First GOP Debate Twitter Sentiment

Data Preparation

Data is downloaded from

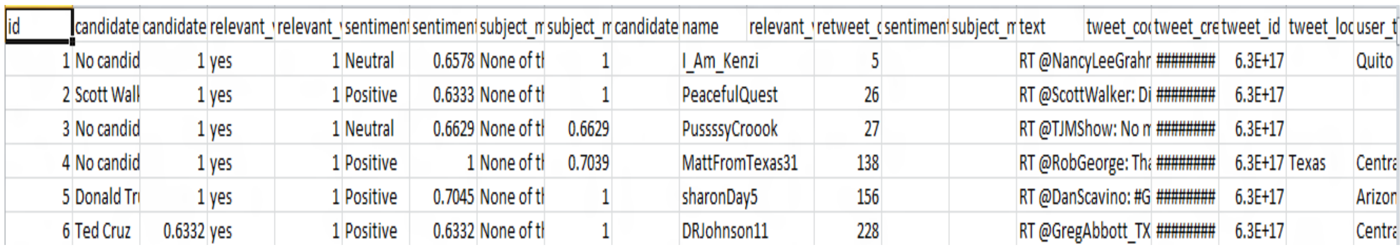
https://www.kaggle.com/crowdflower/first-gop-debate-twitter-sentiment

It contains 14k tweets.

It contains the following fields:

1. id
2. candidate
3. candidate_confidence
4. relevant_yn
5. relevant_yn_confidence
6. sentiment
7. sentiment_confidence
8. subject_matter
9. subject_matter_confidence
10. candidate_gold
11. name
12. relevant_yn_gold
13. retweet_count
14. sentiment_gold
15. subject_matter_gold
16. text

17. tweet_coord
18. tweet_created
19. tweet_id
20. tweet_location
21. user_timezone



id	candidate	candidate	relevant_	relevant_	sentiment	sentiment	subject_m	subject_m	candidate	name	relevant_	retweet_c	sentiment	subject_m	text	tweet_co	tweet_cre	tweet_id	tweet_loc	user_t
1	No candid	1	yes	1	Neutral	0.6578	None of tl	1		I_Am_Kenzi		5			RT @NancyLeeGrahr		########	6.3E+17		Quito
2	Scott Wall	1	yes	1	Positive	0.6333	None of tl	1		PeacefulQuest		26			RT @ScottWalker: Di		########	6.3E+17		
3	No candid	1	yes	1	Neutral	0.6629	None of tl	0.6629		PussssyCroook		27			RT @TJMShow: No m		########	6.3E+17		
4	No candid	1	yes	1	Positive	1	None of tl	0.7039		MattFromTexas31		138			RT @RobGeorge: Th		########	6.3E+17	Texas	Centra
5	Donald Tr	1	yes	1	Positive	0.7045	None of tl	1		sharonDay5		156			RT @DanScavino: #G		########	6.3E+17		Arizon
6	Ted Cruz	0.6332	yes	1	Positive	0.6332	None of tl	1		DRJohnson11		228			RT @GregAbbott_TX		########	6.3E+17		Centra

So firstly we have create schema:

```
val schema = StructType(Array(StructField("id", IntegerType, true),
                    StructField("candidate", StringType, true),
          StructField("candidate_confidence", DoubleType, true),
                    StructField("relevant_yn", StringType, true),
          StructField("relevant_yn_confidence", DoubleType, true),
                 StructField("sentiment", StringType, true),
             StructField("sentiment_confidence", DoubleType, true),
                    StructField("subject_matter", StringType, true),
       StructField("subject_matter_confidence", DoubleType, true),
                    StructField("candidate_gold", StringType, true),
                    StructField("name", StringType, true),
                    StructField("relevant_yn_gold", StringType, true),
                    StructField("retweet_count", DoubleType, true),
                    StructField("sentiment_gold", StringType, true),
                  StructField("subject_matter_gold", StringType, true),
                    StructField("text", StringType, true),
                    StructField("tweet_coord", StringType, true),
                    StructField("tweet_created", StringType, true),
                    StructField("tweet_id", StringType, true),
                    StructField("tweet_location", StringType, true),
                    StructField("user_timezone", StringType, true)
                       ))
```

Now, we load the data in a dataframe using above schema:

```
val df = spark.read.
          format("csv")
          .schema(schema)
          .option("header","true")
          .load("/home/harsh/Desktop/gop debate
twitter/Sentiment.csv")
```

Selecting only those columns which are useful:

```
val df1 = df.select("candidate","candidate_confidence",
"relevant_yn", "relevant_yn_confidence", "sentiment",
"sentiment_confidence", "subject_matter",
"subject_matter_confidence", "retweet_count", "text", "tweet_coord")
```

Combining text (String) columns into a single column:

```
val o = df1.select( concat($"candidate", lit(" "), $"subject_matter",
lit(" "), $"text" ).alias("text"), $"candidate_confidence",
$"relevant_yn", $"relevant_yn_confidence", $"sentiment",
$"sentiment_confidence", $"subject_matter_confidence",
$"retweet_count" )
```

Removing records having null values:

```
val p = o.na.drop
```

Column “relevant_yn” consist of two string values either yes or no. To convert its values in numerical values, we create a udf:

```
def lo(i:String) : Double = { if(i.equals("no")){0} else{1} }

val rel = udf(lo _)
```

Column “sentiment” consist of three string values Negative, Nuetral, Positive. To convert its values in numerical values, we create a udf:

```
def pr(i:String) : Double = { if(i.equals("Negative")){0}
                          else if(i.equals("Neutral")){1}
                          else{2}
                         }

val sent = udf(pr _)
```

For pre-processing (removing website links, emojis, special characters, unnecessary spaces) we create a udf :

```
def prep(d:String) :String = { d.replace("\"","").toLowerCase()
```

```
    .replaceAll("\n", "")
    .replaceAll("rt\\s+", "")
    .replaceAll("\\s+@\\w+", "")
    .replaceAll("@\\w+", "")
    .replaceAll("\\s+#\\w+", "")
    .replaceAll("#\\w+", "")
    .replaceAll("(?:https?|http?)://[\\w/%.-]+", "")
    .replaceAll("(?:https?|http?)://[\\w/%.-]+\\s+", "")
    .replaceAll("(?:https?|http?)//[\\w/%.-]+\\s+", "")
    .replaceAll("(?:https?|http?)//[\\w/%.-]+", "")
    .replaceAll("[^\\u0000-\uFFFF]","")
    .replaceAll("(\u00a9|\u00ae|[\u2000-\u3300]|\ud83c[\ud000-
\udfff]|\ud83d[\ud000-\udfff]|\ud83e[\ud000-\udfff])","")
    .trim()
}


val preProcess = udf(prep _)
```

Applying all udfs on columns:

```
val data = p.select(preProcess($"text" ).alias("text"),
$"candidate_confidence", rel($"relevant_yn").alias("relevant_yn"),
$"relevant_yn_confidence", sent($"sentiment").alias("label"),
$"sentiment_confidence", $"subject_matter_confidence",
$"retweet_count")
```

Then to use nlp stemmer, we need to text have to go through document, then through token, then through normalizer, then through stemmer, then finally through finisher:

```
val document = new DocumentAssembler()
    .setInputCol("text")
    .setOutputCol("document")

val d1 = document.transform(data)

val token = new com.johnsnowlabs.nlp.annotator.Tokenizer()
    .setInputCols("document")
    .setOutputCol("token")

val t1 = token.fit(d1).transform(d1)

val normalizer = new Normalizer()
    .setInputCols("token")
    .setOutputCol("normal")

val n1 = normalizer.fit(t1).transform(t1)

val stemmer = new Stemmer()
```

```
    .setInputCols("normal")
    .setOutputCol("stem")

val s1 = stemmer.transform(n1)

val finisher = new Finisher()
                .setInputCols("stem")
                .setOutputCols("final")

val f1 = finisher.transform(s1)
```

After loading, we need to convert tweets into feature vectors.

```
val hashingTF = new HashingTF()
.setInputCol("final").setOutputCol("rawFeatures").setNumFeatures(10000)

val featurizedData = hashingTF.transform(f1)

val idf = new IDF().setInputCol("rawFeatures").setOutputCol("features")

val idfModel = idf.fit(featurizedData)
val rescaledData = idfModel.transform(featurizedData)
```

Then we need to assemble all numeric columns and categorical feature vectors to form final Features vector:

```
val assembler = new VectorAssembler()
  .setInputCols(Array("candidate_confidence", "relevant_yn", "sentiment_confidence","subject_matter_confidence","retweet_count", "features"))
  .setOutputCol("finalFeatures")


val output = assembler.transform(rescaledData)
```

Extracting label and Final Features (only they are needed to proceed further)

```
val limited  = output.select($"label",$"finalFeatures".alias("features"))
```

Then we split the transformed data into two subsets i.e. training and test(ratio 0.8:0.2)

val Array(training, test) = limited.randomSplit(Array[Double](0.8,0.2))

Model Selection and Model Tuning

We tried Logistic Regression for classification.

```
val lr = new LogisticRegression().setMaxIter(10).setRegParam(0.01).setLabelCol("label").setElasticNetParam(0.5)


val model = lr.fit(training)
val preTr = model.transform(training)

val preTs = model.transform(test)
```

Conclusion

We evaluated accuracy for model using MultiClassClassification Evaluator and got 75 % accuracy for training and 68 % for testing.

```
val evaluator = new MulticlassClassificationEvaluator()
                .setLabelCol("label")
                .setPredictionCol("prediction")
                .setMetricName("accuracy")


val train_accuracy = evaluator.evaluate(preTr)

val test_accuracy = evaluator.evaluate(preTs)
```