In [1]:
```
import pyspark

sc = pyspark.SparkContext(appName="text")
```

In [2]:
```
sc
```

Out[2]:
SparkContext

Spark UI (http://192.168.1.6:4040)
Version
v2.3.1
Master
local[*]
AppName
text

In [3]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
        .master("local").appName("text").getOrCreate()
```

Looking at the data

The dataset can be downloaded from the below link.
http://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip
(http://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip)

By looking at the description of the dataset from the link, the information on each field can be found. 0 - the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive) 1 - the id of the tweet (2087) 2 - the date of the tweet (Sat May 16 23:58:44 UTC 2009) 3 - the query (lyx). If there is no query, then this value is NO_QUERY. 4 - the user that tweeted (robotickilldozr) 5 - the text of the tweet (Lyx is cool)

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [7]:
```
plt.style.use('fivethirtyeight')
%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

In [8]:
```
cols = ['sentiment','id','date','query_string','user','text']
```

In [10]:
```
df = pd.read_csv("../Dropbox/pj_ss/tweet/training.1600000.processed.noem
oticon.csv",
                 header=None, names=cols, engine='python')
```

In [11]:
```
df.head()
```

Out[11]:

	sentiment	id	date	query_string	user	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1600000 entries, 0 to 1599999
Data columns (total 6 columns):
sentiment       1600000 non-null int64
id              1600000 non-null int64
date            1600000 non-null object
query_string    1600000 non-null object
user            1600000 non-null object
text            1600000 non-null object
dtypes: int64(2), object(4)
memory usage: 73.2+ MB
```

In [13]:
```
df.sentiment.value_counts()
```

Out[13]:
```
4    800000
0    800000
Name: sentiment, dtype: int64
```

In [14]:
```
df.query_string.value_counts()
```

Out[14]:
```
NO_QUERY    1600000
Name: query_string, dtype: int64
```

In [15]:
```
df.drop(['id','date','query_string','user'],axis=1,inplace=True)
```

In [16]:
```
df.head()
```

Out[16]:

	sentiment	text
0	0	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	is upset that he can't update his Facebook by ...
2	0	@Kenichan I dived many times for the ball. Man...
3	0	my whole body feels itchy and like its on fire
4	0	@nationwideclass no, it's not behaving at all....

In [17]:
```
df[df.sentiment == 0].head(10)
```

Out[17]:

	sentiment	text
0	0	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	is upset that he can't update his Facebook by ...
2	0	@Kenichan I dived many times for the ball. Man...
3	0	my whole body feels itchy and like its on fire
4	0	@nationwideclass no, it's not behaving at all....
5	0	@Kwesidei not the whole crew
6	0	Need a hug
7	0	@LOLTrish hey long time no see! Yes.. Rains a...
8	0	@Tatiana_K nope they didn't have it
9	0	@twittera que me muera ?

In [18]:
```
df[df.sentiment == 4].head(10)
```

Out[18]:

	sentiment	text
800000	4	I LOVE @Health4UandPets u guys r the best!!
800001	4	im meeting up with one of my besties tonight! ...
800002	4	@DaRealSunisaKim Thanks for the Twitter add, S...
800003	4	Being sick can be really cheap when it hurts t...
800004	4	@LovesBrooklyn2 he has that effect on everyone
800005	4	@ProductOfFear You can tell him that I just bu...
800006	4	@r_keith_hill Thans for your response. Ihad al...
800007	4	@KeepinUpWKris I am so jealous, hope you had a...
800008	4	@tommcfly ah, congrats mr fletcher for finally...
800009	4	@e4VoIP I RESPONDED Stupid cat is helping me ...

In [19]:
```
# set sentiment 0, 4 --> 0, 1
df['sentiment'] = df['sentiment'].map({0: 0, 4: 1})
```

In [20]:
```
df.sentiment.value_counts()
```

Out[20]:
```
1    800000
0    800000
Name: sentiment, dtype: int64
```

Data Dictionary

In [21]:
```
from pprint import pprint
```

In [23]:
```
df['pre_clean_len'] = [len(t) for t in df.text]
```

In [30]:
```
df.head(3)
```

Out[30]:

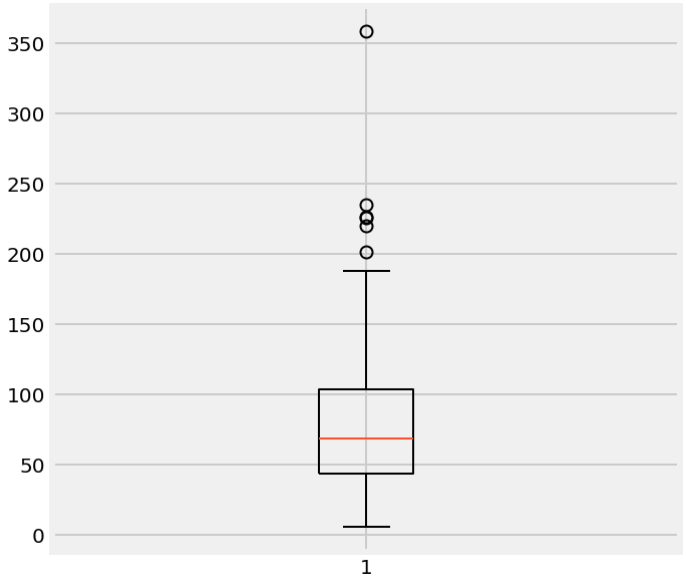
	sentiment	text	pre_clean_len
0	0	@switchfoot http://twitpic.com/2y1zl - Awww, t...	115
1	0	is upset that he can't update his Facebook by ...	111
2	0	@Kenichan I dived many times for the ball. Man...	89

In [31]:
```
data_dict = {
    'sentiment':{
        'type':df.sentiment.dtype,
        'description':'sentiment class - 0:negative, 1:positive'
    },
    'text':{
        'type':df.text.dtype,
        'description':'tweet text'
    },
    'pre_clean_len':{
        'type':df.pre_clean_len.dtype,
        'description':'Length of the tweet before cleaning'
    },
    'dataset_shape':df.shape
}
```

In [32]:
```
data_dict
```

Out[32]:
```
{'sentiment': {'type': dtype('int64'),
  'description': 'sentiment class - 0:negative, 1:positive'},
 'text': {'type': dtype('O'), 'description': 'tweet text'},
 'pre_clean_len': {'type': dtype('int64'),
  'description': 'Length of the tweet before cleaning'},
 'dataset_shape': (1600000, 3)}
```

In [34]:
```
fig, ax = plt.subplots(figsize=(5, 5))
plt.boxplot(df.pre_clean_len)
plt.show()
```



In [35]:
```
df[df.pre_clean_len > 140].head(10)
```

Out[35]:

	sentiment	text	pre_clean_len
213	0	Awwh babs... you look so sad underneith that s...	142
279	0	Whinging. My client&boss don't understand ...	145
343	0	@TheLeagueSF Not Fun & Furious? The new ma...	145
400	0	#3 woke up and was having an accident - "...	144
464	0	My bathtub drain is fired: it haz 1 job 2 do, ...	146
492	0	pears & Brie, bottle of Cabernet, and &quo...	150
747	0	Have an invite for "Healthy Dining" ...	141
957	0	Damnit I was really digging this season of Rea...	141
1064	0	Why do I keep looking...I know that what I rea...	141
1071	0	Used the term "Fail Whale" to a clie...	148

Data Preparation 1: HTML decoding

In [36]:
```
df.text[279]
```

Out[36]:
```
"Whinging. My client&amp;amp;boss don't understand English well. Rewrote so
me text unreadable. It's written by v. good writer&amp;amp;reviewed correct
ly. "
```

In [37]:
```
from bs4 import BeautifulSoup
```

In [38]:
```
example1 = BeautifulSoup(df.text[279], 'lxml')
```

In [39]:
```
print(example1.get_text())
```

```
Whinging. My client&boss don't understand English well. Rewrote some te
xt unreadable. It's written by v. good writer&reviewed correctly.
```

Data Preparation 2: @mention

In [40]:
```
df.text[343]
```

Out[40]:
```
'@TheLeagueSF Not Fun &amp; Furious? The new mantra for the Bay 2 Break
ers? It was getting 2 rambunctious;the city overreacted &amp; clamped d
own '
```

In [41]:
```
import re
```

In [42]:
```
re.sub(r'@[A-Za-z0-9]+','',df.text[343])
```

Out[42]:
```
' Not Fun &amp;amp; Furious? The new mantra for the Bay 2 Breakers? It was
getting 2 rambunctious;the city overreacted &amp;amp; clamped down '
```

Data Preparation 3: URL links

In [43]:
```
df.text[0]
```

Out[43]:
```
"@switchfoot http://twitpic.com/2y1zl - Awww, that's a bummer.  You sho
ulda got David Carr of Third Day to do it. ;D"
```

In [44]:
```
re.sub('https?://[A-Za-z0-9./]+','',df.text[0])
```

Out[44]:
```
"@switchfoot  - Awww, that's a bummer.  You shoulda got David Carr of T
hird Day to do it. ;D"
```

Data Preparation 4: UTF-8 BOM (Byte Order Mark)

In [45]:
```
df.text[226]
```

Out[45]:
```
'Tuesday▯ll start with reflection ▯n then a lecture in Stress reducing
techniques. That sure might become very useful for us accompaniers '
```

In [48]:
```
testing = df.text[226].decode("utf-8-sig")
```

```
---------------------------------------------------------------------
----
AttributeError                            Traceback (most recent call l
ast)
<ipython-input-48-274ea8fd579d> in <module>()
----> 1 testing = df.text[226].decode("utf-8-sig")

AttributeError: 'str' object has no attribute 'decode'
```

In [47]:
```
testing.replace(u"\ufffd", "?")
```

```
---------------------------------------------------------------------
----
NameError                                 Traceback (most recent call l
ast)
<ipython-input-47-261762bf4400> in <module>()
----> 1 testing.replace(u"\ufffd", "?")

NameError: name 'testing' is not defined
```

Data Preparation 5: hashtag / numbers

In [49]:
```
df.text[175]
```

Out[49]:
```
"@machineplay I'm so sorry you're having to go through this. Again.  #t
herapyfail"
```

In [50]:
```
re.sub("[^a-zA-Z]", " ", df.text[175])
```

Out[50]:
```
' machineplay I m so sorry you re having to go through this  Again    t
herapyfail'
```

Defining data cleaning function

In [53]:
```
from nltk.tokenize import WordPunctTokenizer
tok = WordPunctTokenizer()

pat1 = r'@[A-Za-z0-9]+'
pat2 = r'https?://[A-Za-z0-9./]+'
combined_pat = r'|'.join((pat1, pat2))
```

In [54]:
```
def tweet_cleaner(text):
    soup = BeautifulSoup(text, 'lxml')
    souped = soup.get_text()
    stripped = re.sub(combined_pat, '', souped)
    try:
        clean = stripped.decode("utf-8-sig").replace(u"\ufffd", "?")
    except:
        clean = stripped
    letters_only = re.sub("[^a-zA-Z]", " ", clean)
    lower_case = letters_only.lower()
    # During the letters_only process two lines above, it has created un
necessay white spaces,
    # I will tokenize and join together to remove unneccessary white spa
ces
    words = tok.tokenize(lower_case)
    return (" ".join(words)).strip()
```

In [57]:
```
testing = df.text[:10]

test_result = []
for t in testing:
    test_result.append(tweet_cleaner(t))
```

In [58]:
```
test_result
```

Out[58]:
```
['awww that s a bummer you shoulda got david carr of third day to do it
d',
 'is upset that he can t update his facebook by texting it and might cr
y as a result school today also blah',
 'i dived many times for the ball managed to save the rest go out of bo
unds',
 'my whole body feels itchy and like its on fire',
 'no it s not behaving at all i m mad why am i here because i can t see
you all over there',
 'not the whole crew',
 'need a hug',
 'hey long time no see yes rains a bit only a bit lol i m fine thanks h
ow s you',
 'k nope they didn t have it',
 'que me muera']
```

In [67]:
```
%%time

clean_tweet_texts = []

for i in range(0,len(df)):
    if( (i+1)%100000 == 0 ):
        print("%d tweets has been processed" % (i+1))
    clean_tweet_texts.append(tweet_cleaner(df['text'][i]))
```

```
100000 tweets has been processed
200000 tweets has been processed
300000 tweets has been processed
400000 tweets has been processed
500000 tweets has been processed
600000 tweets has been processed
700000 tweets has been processed
800000 tweets has been processed
900000 tweets has been processed
1000000 tweets has been processed
1100000 tweets has been processed
1200000 tweets has been processed
1300000 tweets has been processed
1400000 tweets has been processed
1500000 tweets has been processed
1600000 tweets has been processed
CPU times: user 6min 39s, sys: 22.6 s, total: 7min 1s
Wall time: 7min 6s
```

In [68]:
```
len(clean_tweet_texts)
```

Out[68]:
```
1600000
```

Saving cleaned data as csv

In [69]:
```
clean_df = pd.DataFrame(clean_tweet_texts,columns=['text'])
clean_df['target'] = df.sentiment
clean_df.head(3)
```

Out[69]:

	text	target
0	awww that s a bummer you shoulda got david car...	0
1	is upset that he can t update his facebook by ...	0
2	i dived many times for the ball managed to sav...	0

In [70]:
```
clean_df.to_csv('../pyspark/clean_tweet.csv',encoding='utf-8')
```

In [71]:
```
csv = '../pyspark/clean_tweet.csv'
my_df = pd.read_csv(csv, index_col=0)
my_df.head(3)
```

```
/Users/bongwon/anaconda3/lib/python3.6/site-packages/numpy/lib/arraysetops.py:472: FutureWarning: elementwise comparison failed; returning scalar instead, but in the future will perform elementwise comparison
  mask |= (ar1 == a)
```

Out[71]:

	text	target
0	awww that s a bummer you shoulda got david car...	0
1	is upset that he can t update his facebook by ...	0
2	i dived many times for the ball managed to sav...	0

Reading the data as Data Frame

In [4]:
```
df = spark.read.option("header", "true") \
                    .option("nullValue", "?") \
                    .option("inferSchema", "true") \
                    .csv('../pyspark/clean_tweet.csv')
```

In [5]:
```
df.take(5)
```

Out[5]:
```
[Row(_c0=0, text='awww that s a bummer you shoulda got david carr of third day to do it d', target=0),
 Row(_c0=1, text='is upset that he can t update his facebook by texting it and might cry as a result school today also blah', target=0),
 Row(_c0=2, text='i dived many times for the ball managed to save the rest go out of bounds', target=0),
 Row(_c0=3, text='my whole body feels itchy and like its on fire', target=0),
 Row(_c0=4, text='no it s not behaving at all i m mad why am i here because i can t see you all over there', target=0)]
```

In [6]:
```
df = df.dropna()
```

In [7]:
```
df.count()
```

Out[7]:
```
1596753
```

In [8]:
```
(train_set, val_set, test_set) = df.randomSplit([0.98, 0.01, 0.01], seed
 = 2000)
```

HashingTF + IDF + Logistic Regression

In [9]:
```
from pyspark.ml.feature import HashingTF, IDF, Tokenizer, CountVectorize
r
from pyspark.ml.feature import StringIndexer
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator


from pyspark.ml.feature import HashingTF, IDF, Tokenizer
from pyspark.ml.feature import StringIndexer
from pyspark.ml import Pipeline
```

In [10]:
```
tokenizer = Tokenizer(inputCol="text", outputCol="words")
hashtf = HashingTF(numFeatures=2**16, inputCol="words", outputCol='tf')
idf = IDF(inputCol='tf', outputCol="features", minDocFreq=5) #minDocFre
q: remove sparse terms
label_stringIdx = StringIndexer(inputCol = "target", outputCol = "label"
)
pipeline = Pipeline(stages=[tokenizer, hashtf, idf, label_stringIdx])
```

In [11]:
```
pipelineFit = pipeline.fit(train_set)
```

In [12]:
```
train_df = pipelineFit.transform(train_set)
val_df = pipelineFit.transform(val_set)
```

In [13]:
```
train_df.show(5)
```

```
+---+--------------------+------+--------------------+----------------
---+--------------------+-----+
|_c0|                text|target|               words|
tf|            features|label|
+---+--------------------+------+--------------------+----------------
---+--------------------+-----+
|  0|awww that s a bum...|     0|[awww, that, s, a...|(65536,[8436,884
7...|(65536,[8436,8847...|  0.0|
|  1|is upset that he ...|     0|[is, upset, that,...|(65536,[1444,207
1...|(65536,[1444,2071...|  0.0|
|  2|i dived many time...|     0|[i, dived, many, ...|(65536,[2548,288
8...|(65536,[2548,2888...|  0.0|
|  3|my whole body fee...|     0|[my, whole, body,...|(65536,[158,1165
0...|(65536,[158,11650...|  0.0|
|  4|no it s not behav...|     0|[no, it, s, not, ...|(65536,[1968,448
8...|(65536,[1968,4488...|  0.0|
+---+--------------------+------+--------------------+----------------
---+--------------------+-----+
only showing top 5 rows
```

In [14]:
```
from pyspark.ml.classification import LogisticRegression
```

In [15]:
```
lr = LogisticRegression(maxIter=100)
```

In [16]:
```
lrModel = lr.fit(train_df)
```

In [17]:
```
predictions = lrModel.transform(val_df)
```

In [18]:
```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPredictio
n")
evaluator.evaluate(predictions)
```

Out[18]:
```
0.8595046645453751
```

In [19]:
```
evaluator.getMetricName()
```

Out[19]:
```
'areaUnderROC'
```

In [20]:
```
accuracy = predictions.filter(predictions.label == predictions.predictio
n).count() / float(val_set.count())
```

In [21]:
```
accuracy
```

Out[21]:
```
0.7907094168139359
```

CountVectorizer + IDF + Logistic Regression

In [22]:
```
from pyspark.ml.feature import CountVectorizer

tokenizer = Tokenizer(inputCol="text", outputCol="words")
cv = CountVectorizer(vocabSize=2**16, inputCol="words", outputCol='cv')
idf = IDF(inputCol='cv', outputCol="features", minDocFreq=5) #minDocFre
q: remove sparse terms
label_stringIdx = StringIndexer(inputCol = "target", outputCol = "label"
)
lr = LogisticRegression(maxIter=100)
pipeline = Pipeline(stages=[tokenizer, cv, idf, label_stringIdx, lr])
```

In [23]:
```
%%time
pipelineFit = pipeline.fit(train_set)
```

```
CPU times: user 33.4 ms, sys: 10.8 ms, total: 44.2 ms
Wall time: 1min 7s
```

In [25]:
```
predictions = pipelineFit.transform(val_set)
accuracy = predictions.filter(predictions.label == predictions.predictio
n).count() / float(val_set.count())
roc_auc = evaluator.evaluate(predictions)

print ("Accuracy Score: {0:.4f}".format(accuracy))
print ("ROC-AUC: {0:.4f}".format(roc_auc))
```

```
Accuracy Score: 0.7949
ROC-AUC: 0.8658
```

N-gram Implementation

In [26]:
```
from pyspark.ml.feature import NGram, VectorAssembler
from pyspark.ml.feature import ChiSqSelector
```

In [27]:
```
def build_trigrams(inputCol=["text","target"], n=3):
    tokenizer = [Tokenizer(inputCol="text", outputCol="words")]
    ngrams = [
        NGram(n=i, inputCol="words", outputCol="{0}_grams".format(i))
        for i in range(1, n + 1)
    ]

    cv = [
        CountVectorizer(vocabSize=2**14,inputCol="{0}_grams".format(i),
            outputCol="{0}_tf".format(i))
        for i in range(1, n + 1)
    ]
    idf = [IDF(inputCol="{0}_tf".format(i), outputCol="{0}_tfidf".format
(i), minDocFreq=5) for i in range(1, n + 1)]

    assembler = [VectorAssembler(
        inputCols=["{0}_tfidf".format(i) for i in range(1, n + 1)],
        outputCol="rawFeatures"
    )]
    label_stringIdx = [StringIndexer(inputCol = "target", outputCol = "l
abel")]
    selector = [ChiSqSelector(numTopFeatures=2**14,featuresCol='rawFeatu
res', outputCol="features")]
    lr = [LogisticRegression(maxIter=100)]

    return Pipeline(stages=tokenizer + ngrams + cv + idf+ assembler + la
bel_stringIdx+selector+lr)
```

In [ ]:
```
%%time
# this take about 6 hours
# so don't run it unless you really indent to
trigram_pipelineFit = build_trigrams().fit(train_set)
```

In [ ]:
```
predictions = trigram_pipelineFit.transform(val_set)
accuracy = predictions.filter(predictions.label == predictions.predictio
n).count() / float(dev_set.count())
roc_auc = evaluator.evaluate(predictions)

# print accuracy, roc_auc
print ("Accuracy Score: {0:.4f}".format(accuracy))
print ("ROC-AUC: {0:.4f}".format(roc_auc))
```

In [29]:
```
from pyspark.ml.feature import NGram, VectorAssembler
```

In [30]:
```
def build_ngrams_wocs(inputCol=["text","target"], n=3):
    tokenizer = [Tokenizer(inputCol="text", outputCol="words")]
    ngrams = [
        NGram(n=i, inputCol="words", outputCol="{0}_grams".format(i))
        for i in range(1, n + 1)
    ]

    cv = [
        CountVectorizer(vocabSize=5460,inputCol="{0}_grams".format(i),
            outputCol="{0}_tf".format(i))
        for i in range(1, n + 1)
    ]
    idf = [IDF(inputCol="{0}_tf".format(i), outputCol="{0}_tfidf".format
(i), minDocFreq=5) for i in range(1, n + 1)]

    assembler = [VectorAssembler(
        inputCols=["{0}_tfidf".format(i) for i in range(1, n + 1)],
        outputCol="features"
    )]
    label_stringIdx = [StringIndexer(inputCol = "target", outputCol = "l
abel")]
    lr = [LogisticRegression(maxIter=100)]
    return Pipeline(stages=tokenizer + ngrams + cv + idf+ assembler + la
bel_stringIdx+lr)
```

In [31]:
```
%%time
# it takes about 10 min
trigramwocs_pipelineFit = build_ngrams_wocs().fit(train_set)
```

In [32]:
```
predictions_wocs = trigramwocs_pipelineFit.transform(val_set)
accuracy_wocs = predictions_wocs.filter(predictions_wocs.label == predic
tions_wocs.prediction).count() / float(val_set.count())
roc_auc_wocs = evaluator.evaluate(predictions_wocs)

# print accuracy, roc_auc
print ("Accuracy Score: {0:.4f}".format(accuracy_wocs))
print ("ROC-AUC: {0:.4f}".format(roc_auc_wocs))
```

```
Accuracy Score: 0.8117
ROC-AUC: 0.8871
```