surface Level Implementation details

- **Reading and parsing of streaming data**
  - ❖ Set up a spark streaming context object to start receiving the data stream from the tcp connection with streamer file.
  - ❖ Parse each **RDD**,which is in **JSON** format, to create a spark DataFrame which is later passed on to various stages of the process.
- **Data Exploration**
  - ❖ Data is explored to find the count of characters in each mail, count of Ham and Spam mails, average length of ham and spam mails for each batch.
  - ❖ Appropriate plots are drawn to gather inferences.

- **Preprocessing**
  - ❖ **“Body” and “Subject”** of each mail are **concatenated** together to form a **“data”** column. Also, length of each mail is calculated, which could possibly play a role in classification.
  - ❖ The Sentences are tokenized by splitting based on the regex pattern **‘[\\p{Punct}\\s]+’**, which basically splits based on any number of punctuations and spaces.
  - ❖ All the **Stop Words** are removed from the ‘data’ column.
  - ❖ The classes, Ham and Spam are encoded into their respective numeric values 0 and 1.
  - ❖ Next, the data is fed into 2 types of preprocessing which are selected, one is the frequency-based method called **TF-IDF** and another is the Word Embedding Model called **Word2Vec.**
  - ❖ In the TF-IDF method, TF step creates a hashmap based on occurence of each word and IDF calculates the importance of each word and in the Word2Vec method, the model creates a vector representation for a given set of words.

- **Model building**
  1. **Classification**
  - ❖ Various linear and non-linear models such as **Naive Bayes, Support Vector Machines, Logistic Regression, ANN and Passively Aggressive Classifiers** are incrementally trained on each batch using **partial_fit.**
  - ❖ Hyper parameter tuning is performed on each model with a set of most plausible hyperparameters and the best one is found based on the metrics.
  2. **Clustering**
  - ❖ Dimensionality Reduction is performed on the dataset using **Truncated SVD** (as it is a sparse dataset) to 10 dimensions which retains about 80-85% of variance on an average.
  - ❖ Data is fed into clustering models such as **KMeans and Birch**, without the labels as its unsupervised learning.
  - ❖ A **Custom KMeans** model is built from scratch so that it would give back the difference in distance between old and new sets of clusters for each batch, i.e., to support **endless streaming**.
  - ❖ At the end of the streaming, the trained models are stored locally by pickling.

- **Testing the model**
  - ❖ Finally, the stored models are loaded and evaluated based on metrics such as **Accuracy, Confusion Matrix, Precision, Recall and F1 Score** with their respective comparison plots.

3. Reason behind design decisions

- We chose to accept several parameters as command line arguments so that the flow of the entire project could be controlled by just a flag along with a value.

- Implemented all kinds of models which supported incremental learning from sklearn library to find the best model possible for our dataset.
- We began with a regular preprocessing method called TF-IDF, which are generally used in spam classifiers and then moved on to compare it with Word2Vec embedding model.

4. Take Away from the project

- Got to know about real time or streaming data processing and making predictions for the same by building models incrementally.
- Also, could perceive the importance of spark in handling streaming data providing fault tolerance and along with it an abstraction of rdd as Spark Dataframe really eased the process.
- Got a glimpse of how Real World Spam Classifiers are trained and put into work for making predictions.
- Understood that there is only a fine line difference between supervised and unsupervised learning.