Deploying auto-reply Twitter handle with Kafka, Spark and LSTM

Business Overview

Natural language processing (NLP) is a field of artificial intelligence (AI) that focuses on enabling computers to interpret text and spoken words in the same manner that humans do. Several NLP jobs help the computer understand what it imbibes by breaking down human text in ways that make sense to it.
One such task is social media sentiment analysis, which obtains hidden data insights from social media platforms. Natural language processing (NLP) has become an indispensable commercial tool. Sentiment analysis extracts attitudes and emotions in reaction to products, promotions, and events by analyzing the language used in social media postings, comments, and reviews. Companies may utilize this data for various purposes, including product design, customer service, and advertising campaigns. These tools can be clubbed with NER (named entity recognition), a technique for recognizing words or sentences as valuable entities. The whole process can be automated and deployed on the cloud using data ingestion and streaming services with distributed processing.
The objective of this Twitter Support project is to listen to live tweets with required tags and publish them to a Kafka topic which will be consumed using Spark Stream and passed through an NLP pipeline for further processing and replying. The tweets will then be classified based on sentiment and query category using machine learning models like LSTM before responding to them with a custom ticked ID using Flask and tweepy API.

Aim

This project automatically replies to query-related tweets with a trackable ticket ID generated based on the query category predicted using deep learning models. The whole process is automated with tweepy API and Big Data techniques with final deployment on AWS.

Data Description

For the training purposes in this project, we have used an “airline tweets” dataset, which includes necessary fields like airline names as tags, actual text, sentiment class, i.e., positive, negative, or neutral and a topic class which can be one of these:

1. Baggage Issue
2. Customer Experience
3. Delay and Customer Service
4. Extra Charges
5. Online Booking
6. Reschedule and Refund
7. Reservation Issue

8. Seating Preferences

Tech Stack

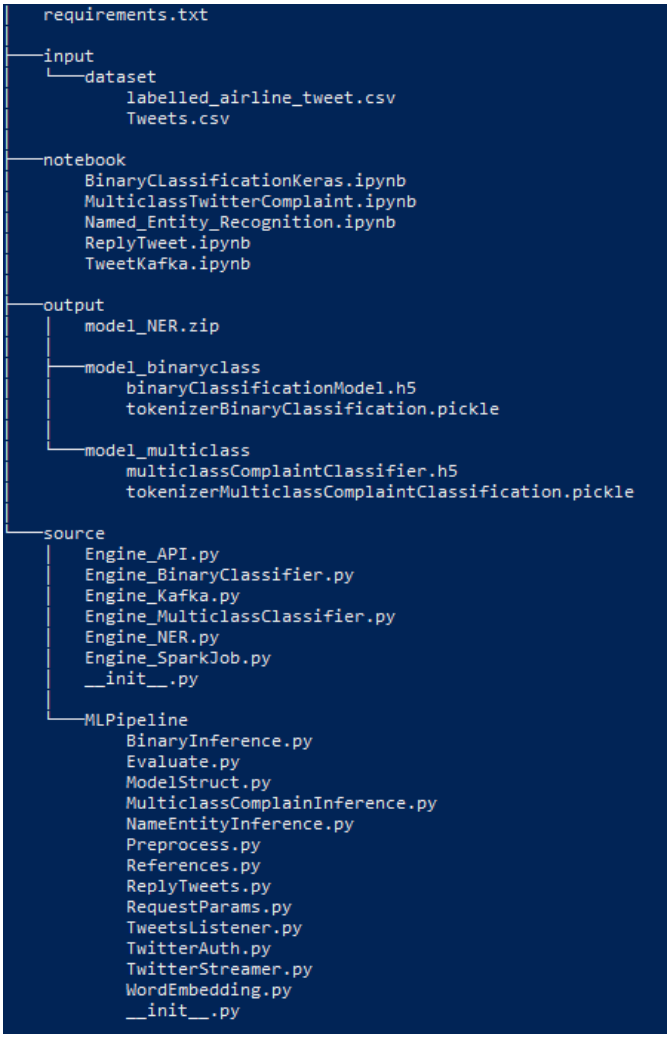
➔ Language: Python3
➔ Services: Confluent Kafka, Spark
➔ Libraries: tweepy, Flask, kafka, spacy, sklearn, keras, numpy, pyspark, nltk, matplotlib, os

● Tweepy for fetching tweets and replying.
● Flask for setting up API for a reply.
● Kafka for data ingestion.
● Sklearn for model evaluation and encoding.
● Nltk for preprocessing.
● Spacy for named entity recognition.
● Pandas for data manipulation and analysis.
● Matplotlib for plotting graphs.
● Numpy for array and math operations.
● Tensorflow and Keras for loading and building models and Embeddings.
● Pyspark for UDFs and Streaming.
● Os for operating folders.

Approach

- Dataset is first processed by removing Stopwords, mentions, URLs, etc., and Lemmatization.
- Tokenizing and Embedding words to text sequences.
- A sequential model is trained on the processed data, which includes LSTM for Sentiment Classification.
- Topic labeling using Latent Dirichlet Allocation (LDA).
- Data is trained on a sequential model including LSTM for Topic Classification.
- Named Entity Extraction.
- Saving all models and embeddings for runtime usage.
- Sink enriched data to a parquet file.
- Reply to the tweet using Flask and tweepy.

Modular code structure



Part 1: Engines

File Name: Enginer_BinaryClassifier.py
File Description: File to train and infer sentiment analysis.

File Name: Enginer_MulticlassClassifier.py
File Description: File to train and infer Complain classification.

File Name: Enginer_Kafka.py
File Description: File to push tweets to kafka topic.

File Name: Enginer_NER.py
File Description: File to train and infer Name Entities.

File Name: Enginer_SparkJob.py
File Description: Main file to create Spark job and embedded using UDF.

Part 2: Helpers

File Name: BinayInference.py
File Description: Helper class to infer sentiment analysis.

File Name: Evaluate.py
File Description: Helper class to evaluate model performance.

File Name: ModelStruct.py
File Description: Helper class to structure models and save them.

File Name: MulticlassComplainInference.py
File Description: Helper class to classify categories.

File Name: NameEntityInference.py
File Description: Helper class to infer name entities.

File Name: Preprocess.py
File Description: Helper class to preprocess text.

File Name: References.py
File Description: Helper class for all the constants and URL

File Name: ReplyTweets.py
File Description: Helper class to reply to tweets

File Name: RequestParams.py
File Description: Helper class to send a request to API

File Name: TweetsListener.py
File Description: Helper class to structure twitter input

File Name: TwitterAuth.py
File Description: Helper class to authenticate the Twitter app

File Name: TwitterStreamer.py
File Description: Helper class to stream tweets

File Name: WordEmbedding.py
File Description: Helper class to convert text to sequences

Key Takeaways

- Understanding the project and how to use AWS EC2 Instance
- Visualizing the complete Architecture of the system
- Exploring Tweepy API and dataset.
- Installing Kafka and using it for creating topics
- Installing Spark and using it for UDFs and Streaming
- Understanding the basics of text classification and its applications.
- Understanding the basics of NER.
- Data cleaning and preparation procedure for NLP tasks using regex.
- Understanding LDA and data labeling.
- Implementing Flask API.
- Understanding the basics of Spark and UDFs.
- Understanding the basics of Kafka.
- Exploring TensorFlow and Keras libraries.
- Making use of pre-trained models.
- Text Embedding for NLP tasks.
- Use of Google Colab for training purposes.

Architecture Diagram