Assignment 3 - Hashtag Prediction on Live Stream of Tweets

Luka Beverin[1], **Sanya Anand**[1], **Prerna**[1], **and Dishani Sen**[1]

[1]KU Leuven, Master of Statistics and Data Science, Advanced Analytics in Business

Introduction

> Twitter is a great place to tell the world what you're thinking before you've had a chance to think about it.
>
> *Chris Pirillo*

The goal of this assignment is to develop a model that predicts the hashtags of a tweet in real time. Each tweet contains one of the following hashtags as it's label: #vaccine, #stopasianhate, #covid, #china, #inflation and #biden. All other hashtags that might have been used in the tweet are removed. In the past many machine learning models have been in use to perform text classification and prediction on tweets, however, it's rare that predictions are performed on a live stream of tweets.

In this assignment, we split the task into two parts. In the first part we collect twitter data using Apache Spark and then make use of popular python libraries such as pandas and Sklearn to explore the data and investigate the performance of various prediction models. The second part is built upon the first by implementing the best performing model in Spark ML to predict twitter hashtags in real time.

A logistic regression using term frequency-inverse document frequency (tf-idf) features is used to predict the hashtags of incoming tweets. The model was trained on data which was collected on our local computer using Apache Spark. The entire machine learning pipe has been implemented using the Spark ML, from collecting the data, pre-proccesing the data, training the model, and finally predicting hashtags as they arrive.

Data Collection

We constructed a historical data set using the spark streaming service. By using the example code from file spark_streaming_example_saving, the textual data from steaming server "seppe.net:7778" is collected and saved locally. The streaming server is running at seppe.net:7778. The stream is text-based with each line containing one tweet (one instance) formatted as a JSON dictionary with the following keys (features): "tweet_id" (id of the tweet, not to be used), "tweet_text" (the tweet with the hashtags blanked out) and "label" (the target). The data collected through the stream was then converted into a comma separated value file for ease of use. That is our historical data set which has been used for training the prediction model. We collected approximately 5000 tweets.

Data Preprocessing

Text data from twitter is known to unstructured and disorganized. For this reason, it is essential that the data is cleaned and pre-processed immediately, as this is the first and foremost step of any machine learning task. It's crucial to preprocess the data into a readable format that can be later on used for textual representations or word embeddings, which are utilized by algorithms and machine learning models. In the context of twitter text data, this involves lower casing the text, removing stop-words, removing unicode characters, removing terms like hashtags, mentions, links, etc.

Lower casing the text. It's recommended to lowercase all of the characters of the tweet text first. The reason is to avoid any case-sensitive process or influence later on in the machine learning pipeline. For example, our model might treat a word which is in the beginning of a sentence with a capital letter different from the same word which appears later in the sentence but in lowercase.

Removing Unicode Characters. The other hashtags (excluding the "label") used in the text of the tweets have been blanked out, which is an ASCII format. We remove them to get the clean tweet texts.

Removing terms like mentions, hashtags, links, and more. We must remove several terms including mentions, hashtags, links, punctuation, etc. to obtain a cleaner version of the tweet text. A python library named `re`, which stands for regular expression is used to help us on this task. A Regex, or regular expression, is a special string that contains a pattern that can match words associated with that pattern.

Removing Stop Words. Stop words are commonly used words that have no significant contribution to the meaning of the text, such as "the", "a", "an" and "in". Therefore, we can remove those words. To retrieve the stop words, we download a corpus from the NLTK library. We then go onto removing all the stop words from our twitter text.

Stemming and Lemmatization. Stemming refers to the removal of suffices, like "ing", "ly", "s", etc. by a simple rule-based approach. Lemmatization is one step ahead of Stemming. Since it transforms the word into its root word rather than only stripping the suffices, lemmatization is a more successful alternative than stemming. To find the root word, it uses the vocabulary and performs a morphological study.

Data Analysis

The maximum number of characters allowed in a single tweet is 140. This cap is partly what makes twitter so interesting, and makes analysing the tweets data even more so. In this section we perform exploratory data analysis of the pre-processed tweets in hopes to gather insights and patterns from tweets. We are also confident that the features and patterns analysed in this section can also be used as input features in a prediction model. For example, we created a new variable called 'length of tweet', which may show to have a relationship with the type of hashtag being posted. Other features that were explored are character count, number of special characters, the average number of characters per word in a tweet, and more.

Number of Words. One of the most basic features we can extract is the number of words in each tweet. The basic intuition is that a more influential or informative tweet will have more number of words than a casual tweet. After pre-processing

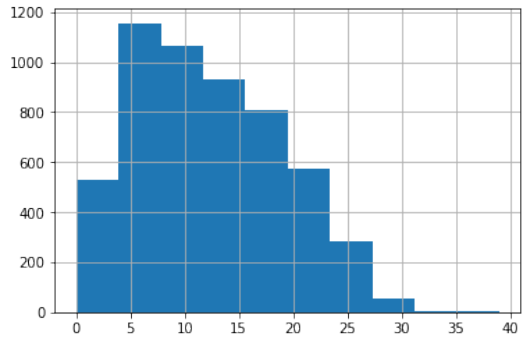

Fig. 1. Number of Words in Tweet Text

the tweets, we can see that most tweets contain 5 to 15 words.

Number of Characters. This feature is based on the previous feature. Here, the number of characters in each tweet is calculated. A similar pattern is seen compared to the number of

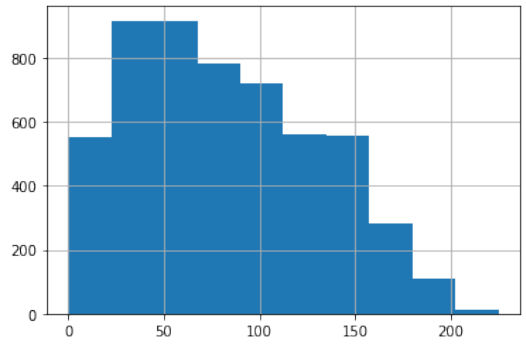

Fig. 2. Number of characters in tweet text

words in the tweet.

Average Character Length. Another feature that we extracted is the average character length that measures the average word length of each tweet. This may also show to be important to various models. The average length of words in

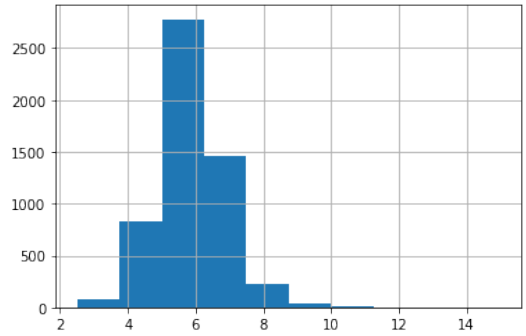

Fig. 3. Average word length

a tweet is roughly 5 characters long.

Number of Stop Words. Generally, while solving an NLP problem, the first thing we do is to remove the stop words. Even though we have removed them in our analysis as well, we calculated the number of stop words in the hope that it will also give us some extra information which we might need to improve our model.

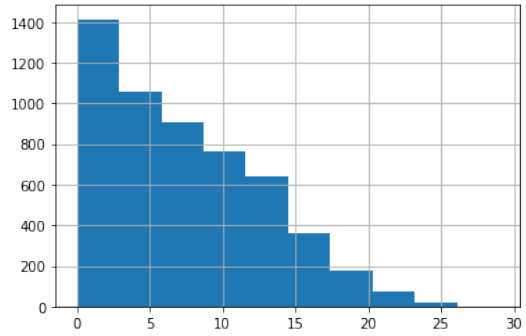

Fig. 4. Number of Stop Words in Tweet Text

Number of Numerics. Just like we calculated the number of words, we can also calculate the number of numerics which are present in the tweets. As is shown on the plot below, not many tweets contain numbers, and we do not expect a significant relationship between the amount of numerics on a tweet and its hashtag.

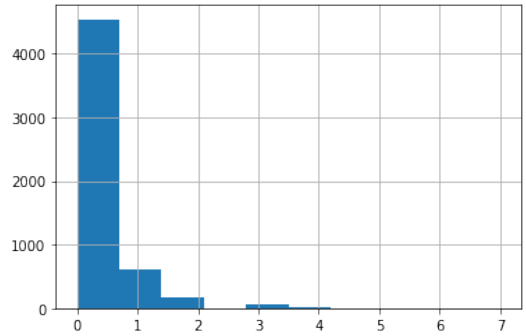

Fig. 5. Number of Numerics in Tweet Text

Text Representation

A common problem with building text prediction models is that the data is chaotic, and machine learning algorithms prefer well defined numeric inputs and outputs. Machine learning algorithms cannot make connections from raw text directly; and for this reason the text must be converted into vector numbers. Up until now we have done all the basic preprocessing steps required to clean our data for the next step, which is extracting features using NLP techniques.

Bag of Words. The Bag of Words (BoW) model is the simplest form of text representation in numbers. Like the term itself, we can represent a sentence as a bag of words vector (a string of numbers). This is based on the assumption that two identical text fields would contain similar types of words and hence have a similar bag of words. Furthermore, we can deduce much about the document's context solely from the text.

The simplicity of BoW that many people appreciate also comes with its own well known drawbacks. The major disadvantage of a BoW model is that the semantic analysis of the sentence is not taken into consideration and the word arrangement is discarded. For example, in the BoW model, the sentence "Red means stop" is represented the same way as "Stop means read" which of course is incorrect.

Term Frequencies. The BoW model is able to differentiate between how many times a word is used, but the issue with scoring word frequency is that highly frequent words start to dominate in the tweet text. These common words may not contain as much useful information as the more rare domain specific words. To overcome this problem we can penalize the very common words by re-scaling the frequency of often they appear in all the tweets.

Term frequency is simply the ratio of the count of a word present in a sentence, to the length of the sentence. TF = (Number of times term T appears in the particular row) / (number of terms in that row). IDF is a measure of how important a term is. We need the IDF value because computing just the TF alone is not sufficient to understand the importance of words.The intuition behind inverse document frequency (IDF) is that a word is not of much use to us if it's appearing in all the documents.

Therefore, the IDF of each word is the log of the ratio of the total number of rows to the number of rows in which that word is present. IDF = log(N/n), where, N is the total number of rows and n is the number of rows in which the word was present.

Term Frequency – Inverse Document Frequency (TF-IDF) is the multiplication of the TF and IDF. TF-IDF also gives larger values for less frequent words and is high when both IDF and TF values are high i.e the word is rare in all the documents combined but frequent in a single document.

Methods

The two main machine learning models for text classification are Logistic Regression and Convolutional Neural Networks (CNN).

Logistic Regression. The logistic regression model is one of the most well known models used in machine learning applications. The model gives the probability of a certain class, usually dichotomous outcome variables which are often labeled "0" and "1". Nevertheless, logistic regression can be extended to model several classes, such as in our case, where we would like to predict several hashtags.

The benefit of a logistic regression model is that it's fairly simple to implement with very few parameters needed to be tuned.

Convolutional Neural Networks. A convolutional neural network is comprised of filters which are designed to find spatial patterns in text documents. They are most commonly used to classify the sentiment (positive or negative) of text data.

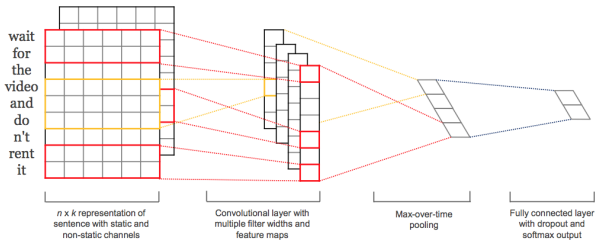

Fig. 6. Example of a CNN for text classification

Results

The best performing model was the logisitc regression with tf-idf features. Running 5 fold cross validation on this model gave us an accuracy score of 0.604. The other models such as logistic regression with BoW and CNN were not far behind with an accuracy score of 59%.

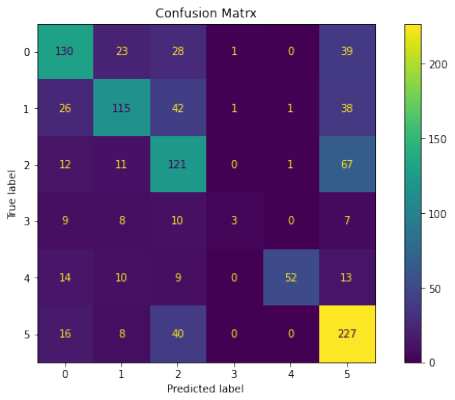

Fig. 7. Confusion Matrix: Logistic Regression + tf-idf features

Spark ML Pipeline

Collect and Save the Tweets. The code below connects to the Spark session in order to collect the required tweets. The tweets are then saved on our local computer and transformed into a pandas Dataframe format.

```
from pyspark import SparkContext
sc=SparkContext("local[*]","PySparkShell")

from pyspark.sql import SparkSession
spark=SparkSession.builder.appName('PySparkShell').getOrCreate()

from threading import Thread

class StreamingThread(Thread):
    def __init__(self, ssc):
        Thread.__init__(self)
        self.ssc = ssc
    def run(self):
        ssc.start()
        ssc.awaitTermination()
    def stop(self):
        print('----- Stopping... this may take a few seconds -----')
        self.ssc.stop(stopSparkContext=False, stopGraceFully=True)

from pyspark.streaming import StreamingContext

ssc = StreamingContext(sc, 2)

lines = ssc.socketTextStream("seppe.net", 7778)
lines.saveAsTextFiles("path")

ssc_t = StreamingThread(ssc)
ssc_t.start()

ssc_t.stop()

data=sc.textFile("path")
df=spark.read.json(data)
```

```
+--------------+-------------------+--------------------+
|         label|           tweet_id|          tweet_text|
+--------------+-------------------+--------------------+
|    #inflation|1392894115793326090|@PeterSchiff @qry...|
|      #vaccine|1392894443426963456|@ReallySwara You ...|
|      #vaccine|1392894360635707398|PARENTS - What to...|
|    #inflation|1392894738995548166|Odds of a lower c...|
|    #inflation|1392894448758108160|Strongest #[redacted]...|
|        #biden|1392894964850331648|Brad and Britt Ca...|
|        #biden|1392894942943580169|"#[redacted] #[redacted]...|
|        #biden|1392894911700144134|⏰Running OUT⏳ #[redacted]...|
|    #inflation|1392895011964997633|#[redacted] ground s...|
|    #inflation|1392895003618385922|#[redacted] #[redacted]...|
|    #inflation|1392895225111011329|We all can use so...|
|    #inflation|1392895201409175563|#[redacted] ground s...|
|        #covid|1392895857033453571|@TomSwarbrick1 if...|
|        #covid|1392895823206359047|Oxygen cylinders ...|
|        #covid|1392895789769412608|This #[redacted] #[redacted]...|
|#stopasianhate|1392895918890962945|#[redacted] #[redacted]...|
|#stopasianhate|1392895867615653891|"you absolutely d...|
|#stopasianhate|1392895866034434049|The audacity of a...|
|        #biden|1392896270990069766|#[redacted]
#[redacted]...|
|        #biden|1392896259242041351|White House moves...|
+--------------+-------------------+--------------------+
only showing top 20 rows
```

Fig. 8. Tweet collection

Data Preprocessing. In the data pre-processing step we remove unneccessary strings with the `RegexTokenizer` and also remove stop words with the `StopWordsRemover` function. The logistic regression model in combination with tf-idf features was the best performing model in the first stage. The `HashingTF` and `IDF` functions creates those feautures. Lastly, we convert the hashtag labels into numeric values using the `StringIndexer`.

```
regexTokenizer = RegexTokenizer(inputCol="tweet_text", outputCol="words", pattern="\\W")

#add custom stop words we want to remove
add_stopwords = ["http","https","amp","rt","t","c","the"]
stopwordsRemover = StopWordsRemover(inputCol="words", outputCol="filtered").setStopWords(add_stopwords)

hashingTF = HashingTF(inputCol="filtered", outputCol="rawFeatures", numFeatures=10000)
idf = IDF(inputCol="rawFeatures", outputCol="features", minDocFreq=5)

label_stringIdx = StringIndexer(inputCol = "label", outputCol = "target")
```

The benefit of using the Spark ML library is that we are able to streamline the data pre-processing stage with the help of pipelines. All the pre-processings steps are defined within the pipeline and then used to transform the data.

```
pipeline = Pipeline(stages=[regexTokenizer, stopwordsRemover, hashingTF,idf, label_stringIdx])

pipelineFit = pipeline.fit(data)
dataset = pipelineFit.transform(data)
dataset.show(1)
```

Training the Model. With the next few lines of code we train the logistic regression model. We split the data into train and test for evaluation purposes and model checking.

```
(trainingData, testData) = dataset.randomSplit([0.7, 0.3], seed = 100)

lr = LogisticRegression(labelCol="target",family = "multinomial", featuresCol="features")

lrModel = lr.fit(trainingData)
predictions = lrModel.transform(testData)
```

Saving the Model. The model is then saved to our local computer for use later on.

```
lrModel.write().overwrite().save("path")
```

Making Predictions. The crux of this task, to make twitter hashtag predictions in real time. The function below uses the model we had saved earlier to make predictions on data as it is being received. Note that the data pre-processing is being utilised here because when new data comes in it is unstructured and needs to be in the same format as what was used to train the model.

```
reloaded_model = lrModel.load("path")
globals()['models_loaded'] = False
globals()['my_model'] = None

def process(time, rdd):
    if rdd.isEmpty():
        return

    print("========= %s =========" % str(time))

    # Convert to data frame
    df = spark.read.json(rdd)
    df.show()

    #transform new incoming data
    dataset = pipelineFit.transform(df)

    # Load in the model if not yet loaded:
    if not globals()['models_loaded']:
        # load in your models here
        globals()['my_model'] = reloaded_model
        globals()['models_loaded'] = True

    # And then predict using the loaded model:
    df_result = globals()['my_model'].transform(dataset)
    df_result = df_result.select("label","tweet_id","tweet_text","target","prediction")

```

```
    df_result.show()
```

```
ssc = StreamingContext(sc, 10)

lines = ssc.socketTextStream("seppe.net", 7778)
lines.foreachRDD(process)

ssc_t = StreamingThread(ssc)
ssc_t.start()
```

Below is a snippet of our model working in real time. The target column is the ground truth value of the hashtag. The prediction column is the predicted hashtag by our model. In the first instance our model predicted #vaccine correctly, but did not do so well on the other two. In the future we would like to transform the target column back into the original hashtag so that the predictions are easier to read and understand.

```
========= 2021-05-22 18:03:00 =========
+--------+-------------------+--------------------+
|   label|           tweet_id|          tweet_text|
+--------+-------------------+--------------------+
|#vaccine|1396131648882630659|Why would you get...|
+--------+-------------------+--------------------+

+--------+-------------------+--------------------+------+----------+
|   label|           tweet_id|          tweet_text|target|prediction|
+--------+-------------------+--------------------+------+----------+
|#vaccine|1396131648882630659|Why would you get...|   0.0|       0.0|
+--------+-------------------+--------------------+------+----------+

========= 2021-05-22 18:03:10 =========
+------+-------------------+--------------------+
| label|           tweet_id|          tweet_text|
+------+-------------------+--------------------+
|#covid|1396132504395001857|#[redacted] #[redacted]...|
+------+-------------------+--------------------+

+------+-------------------+--------------------+------+----------+
| label|           tweet_id|          tweet_text|target|prediction|
+------+-------------------+--------------------+------+----------+
|#covid|1396132504395001857|#[redacted] #[redacted]...|   2.0|       5.0|
+------+-------------------+--------------------+------+----------+

========= 2021-05-22 18:03:20 =========
+------+-------------------+--------------------+
| label|           tweet_id|          tweet_text|
+------+-------------------+--------------------+
|#covid|1396132308025962497|@IndiaTVHindi @in...|
+------+-------------------+--------------------+

+------+-------------------+--------------------+------+----------+
| label|           tweet_id|          tweet_text|target|prediction|
+------+-------------------+--------------------+------+----------+
|#covid|1396132308025962497|@IndiaTVHindi @in...|   2.0|       0.0|
+------+-------------------+--------------------+------+----------+
```

Fig. 9. Real time Twitter hashtags predictions