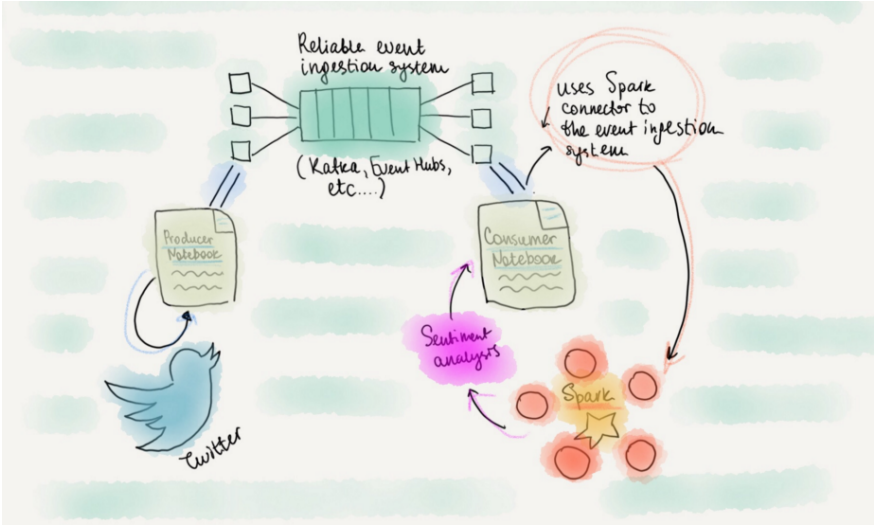
Streaming Sentiment Analysis Training and Deployment



General Pipeline for The Application

As a Data Scientist, in the era of continuous data generation, you need to process data in real-time. A traditional data processing system such as the **E**xtract **T**ransform **L**oad (ETL) process is only adequate for the processing of static data. Traditional ETL tools process data in batches and are not capable of handling real-time data, but only data that is already stored in a database system.

For the processing of real-time data, we need systems that support stream processing, such as Apache Spark. Those processing systems provide the option for continuous computations, as data is continuously flowing through them. Examples of such functionalities are the regular cleaning and aggregation of the incoming data before storage.

Apache Spark provides two ways of working with streaming data; **Spark Streaming** and **Spark Structured Streaming**. In this tutorial, we use Spark Structure Streaming because is more inclined towards real-time stream processing. Read this article for a comparison between the two different ways of working.

The Goal

In this tutorial, we want to explore different models from different families like classical machine learning, RNN and its variants, and Transformer-Based models (BERT). We also want to explore different methods to encode the sentence. Then we will deploy the best model on a spark pipeline that gets tweets from Twitter API relevant to a specific keyword and classifies its sentiment (Positive, Negative, or Neutral).

Training Models

For training, we used the Sentiment140 dataset with 1.6 million tweets dataset that exists on Kaggle. Unfortunately, the dataset consists of two labels only as opposed to what they stated in the dataset description. For all models, we did a sequence of preprocessing steps on the text. We removed mentions and URLs but left the hashtags. Then, we removed all non-alphabet characters and converted the string to lowercase. Finally, we split the string into tokens on whitespace.

Classical Models

We used a Logistic Regression, LinearSVC, and MultinomialNB. For the sentence representation, we used TF-IDF with 1-to-4 N-Grams and a feature vector of size 50,000. We used scikit-learn to train these models.

RNN and Its Variants

First of all, to represent tokens we used a learnable embedded matrix of size 10,501 tokens (10,000 for the most frequent tokens, 500 for out of vocab buckets, and one for '<pad>' token) and a feature vector of size 200.

The first explored model consists of an embedding layer (with size previously mentioned), 2 bidirectional GRUs each with 100 units, a GRU layer with 200 units, and finally output layer with 1 unit and with a sigmoid activation function of course.

The second explored model consists of an embedding layer (with size previously mentioned), 2 bidirectional LSTMs each with 100 units, an LSTM layer with 200 units, and finally output layer with 1 unit and again with a sigmoid activation function.

Keras and Tensorflow were used to train these models.

BERT-BASE-Uncased

We fine-tuned the BERT model for 2 epochs only using AdamW optimizer with a learning rate of 1e-5. PyTorch is used to train this model.

Results

As we might expect fine-tuned BERT achieved the highest accuracy among all the models. Then, LSTMs and GRUs come next and the classical models come in the end. All results are measured on the validation set.

Model	Accuracy
BERT-BASE-Uncased	86.3
GRUs	83.6
LSTMs	83.3
Logistic Regression	79.2
Linear SVC	78.8
MultinomialNB	77.5

. . .

Deploy BERT in Spark Pipeline

We present an end-to-end architecture on how to stream data from Twitter, clean it, pass it to RESTful API, and apply a sentiment analysis model to detect the polarity of each tweet.

The Architecture

We create a TCP socket between Twitter's API and Spark, which waits for the call of the Spark Structured Streaming and then sends the Twitter data. After that, we receive the data from the TCP socket and preprocess it with the PySpark library, which is Python's API for Spark. Then, we pass the data to RESTful API where the model is located and apply sentiment analysis using fine-tuned BERT-Base-Uncased and return Positive, Negative, or Neutral. Finally, we save the tweet and the sentiment analysis polarity in a parquet file.

. . .

Assumptions

One side note regarding model predictions, since the training data only contains Positive and Negative labels, we modeled the task as binary classification. Then, we assume that the model is uncertain about the tweet is positive or negative that means it's probably Neutral. In mathematical terms, when the model predicts probability between 0.4 and 0.6, we assume that it's Neutral.

. . .

Suggested Enhancements

- Fine-tune RoBERTa instead of BERT which outperforms BERT by 2–20%.
- Get data that contain Neutral labels.
- Automate running the scripts instead of running them manually.
- Make a frontend with a dashboard and visualization which enhances user experience