fakenews

October 11, 2023

[3]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
import matplotlib.pyplot as plt
import itertools
from sklearn import svm
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn import metrics
import spacy
from sklearn.feature_extraction.stop_words import ENGLISH_STOP_WORDS
import string
import re
import nltk
import collections
from nltk.corpus import stopwords
from sklearn.feature_extraction import DictVectorizer
from sklearn.pipeline import Pipeline, FeatureUnion
from empath import Empath
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
import pickle
```

```
Using TensorFlow backend.
```

[4]:
```
df = pd.read_csv('Dataset/data.csv')
df.loc[df['Label']== 0, 'Label'] = 'REAL'
df.loc[df['Label']== 1, 'Label'] = 'FAKE'
df.columns
df['Label'].value_counts()
```

[4]:
```
REAL    2137
FAKE    1872
Name: Label, dtype: int64
```

[5]:
```
#Dropping the column URLs from the table
df.drop(['URLs'], axis = 1, inplace = True)
df.columns
```

[5]:
```
Index(['Headline', 'Body', 'Label'], dtype='object')
```

[6]:
```
#Selecting only fake news from all the types of news and then replacing the␣
 ↪'fake' by 0
df1 = pd.read_csv('Dataset/fake.csv')
df1.columns
df1['type'].value_counts()
df1 = df1.loc[df1['type']=='fake']
df1.loc[df1['type']== 'fake', 'type'] = 'FAKE'
```

[7]:
```
#Selecting some columns from the table and renaming them\n",
df1 = df1[['title','text','type']]
df1.columns = ['Headline', 'Body', 'Label']
df1['Label'].value_counts()
```

[7]:
```
FAKE    19
Name: Label, dtype: int64
```

[8]:
```
df2 = pd.read_csv('Dataset/fake_or_real_news.csv')
df2.columns
```

[8]:
```
Index(['Unnamed: 0', 'title', 'text', 'label'], dtype='object')
```

[9]:
```
#Selecting few columns from the table and renaming the columns
df2 = df2[['title','text','label']]
df2.columns = ['Headline', 'Body', 'Label']
df2.columns
df2['Label'].value_counts()
```

[9]:
```
REAL    3171
FAKE    3164
Name: Label, dtype: int64
```

[10]:
```
df3 = pd.read_csv('Dataset/train.csv')
df3.columns
```

[10]:
```
Index(['id', 'title', 'author', 'text', 'label'], dtype='object')
```

[11]:
```
#Selecting few columns from the table and renaming the columns
df3 = df3[['title','text','label']]
df3.columns = ['Headline', 'Body', 'Label']
df3.loc[df3['Label']== 0, 'Label'] = 'REAL'
df3.loc[df3['Label']== 1, 'Label'] = 'FAKE'
```

```
df3.columns
df3['Label'].value_counts()
```

[11]:
```
FAKE    10413
REAL    10387
Name: Label, dtype: int64
```

[12]:
```
#Appending df1,df2,df3 to df
df = df.append(df1, ignore_index = True)
df = df.append(df2, ignore_index = True)
df = df.append(df3, ignore_index = True)
```

[13]:
```
df = df.drop_duplicates()

# df.iloc[3647]
# print(df['Headline'][3647])
# print(len(df['Body'][3647]))
#df = df.dropna(how='any',axis=0)
cnt = 0
ind = []
for art in df['Body']:
    #print(type(art))
    if len(str(art)) < 10:
        ind.append(cnt)
    cnt+=1
df = df.drop(df.index[ind])


df
# print(df['Headline'][3647])
# print(len(df['Body'][3647]))
```

[13]:

	Headline
0	Four ways Bob Corker skewered Donald Trump
1	Linklater's war veteran comedy speaks to moder…
2	Trump’s Fight With Corker Jeopardizes His Legi…
3	Egypt's Cheiron wins tie-up with Pemex for Mex…
4	Jason Aldean opens 'SNL' with Vegas tribute
5	JetNation FanDuel League; Week 4
6	Kansas Tried a Tax Plan Similar to Trump’s. It…
7	India RBI chief: growth important, but not at …
8	EPA chief to sign rule on Clean Power Plan exi…
9	Talks on sale of Air Berlin planes to easyJet …
10	U.S. President Donald Trump Quietly Signs Law …
11	2017 Fantasy Football Team Defense Rankings - …
12	Just Shut Up & Play Some Damn Baseball!!
13	Deloitte cyber attack affected up to 350 clien…

```
14      10/7: Chuck Axed; HBD Brickyard, Adam, Moonlig…
15      Gunman’s Girlfriend Said She Didn’t Know He Pl…
16      Marilou Danley, Gunman’s Girlfriend, Says She …
17      Trump’s Immigration Rhetoric Echoes a Bitter F…
18      Trump Bemoans ‘Little Appreciation’ As San Jua…
19      In Meeting With Military, Trump Talks Of 'Calm…
20      Teacher Sparks Outrage By Asking Kids To Make …
21      9/28 Through the 40s: The Gloaming; HBD Bill, …
22      Weinstein Co board ousts Harvey Weinstein afte…
23      Hillary Clinton Suggests That Trump May Order …
24      9/29 Through the 40s: HBD Cannonball & Paul, C…
25           Sharapova storms into Tianjin quarter-finals
26      10/3 Expo Park-Forbes Field Era: Pirates, Gray…
27             Weinstein scandal no surprise to Hollywood
28                   Blackhawks Roster Breakdown: Goalies
29      With Christian Pulisic Driving, United States …
…                                                     …
31128   NFL Preview: Championship Match-Ups Prove Team…
31129   President Trump’s Father’s Day Proclamation: D…
31130   Former Ambassador Andrew Young Calls for End t…
31131   Osama bin Laden’s older brother rents out luxu…
31132        WORLD WAR 3   Mr.President #004   xFrozenLPx
31133   HUMA ABEDIN SWORE UNDER OATH SHE GAVE UP ‘ALL …
31134                                                 NaN
31135                                                 NaN
31137   Government Report: Islamists Building ’Paralle…
31140   Editor of Austria’s Largest Paper Charged with…
31141   This Is a Jobs Report That Democrats Can Boast…
31142   Christians in 2017 ’Most Persecuted Group in t…
31143   Florida Woman Charged in Death of Infant in ‘C…
31144   Time is Running Out to Stop Kratom Ban - Need …
31145   The Fix Is In: NBC Affiliate Accidentally Post…
31146   Samsung, Kim Jong-un, Rex Tillerson: Your Morn…
31147   Comment on World Heaves Sigh of Relief after T…
31148   Ann Coulter: How to Provide Universal Health C…
31149   Government Forces Advancing at Damascus-Aleppo…
31150   Sally Yates Won’t Say If Trump Was Wiretapped …
31151   Maine’s Gov. LePage Threatens To ‘Investigate’…
31152   Sen. McConnell: The Supreme Court Vacancy Was …
31153   Nikki Haley Blasts U.N. Human Rights Office fo…
31155   Jakarta Bombing Kills Three Police Officers, L…
31156   Idiot Who Destroyed Trump Hollywood Star Gets …
31157   Trump: Putin ’Very Smart’ to Not Retaliate ove…
31158   Rapper T.I.: Trump a ’Poster Child For White S…
31159   N.F.L. Playoffs: Schedule, Matchups and Odds -…
31160   Macy’s Is Said to Receive Takeover Approach by…
31162                          What Keeps the F-35 Alive
```

	Body	Label
0	Image copyright Getty Images\nOn Sunday mornin…	FAKE
1	LONDON (Reuters) - "Last Flag Flying", a comed…	FAKE
2	The feud broke into public view last week when…	FAKE
3	MEXICO CITY (Reuters) - Egypt’s Cheiron Holdin…	FAKE
4	Country singer Jason Aldean, who was performin…	FAKE
5	JetNation FanDuel League; Week 4\n% of readers…	REAL
6	In 2012, Kansas lawmakers, led by Gov. Sam Bro…	FAKE
7	The Reserve Bank of India (RBI) Governor Urjit…	FAKE
8	Scott Pruitt, Administrator of the U.S. Enviro…	FAKE
9	FILE PHOTO - An Air Berlin sign is seen at an …	FAKE
10	By Aaron Kesel\nAs former White House chief of…	REAL
11	2017 Fantasy Football Team Defense Rankings - …	REAL
12	Just Shut Up & Play Some Damn Baseball!!\n(Bef…	REAL
13	FILE PHOTO: The Deloitte Company logo is seen …	FAKE
14	A Potato Battery Can Light up a Room for Over …	REAL
15	• The authorities found evidence that the gunm…	FAKE
16	The statement, which was read by her lawyer, M…	FAKE
17	In bold documentary style, Retro Report looks …	FAKE
18	Red Flag Warning: These California Wildfires A…	REAL
19	In Meeting With Military, Trump Talks Of 'Calm…	REAL
20	Red Flag Warning: These California Wildfires A…	REAL
21	Vietnam Is in Great Danger, You Must Publish a…	REAL
22	(Reuters) - The Weinstein Co has fired co-Chai…	FAKE
23	Hillary Clinton Suggests That Trump May Order …	REAL
24	Red Flag Warning: These California Wildfires A…	REAL
25	(Reuters) - Maria Sharapova overpowered Poland…	FAKE
26	Red Flag Warning: These California Wildfires A…	REAL
27	Chat with us in Facebook Messenger. Find out w…	FAKE
28	Blackhawks Roster Breakdown: Goalies\n(Before …	REAL
29	When Pulisic tore open the left side of the Pa…	FAKE
…	…	…
31128	The NFL is a league, so it should come as no…	REAL
31129	President Donald Trump officially declared tod…	REAL
31130	By Brandon Turbeville Anti-fluoridation activi…	FAKE
31131	Osama bin Laden’s older brother rents out luxu…	FAKE
31132	source Add To The Conversation Using Facebook …	FAKE
31133	Home › POLITICS	US NEWS › HUMA ABEDIN SWORE …	FAKE
31134	DYN's Statement on Last Week's Botnet Attack h…	FAKE
31135	Kinda reminds me of when Carter gave away the …	FAKE
31137	Aided by a politically correct culture of "tol…	REAL
31140	Breitbart October 26, 2016 \nAn editor of Aust…	FAKE
31141	There’s not much to say about the July jobs nu…	REAL
31142	In many parts of the world, Christians gatheri…	REAL
31143	Early on Oct. 6, Erin was awakened by the so…	REAL
31144	By Brandon Turbeville When the DEA announced t…	FAKE

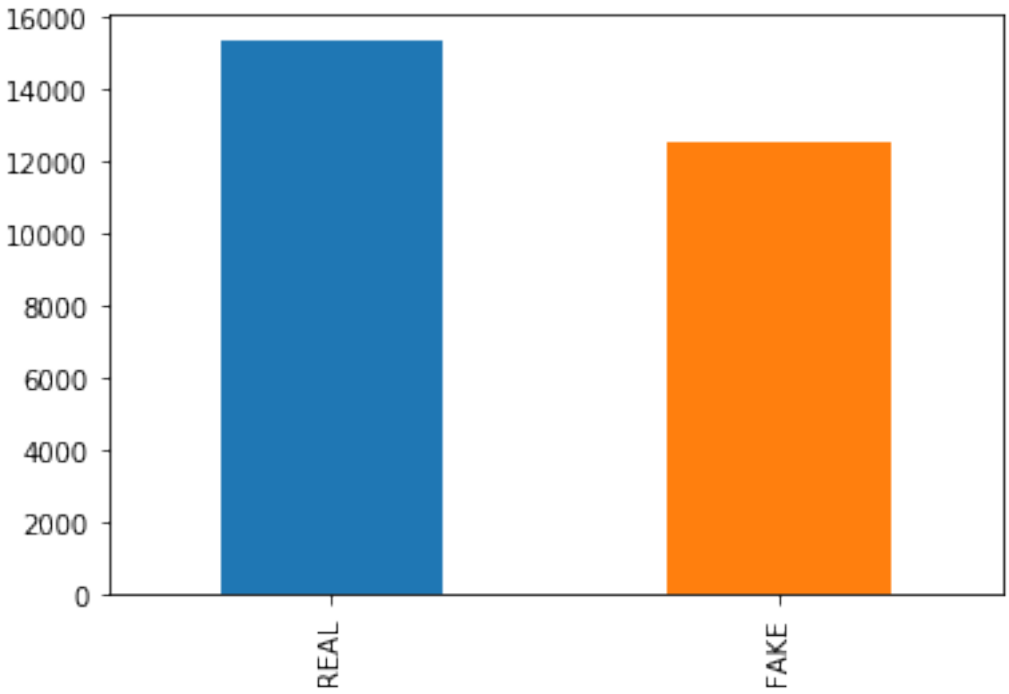
```
31145  Home » Headlines » World News » The Fix Is In:…  FAKE
31146  Good morning.  Here's what you need to know: •…  REAL
31147    Finian Cunningham has written extensively on…  FAKE
31148  The first sentence of Congress' Obamacare repe…  REAL
31149  #FROMTHEFRONT #MAPS 22.11.2016 - 1,361 views 5…  FAKE
31150  Former Deputy Attorney General Sally Yates dec…  REAL
31151  Google Pinterest Digg Linkedin Reddit Stumbleu…  FAKE
31152  Senate Majority Leader Mitch McConnell (R, KY)…  REAL
31153  U. S Ambassador to the United Nations Nikki Ha…  REAL
31155  Two suicide bombers attacked a bus station in …  REAL
31156  Share This \nAlthough the vandal who thought i…  FAKE
31157  Donald Trump took to Twitter Friday to praise …  REAL
31158  Rapper T. I. unloaded on black celebrities who…  REAL
31159  When the Green Bay Packers lost to the Washing…  REAL
31160  The Macy's of today grew from the union of sev…  REAL
31162    David Swanson is an author, activist, journa…  FAKE

[27865 rows x 3 columns]
```

[14]:
```
df['Label'].value_counts()
```

[14]:
```
REAL    15343
FAKE    12522
Name: Label, dtype: int64
```

[15]:
```
df['Label'].value_counts().plot(kind = 'bar')
```

[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7efe8a634ef0>
```



[16]:
```
df['headline_length'] = [len(str(a)) for a in df['Headline']]
df['headline_length'].describe()
```

[16]:
```
count    27865.000000
mean        69.775381
std         24.885773
min          1.000000
25%         55.000000
50%         70.000000
75%         85.000000
max        653.000000
Name: headline_length, dtype: float64
```

[17]:
```
df['body_length'] = [len(a) for a in df['Body']]
df['body_length'].describe()
```

[17]:
```
count     27865.000000
mean       4429.890903
std        4854.862554
min          10.000000
25%        1589.000000
50%        3348.000000
75%        6106.000000
max      142961.000000
```

```
Name: body_length, dtype: float64
```

[18]:
```
df.describe()
```

[18]:

	headline_length	body_length
count	27865.000000	27865.000000
mean	69.775381	4429.890903
std	24.885773	4854.862554
min	1.000000	10.000000
25%	55.000000	1589.000000
50%	70.000000	3348.000000
75%	85.000000	6106.000000
max	653.000000	142961.000000

[19]:
```
df["Text"] = df["Headline"].map(str) + df["Body"]
y = df.Label
y = y.astype('str')
X_train, X_test, Y_train, Y_test = train_test_split(df['Text'],y, test_size=0.
 ↪33)
X_train
```

[19]:
```
4258     Donald Trump's GOP civil warPanama City, Flori…
24466    What You Should Watch: Amazon Pilots and ‘Miss…
28155    Part 4 Of O’Keefe’s Project Veritas Videos Has…
28266    Misophonia Sufferers: Scientists May Have Foun…
25623    IS Urges Jihadis To ’Dress Up Like Jews’, Carr…
14063    Freedom Rider: Russophobia: War Party Propagan…
20265    Bob Dylan, the Newest Nobel Laureate, Maintain…
30423    Airbag Propellant Bound for Takata Factory Det…
25876    If Clinton Goes Down, Loretta Lynch Will Go Do…
22322    Report: NBC Cutting Back ’Today’ to Make Room …
5265     Re: The U.S. Will Lose Global Reserve Status, …
4785     What Brian Williams’s Chopper Whopper Says Abo…
14883    Woody Harrelson Quits Smoking Pot after 30 Yea…
25502    Be More Productive. Shorten the Workweek. - Th…
14800    Anthony Weiner and Huma Abedin to Separate Aft…
19216    Irwin Stambler, 92, Dies Reference Book Writer…
690      State Department denies Tillerson called Trump…
3586     DNA confirms amazing Australian isle insect no…
24347    Connecticut High School Principal Denounces ’T…
15566    Revisiting 1917, a Year That Reverberates for …
9081     Why I’m Suing Vanderbilt UniversityTaki's Maga…
6194     Democrats are actually more enthusiastic than …
28878    The Electoral College Is Hated by Many. So Why…
15395    U.S. Team Wins Bocuse d’Or Competition for Fir…
29741    nanWow ever notice how when a liar is caught h…
21980    Donald Trump Delays Moving Embassy in Israel t…
```

```
5459     Baltimore protests: Crowds stand firm after cu…
30498    Processed red meat found to sort you outProces…
13586    Hedge Fund Titan’s Surefire Bet Turns Into a $…
14951    Paul: Will a GOP Congress Ever Balance the Bud…
                              …
9004     Sanders campaign sues DNC after database breac…
18462    Stoke-on-Trent Imam ’Told Congregation to Supp…
30923    Drone Restrictions Can Help Peaceful Protester…
12516    UK to Teach ’Sex and Relationship Education’ t…
807      9/29 Through the 40s: HBD Cannonball & Paul, C…
10907    Woodward: Trump Dossier Is a ’Garbage Document…
25078    Rep. DeSantis: Five Key Facts About the Obamac…
22945    Colombia Mudslide Sends Rescuers and Relatives…
26118    CNN: Democrats Face Their Powerlessness - Brei…
25987    Katastrophenschutz warnt: Werwölfe heute Nacht…
13791    Nazis sick and tired of being tarnished with s…
28065    Snap Is Said to Pick Bankers for an I.P.O. - T…
30812    Conway: We’re Seeing ’Hysterical’ Democratic P…
640      Puerto Rico Hoax (Video)(Before It's News)\nSa…
22094    Neighbor Shoots Knife-Wielding Man Trying to D…
11408    How to report voter fraud - Crash Course.By Sh…
11818    Türkei: Kritischer Journalist interviewt Oppos…
19941    Pomegranate Pistachio Pancakes - David Avocado…
3665     The Girl in the No. 8 JerseyStacee’s family so…
8402     Comment on Are We About To See One Of The Grea…
18380    If You Are On Social Media, Then You Are Alrea…
21031    Aston Martin Broadens the Brand - The New York…
7683     Why Isn’t NSA Surveillance an Election Issue?B…
13361    Oregon Man Accused of Trying to Shove Co-Worke…
4840     Confirmed: Public overwhelmingly (10-to-1) say…
7860     DNC To Sue Trump For Telling Truth: Trump Admi…
6120     Michael Moore Owes Me $4.99(128 fans) - Advert…
30804    Bargain-Hunting Frenzy Threatens Traditional D…
30794    The Mothers\nOctober 28, 2016 The Mothers by s…
18277    Report: U.S., Russia Said Agree To Assist Isra…
Name: Text, Length: 18669, dtype: object
```

[20]:
```
#Tf-idf Bigrams
#Initialize the `tfidf_vectorizer`
tfidf_vectorizer = TfidfVectorizer(stop_words='english', ngram_range = (2,2))

# Fit and transform the training data
tfidf1_train = tfidf_vectorizer.fit_transform(X_train.astype('str'))

# Transform the test set
tfidf1_test = tfidf_vectorizer.transform(X_test.astype('str'))
```

```
pickle.dump(tfidf1_train, open("tfidf1_train.pickle", "wb"))

pickle.dump(tfidf1_test, open("tfidf1_test.pickle", "wb"))
```

[21]:
```
#Top 10 tfidf bigrams
tfidf_vectorizer.get_feature_names()[-10:]
```

[21]:
```
['  cooperación',
 '  coopération',
 '      ',
 '   mediamass',
 '    ',
 ' ww reverbnation',
 '      despite',
 '      foreigner',
 '  translates',
 '   2012']
```

[22]:
```
tfidf1_train
```

[22]:
```
<18669x4013463 sparse matrix of type '<class 'numpy.float64'>'
        with 6804483 stored elements in Compressed Sparse Row format>
```

[41]:
```
#Confusion Matrix
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix')

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
```

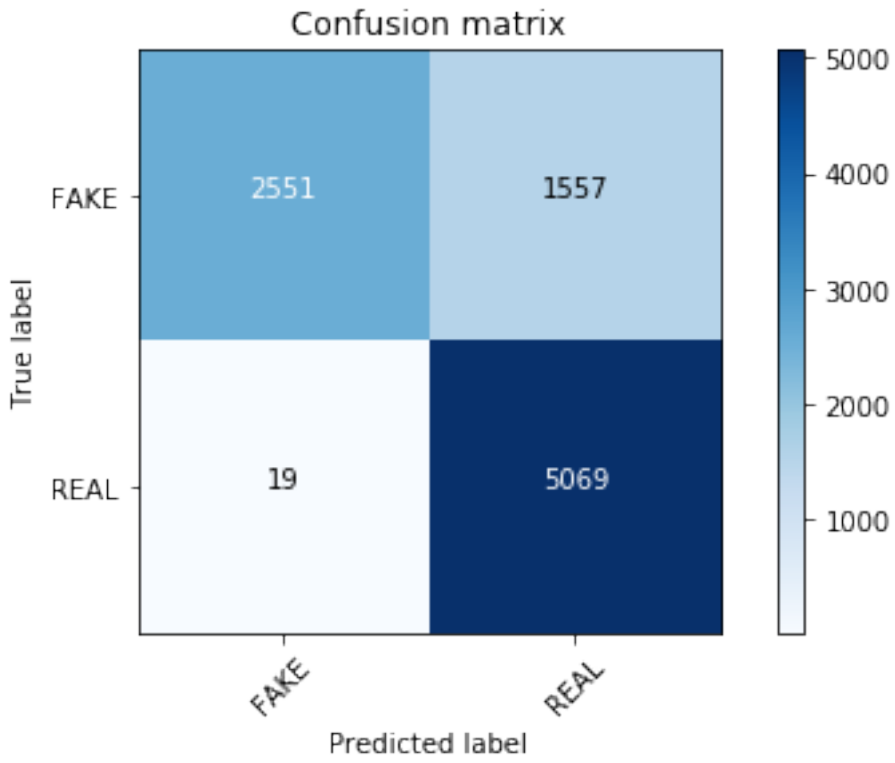
```
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

[96]:
```
clf = MultinomialNB()
clf.fit(tfidf1_train, Y_train)
pickle.dump(clf, open('tfidf_nb', 'wb'))
pred = clf.predict(tfidf1_test)
score = metrics.accuracy_score(Y_test, pred)
print("Accuracy with Multinomial Naive Bayes:   %0.3f" % score)
```

```
Accuracy with Multinomial Naive Bayes:   0.829
```

[97]:
```
cm = metrics.confusion_matrix(Y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
Confusion matrix
```



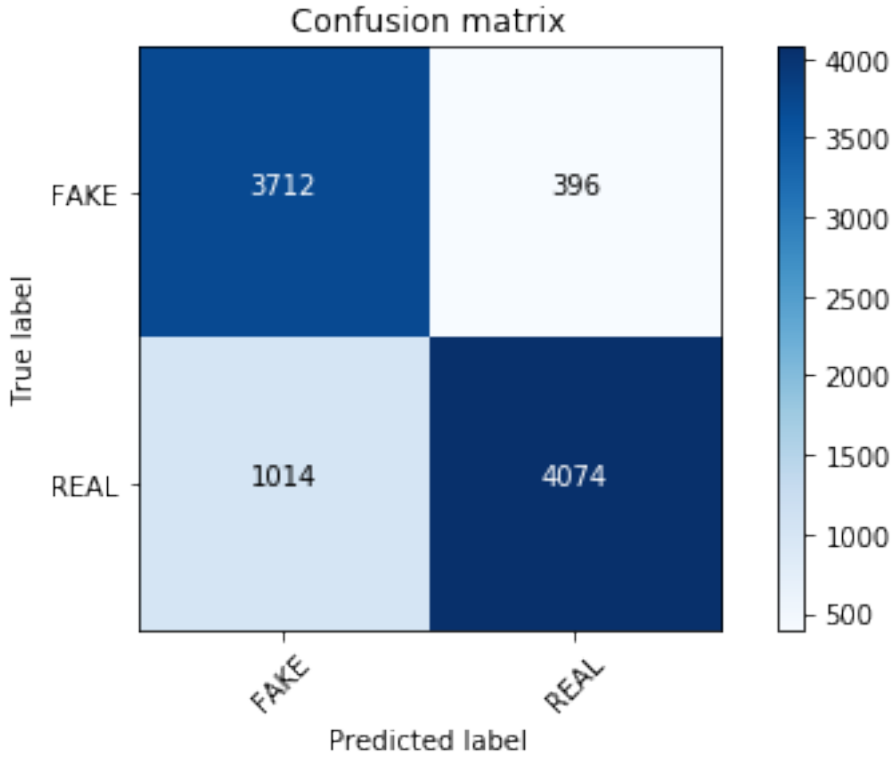
[100]:
```
clf = GradientBoostingClassifier()
clf.fit(tfidf1_train, Y_train)
pickle.dump(clf, open('tfidf_gb', 'wb'))
```

```
#model = pickle.load(open('tfidf_gb', 'rb'))
pred = clf.predict(tfidf1_test)
score = metrics.accuracy_score(Y_test, pred)
print("Accuracy with Gradient Boosting:   %0.3f" % score)
```

```
Accuracy with Gradient Boosting:   0.847
```

[101]:
```
cm = metrics.confusion_matrix(Y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
Confusion matrix
```

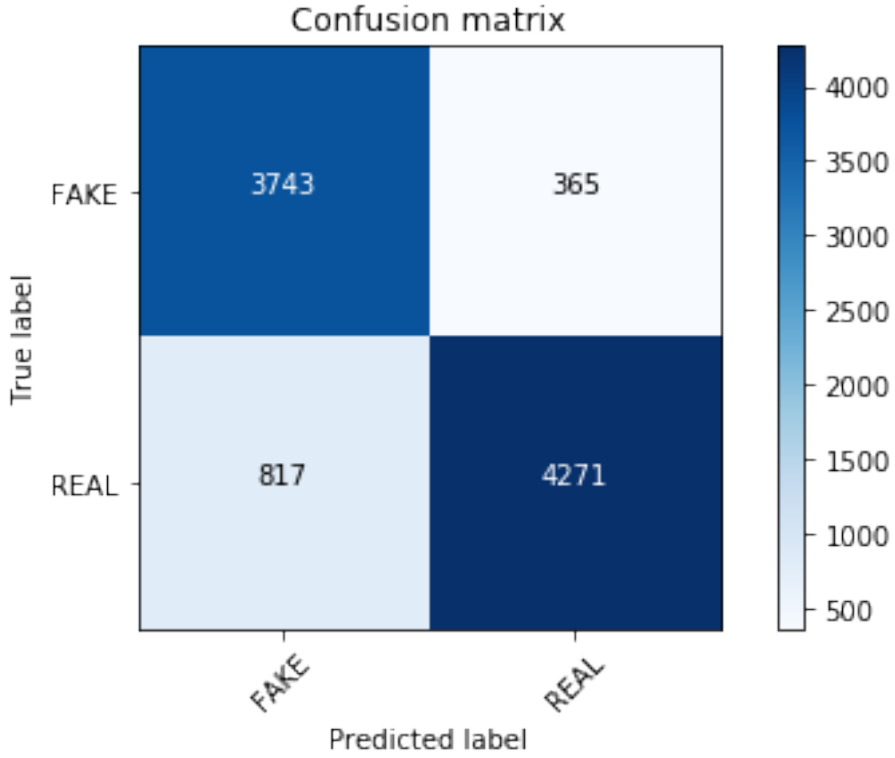


[102]:
```
clf = RandomForestClassifier()
clf.fit(tfidf1_train, Y_train)
pickle.dump(clf, open('tfidf_rf', 'wb'))
pred = clf.predict(tfidf1_test)
score = metrics.accuracy_score(Y_test, pred)
print("Accuracy with RandomForestClassifier:   %0.3f" % score)
```

```
Accuracy with RandomForestClassifier:   0.865
```

[34]:
```
cm = metrics.confusion_matrix(Y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

Confusion matrix



[ ]:
```
#Generating the POS tags for all the articles and adding a new column by␣
 ↪replacing text with their POS tags
nlp = spacy.load('en_core_web_sm')
x = []
df["Text"] = df["Headline"].map(str) + df["Body"]
for text in df['Text']:
    text_new = []
    doc = nlp(text)
    for token in doc:
        text_new.append(token.pos_)
    txt = ' '.join(text_new)
    x.append(txt)
df['Text_pos'] = x
df.to_pickle('newdata.pkl')
```

[40]:
```
df = pd.read_pickle('newdata.pkl')
cnt = 0
ind = []
for art in df['Body']:
    #print(type(art))
```

```
    if len(str(art)) < 10:
        ind.append(cnt)
    cnt+=1
df = df.drop(df.index[ind])
```

[41]:
```
y = df.Label
y = y.astype('str')
x_train, x_test, y_train, y_test = train_test_split(df['Text_pos'],y,␣
 ↪test_size=0.33)
x_train
```

[41]:
```
4574     PROPN PROPN PROPN VERB NOUN ADP PROPN PROPN AD…
9417     ADV PROPN VERB PROPN ADP PROPN VERB VERB PART …
21627    NOUN PUNCT ADJ NOUN VERB NUM ADP VERB ADJ ADP …
4755     PUNCT PUNCT PUNCT PROPN PUNCT PUNCT PUNCT NOUN…
19184    PROPN PROPN PROPN PROPN PROPN PROPN VERB VERB …
19182    ADP PRON VERB ADJ ADP DET PROPN ADP DET PROPN …
28908    NOUN VERB ADP PROPN PROPN NOUN PUNCT PUNCT PRO…
1589     PROPN PART PROPN PUNCT PROPN SYM PROPN PROPN P…
14069    NUM PROPN PROPN CCONJ PROPN PROPN VERB ADP PRO…
20099    PROPN NUM PUNCT PROPN PROPN PROPN NUM PUNCT AD…
1578     VERB PRON VERB ADP DET PROPN PROPN PRON VERB A…
21617    PROPN PROPN PROPN PROPN ADP PROPN PROPN PROPN …
24919    NUM NOUN ADP PROPN PART PROPN PROPN PUNCT DET …
17073    PROPN VERB VERB PROPN PROPN PROPN ADV ADP NOUN…
9210     PROPN VERB PRON ADV PUNCT ADV ADP PROPN ADJ AD…
26110    PROPN PROPN PROPN PROPN PROPN PUNCT NOUN VERB …
19032    PROPN PUNCT ADJ VERB ADV ADP PROPN PROPN PUNCT…
21120    NUM PROPN PROPN PROPN VERB DET NOUN PUNCT NOUN…
28216    PROPN ADP PROPN PROPN PROPN PROPN PROPN ADP PR…
3745     PROPN PROPN VERB DET PROPN PROPN PUNCT PROPN P…
26307    ADV DET PROPN PROPN PUNCT PROPN PROPN ADV VERB…
16599    PROPN CCONJ PROPN PROPN ADP PROPN ADP PROPN PR…
16018    PROPN PART PROPN VERB VERB DET PROPN ADP PROPN…
20269    PROPN PROPN VERB NOUN NOUN VERB VERB DET ADJ A…
19783    PUNCT ADV NOUN NOUN VERB PROPN PROPN NOUN PUNC…
11410    NOUN PUNCT PROPN PROPN PROPN PROPN PROPN CCONJ…
22889    PROPN VERB ADP PROPN PROPN PROPN PRON VERB PAR…
19269    PROPN PROPN PART VERB NOUN PROPN VERB ADP ADJ …
15738    NOUN PUNCT PROPN PROPN PROPN PROPN VERB NOUN N…
30577    PROPN PROPN ADP PROPN PROPN PROPN PUNCT VERB A…
                                  …
10029    DET PROPN PROPN NOUN ADP PROPN CCONJ PROPN DET…
7194     PROPN PROPN PROPN VERB DET ADJ NOUN ADP ADJ NO…
22436    PROPN PROPN VERB PART PUNCT PROPN PROPN PROPN …
8468     PROPN NOUN NOUN VERB ADJ NOUN ADP NUM PUNCT NO…
25138    PROPN PUNCT PROPN CCONJ ADV PUNCT VERB PUNCT P…
```

```
28411    PROPN CCONJ PROPN VERB NOUN NOUN ADV ADP DET N…
19989    PROPN PROPN PUNCT PROPN VERB VERB PROPN ADP NU…
27388    PROPN NOUN VERB PROPN PUNCT PROPN VERB PROPN N…
2201     PROPN PROPN PROPN NUM ADP PROPN PROPN PROPN AD…
19867    ADJ PROPN PUNCT PROPN ADP PROPN NUM PUNCT NUM …
18790    PROPN PART PROPN PUNCT PROPN PART NUM PROPN PR…
18553    NOUN ADP PROPN PROPN PROPN PROPN PROPN PART PR…
25688    PROPN PART PROPN PROPN VERB PROPN ADP PROPN PR…
18711    PROPN PROPN PART VERB DET PROPN PROPN PUNCT VE…
28233    PROPN PROPN VERB PROPN PROPN PART PROPN ADP PR…
23720    PUNCT ADP NUM NOUN PRON VERB VERB PUNCT PUNCT …
2642     PROPN PROPN PROPN PROPN ADP PROPN VERB ADP PRO…
9161     PROPN PROPN PUNCT PRON VERB PROPN PROPN PART N…
13332    PROPN NOUN PUNCT ADV ADV VERB PRON VERB DET PR…
10227    PROPN PROPN PART PROPN PROPN PROPN PROPN PROPN…
15856    PROPN PROPN NUM PUNCT PROPN VERB PROPN DET ADJ…
16990    PROPN CCONJ PROPN PROPN PROPN PROPN ADP PROPN …
17590    ADJ NOUN ADP PROPN NUM SYM PROPN PROPN PROPN P…
15295    PROPN PUNCT PROPN PROPN PROPN PUNCT PROPN PUNC…
30775    PROPN PROPN VERB NOUN ADP NOUN ADP DET VERB NO…
17756    PROPN VERB ADJ PUNCT CCONJ PROPN VERB PRON PUN…
6451     PROPN CCONJ PROPN PROPN PART VERB PROPN PROPN …
26272    PROPN PROPN PUNCT PROPN VERB NOUN PUNCT NOUN N…
8193     PROPN PROPN PROPN PROPN PART VERB PROPN ADP PR…
10999    ADP PROPN PUNCT DET PROPN ADP PROPN PROPN PART…
Name: Text_pos, Length: 18669, dtype: object
```

[42]:
```
#Initialize the `tfidf_vectorizer`
tfidf_vectorizer = TfidfVectorizer(stop_words='english', ngram_range = (2,2))

# Fit and transform the training data
tfidf_train = tfidf_vectorizer.fit_transform(x_train.astype('str'))

# Transform the test set
tfidf_test = tfidf_vectorizer.transform(x_test.astype('str'))

pickle.dump(tfidf_train, open("tfidf_train.pickle", "wb"))

pickle.dump(tfidf_test, open("tfidf_test.pickle", "wb"))
```

[28]:
```
tfidf_vectorizer.get_feature_names()[-10:]
```

[28]:
```
['verb det',
 'verb intj',
 'verb noun',
 'verb num',
 'verb pron',
```

```
 'verb propn',
 'verb punct',
 'verb space',
 'verb sym',
 'verb verb']
```

[29]:
```
tfidf_train
```

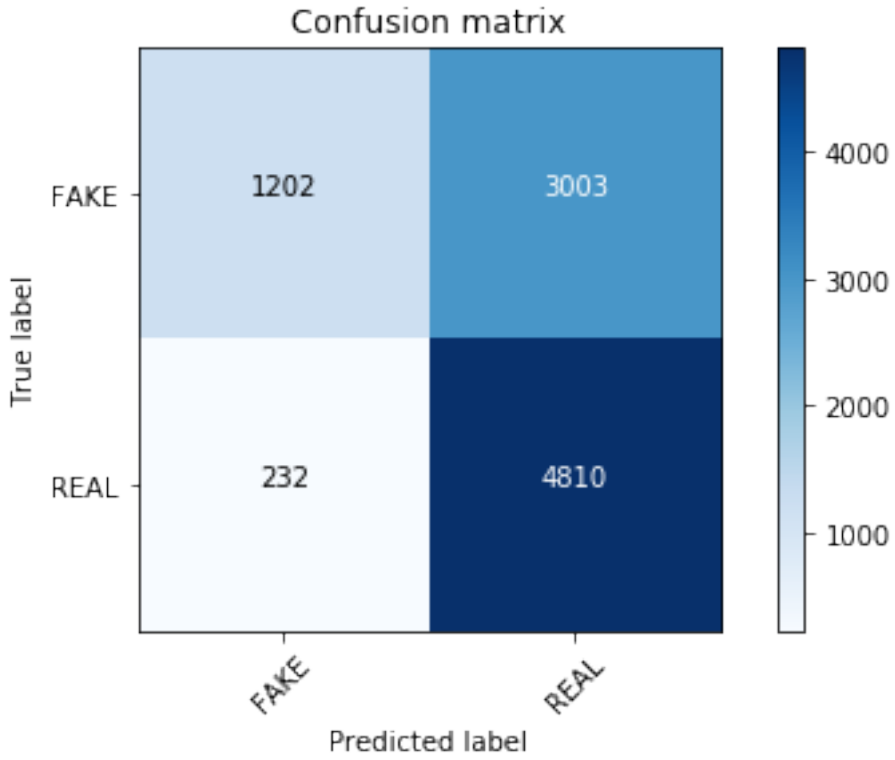
[29]:
```
<18772x196 sparse matrix of type '<class 'numpy.float64'>'
        with 1612837 stored elements in Compressed Sparse Row format>
```

[30]:
```
clf = MultinomialNB()
clf.fit(tfidf_train, y_train)
pickle.dump(clf, open('pos_nb', 'wb'))
pred = clf.predict(tfidf_test)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with Multinomial Naive Bayes:   %0.3f" % score)
```

```
Accuracy with Multinomial Naive Bayes:   0.665
```

[43]:
```
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
Confusion matrix
```

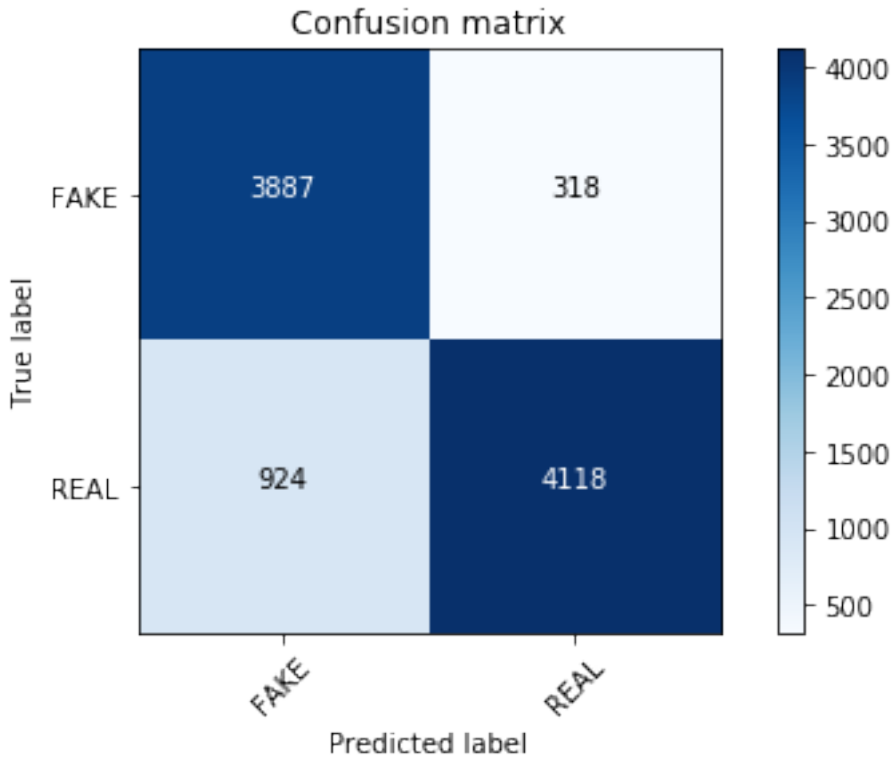


[44]:
```
clf = RandomForestClassifier()
clf.fit(tfidf_train, y_train)
pickle.dump(clf, open('pos_rf', 'wb'))
pred = clf.predict(tfidf_test)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with RandomForestClassifier:   %0.3f" % score)
```

```
Accuracy with RandomForestClassifier:   0.866
```

[45]:
```
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
Confusion matrix
```

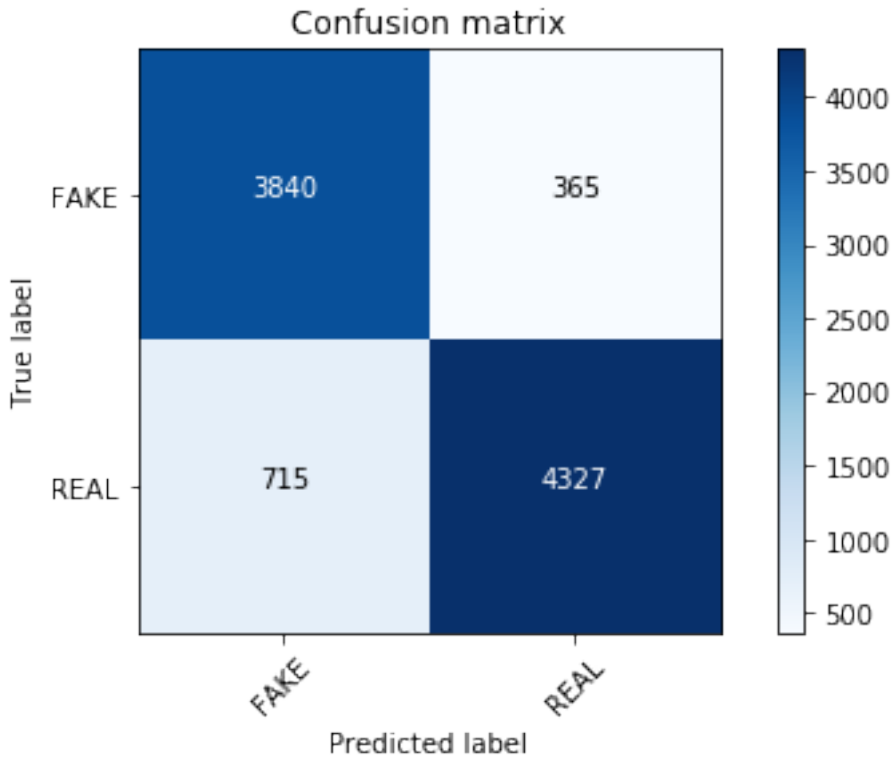


[46]:
```
clf = GradientBoostingClassifier()
clf.fit(tfidf_train, y_train)
pickle.dump(clf, open('pos_gb', 'wb'))
pred = clf.predict(tfidf_test)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with Gradient Boosting:   %0.3f" % score)
```

```
Accuracy with Gradient Boosting:   0.883
```

[47]:
```
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
Confusion matrix
```



[31]:
```
#Getting the score of semantic categories generated by Empath of each article␣
 ↪and generating a tfidf vector of the unigrams
lexicon = Empath()
semantic = []
cnt = 0
df["Text"] = df["Headline"].map(str) + df["Body"]

for article in df['Text']:
    if article == '':
        continue
    cnt+=1
    d = lexicon.analyze(article, normalize = False)
    x = []
    for key, value in d.items():
        x.append(value)
    x = np.asarray(x)
```

```
    semantic.append(x)
df['Semantic'] = semantic
```

[32]:
```
categories = []
a = lexicon.analyze("")
for key, value in a.items():
    categories.append(key)
categories
```

[32]:
```
['exercise',
 'horror',
 'hearing',
 'college',
 'science',
 'car',
 'government',
 'toy',
 'rural',
 'poor',
 'strength',
 'music',
 'weather',
 'payment',
 'disappointment',
 'dispute',
 'leader',
 'trust',
 'shame',
 'help',
 'musical',
 'appearance',
 'breaking',
 'ocean',
 'clothing',
 'farming',
 'traveling',
 'fabric',
 'social_media',
 'nervousness',
 'pride',
 'joy',
 'achievement',
 'zest',
 'writing',
 'ridicule',
 'anticipation',
 'suffering',
```

```
'leisure',
'driving',
'party',
'occupation',
'sympathy',
'reading',
'power',
'banking',
'communication',
'healing',
'ancient',
'masculine',
'emotional',
'affection',
'messaging',
'cooking',
'terrorism',
'swimming',
'confusion',
'death',
'negative_emotion',
'sound',
'valuable',
'beach',
'law',
'beauty',
'anger',
'superhero',
'sailing',
'restaurant',
'family',
'cold',
'rage',
'economics',
'cleaning',
'play',
'exasperation',
'exotic',
'weapon',
'positive_emotion',
'ugliness',
'royalty',
'speaking',
'dominant_personality',
'politics',
'hygiene',
'feminine',
```

```
'alcohol',
'religion',
'violence',
'envy',
'medical_emergency',
'fight',
'animal',
'domestic_work',
'war',
'contentment',
'phone',
'shape_and_size',
'timidity',
'independence',
'business',
'torment',
'internet',
'heroic',
'vacation',
'crime',
'gain',
'philosophy',
'divine',
'giving',
'money',
'love',
'home',
'monster',
'sexual',
'blue_collar_job',
'meeting',
'dance',
'stealing',
'noise',
'sadness',
'school',
'order',
'fire',
'plant',
'neglect',
'vehicle',
'ship',
'smell',
'legend',
'weakness',
'worship',
'fashion',
```

```
'negotiate',
'movement',
'journalism',
'sleep',
'fear',
'celebration',
'programming',
'children',
'work',
'childish',
'medieval',
'night',
'friends',
'urban',
'dominant_heirarchical',
'body',
'surprise',
'youth',
'hipster',
'aggression',
'wealthy',
'competing',
'shopping',
'lust',
'furniture',
'warmth',
'wedding',
'art',
'optimism',
'real_estate',
'fun',
'office',
'military',
'irritability',
'water',
'cheerfulness',
'sports',
'pain',
'politeness',
'technology',
'injury',
'health',
'prison',
'anonymity',
'computer',
'disgust',
'hate',
```

```
 'pet',
 'eating',
 'white_collar_job',
 'liquid',
 'kill',
 'attractive',
 'hiking',
 'magic',
 'air_travel',
 'deception',
 'morning',
 'swearing_terms',
 'tourism',
 'listen',
 'tool']
```

[33]:
```
#TF-IDF vector by taking the score for a semantic class as its frequency.
sem = []
for i in range(df.shape[0]):
    a = []
    for j in range(len(semantic[0])):
        for k in range(int(semantic[i][j])):
            a.append(categories[j])
    b = " ".join(a)
    sem.append(b)
#print(len(sem))
df['Semantics'] = sem
df.to_pickle('Semantic.pkl')
```

[34]:
```
df = pd.read_pickle('Semantic.pkl')
print(df.columns)
print(df.shape)
```

```
Index(['Headline', 'Body', 'Label', 'headline_length', 'body_length',
       'Body_pos', 'Text_pos', 'Text', 'Semantic', 'Semantics'],
      dtype='object')
(27865, 10)
```

[45]:
```
y = df.Label
y = y.astype('str')
x_train, x_test, y_train, y_test = train_test_split(df['Semantics'],y,␣
 ↪test_size=0.33)
x_train
```

[45]:
```
11942    [3.0, 1.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, …
16974    [0.0, 0.0, 0.0, 2.0, 5.0, 0.0, 2.0, 1.0, 0.0, …
27421    [10.0, 6.0, 0.0, 7.0, 5.0, 2.0, 0.0, 10.0, 0.0…
```

```
2099     [0.0, 3.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, …
456      [1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 1.0, …
20244    [4.0, 10.0, 0.0, 1.0, 2.0, 3.0, 1.0, 37.0, 0.0…
18196    [2.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
4110     [6.0, 4.0, 1.0, 33.0, 3.0, 1.0, 0.0, 10.0, 0.0…
7468     [3.0, 0.0, 0.0, 0.0, 4.0, 3.0, 1.0, 0.0, 2.0, …
16502    [0.0, 0.0, 0.0, 5.0, 2.0, 0.0, 0.0, 0.0, 0.0, …
18146    [3.0, 4.0, 0.0, 9.0, 1.0, 2.0, 1.0, 5.0, 2.0, …
6491     [0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, …
111      [1.0, 1.0, 1.0, 1.0, 0.0, 3.0, 0.0, 0.0, 0.0, …
9181     [7.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 1.0, …
30433    [8.0, 8.0, 4.0, 3.0, 2.0, 0.0, 0.0, 2.0, 0.0, …
14557    [6.0, 0.0, 0.0, 2.0, 5.0, 1.0, 2.0, 0.0, 2.0, …
4720     [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
3608     [2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 3.0, 0.0, …
13548    [1.0, 3.0, 1.0, 2.0, 10.0, 0.0, 0.0, 0.0, 2.0,…
2625     [1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, …
18943    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
718      [0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, …
27068    [0.0, 3.0, 9.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, …
19956    [0.0, 1.0, 0.0, 1.0, 2.0, 1.0, 0.0, 0.0, 0.0, …
10544    [1.0, 2.0, 3.0, 1.0, 2.0, 2.0, 4.0, 0.0, 2.0, …
23551    [5.0, 21.0, 2.0, 0.0, 2.0, 2.0, 2.0, 0.0, 2.0,…
7966     [1.0, 3.0, 1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, …
5769     [8.0, 1.0, 0.0, 4.0, 5.0, 3.0, 1.0, 0.0, 0.0, …
2996     [1.0, 3.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, …
18231    [1.0, 1.0, 0.0, 1.0, 1.0, 2.0, 1.0, 0.0, 0.0, …
                                  …
3976     [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
16011    [5.0, 3.0, 1.0, 2.0, 3.0, 2.0, 1.0, 1.0, 0.0, …
30158    [1.0, 0.0, 0.0, 4.0, 0.0, 0.0, 1.0, 0.0, 0.0, …
24756    [1.0, 1.0, 1.0, 29.0, 4.0, 12.0, 3.0, 0.0, 0.0…
9744     [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, …
6742     [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
29164    [0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 2.0, …
22397    [1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, …
23677    [0.0, 1.0, 0.0, 0.0, 3.0, 0.0, 0.0, 1.0, 1.0, …
17697    [3.0, 5.0, 0.0, 2.0, 6.0, 1.0, 0.0, 0.0, 0.0, …
9816     [0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, …
25184    [0.0, 1.0, 1.0, 1.0, 0.0, 0.0, 3.0, 0.0, 0.0, …
10549    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
28306    [4.0, 3.0, 1.0, 11.0, 2.0, 4.0, 1.0, 5.0, 18.0…
9630     [4.0, 2.0, 0.0, 2.0, 1.0, 0.0, 1.0, 3.0, 1.0, …
22319    [0.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, …
22338    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
3762     [3.0, 1.0, 0.0, 3.0, 2.0, 0.0, 0.0, 0.0, 1.0, …
713      [3.0, 3.0, 0.0, 14.0, 0.0, 1.0, 0.0, 0.0, 0.0,…
```

```
19414    [0.0, 2.0, 0.0, 1.0, 4.0, 2.0, 0.0, 0.0, 0.0, …
20956    [6.0, 2.0, 3.0, 4.0, 3.0, 0.0, 0.0, 0.0, 1.0, …
14044    [2.0, 3.0, 0.0, 6.0, 0.0, 0.0, 0.0, 0.0, 1.0, …
29172    [1.0, 2.0, 4.0, 1.0, 0.0, 0.0, 0.0, 2.0, 0.0, …
29737    [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, …
12776    [1.0, 4.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, …
2003     [0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 1.0, 1.0, 2.0, …
4296     [1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, …
18732    [6.0, 8.0, 2.0, 0.0, 2.0, 0.0, 0.0, 1.0, 3.0, …
10326    [4.0, 2.0, 4.0, 6.0, 3.0, 1.0, 0.0, 1.0, 2.0, …
3600     [4.0, 3.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.0, 0.0, …
Name: Semantic, Length: 18669, dtype: object
```

[ ]:
```
print(type(x_train))
print(x_train.shape)
```

[ ]:
```
#Initialize the `tfidf_vectorizer`
tfidf2_vectorizer = TfidfVectorizer(stop_words='english', ngram_range = (1,1))

# Fit and transform the training data
tfidf2_train = tfidf2_vectorizer.fit_transform(x_train.astype('str'))

# Transform the test set
tfidf2_test = tfidf2_vectorizer.transform(x_test.astype('str'))

pickle.dump(tfidf2_train, open("tfidf2_train.pickle", "wb"))

pickle.dump(tfidf2_test, open("tfidf2_test.pickle", "wb"))
```

[ ]:
```
clf = MultinomialNB()
#type(x_train.tolist())
clf.fit(x_train.tolist(), y_train)
pickle.dump(clf, open('sem_nb', 'wb'))
pred = clf.predict(x_test.tolist())
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with Multinomial Naive Bayes:   %0.3f" % score)
```

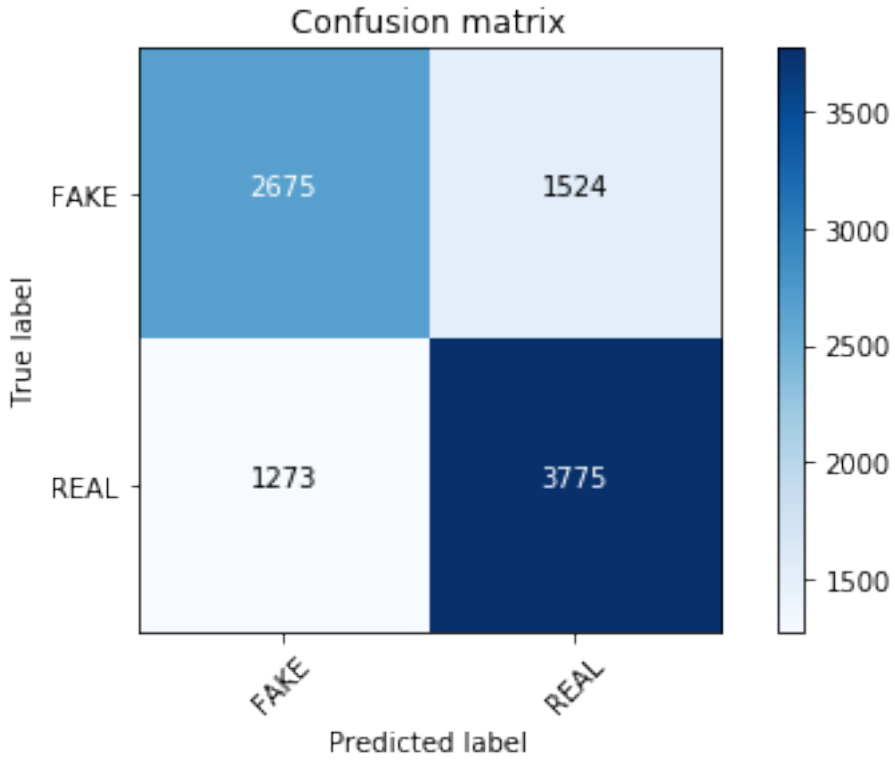
[ ]:
```
clf = RandomForestClassifier()
clf.fit(x_train.tolist(), y_train)
pickle.dump(clf, open('sem_rf', 'wb'))
pred = clf.predict(x_test.tolist())
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with RandomForestClassifier:   %0.3f" % score)
```

[ ]:
```
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

[ ]:
```
clf = GradientBoostingClassifier()
clf.fit(x_train.tolist(), y_train)
pickle.dump(clf, open('sem_gb', 'wb'))
pred = clf.predict(x_test.tolist())
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with Gradient Boosting:   %0.3f" % score)
```

[58]:
```
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
Confusion matrix
```



[ ]:
```
#Combining the 3 feature vectors
import scipy.sparse as sp
# ui = sp.vstack(tfidf_train, tfidf1_train)
# yu = tfidf_train.data.tolist()
# yu.append(tfidf1_train.tolist())
# test = tfidf_test.data.tolist() + x_test.tolist()
#print(type(tfidf_train), tfidf_train.shape)
#print(type(tfidf1_train), tfidf1_train.shape)
# print(type(x_train), x_train.shape)
diff_n_rows = tfidf_train.shape[0] - tfidf1_train.shape[0]
```

```
Xb_new = sp.vstack((tfidf1_train, sp.csr_matrix((diff_n_rows, tfidf1_train.
 ↪shape[1]))))
#where diff_n_rows is the difference of the number of rows between Xa and Xb

c = sp.hstack((tfidf_train, Xb_new))

diff_n_rows = c.shape[0] - tfidf2_train.shape[0]

Xb_new = sp.vstack((tfidf2_train, sp.csr_matrix((diff_n_rows, tfidf2_train.
 ↪shape[1]))))
#where diff_n_rows is the difference of the number of rows between Xa and Xb

X = sp.hstack((c, Xb_new))
X

dif_n_rows = tfidf_test.shape[0] - tfidf1_test.shape[0]

Xb_ne = sp.vstack((tfidf1_test, sp.csr_matrix((dif_n_rows, tfidf1_test.
 ↪shape[1]))))
#where diff_n_rows is the difference of the number of rows between Xa and Xb

d = sp.hstack((tfidf_test, Xb_ne))

dif_n_rows = d.shape[0] - tfidf2_test.shape[0]

Xb_ne = sp.vstack((tfidf2_test, sp.csr_matrix((dif_n_rows, tfidf2_test.
 ↪shape[1]))))
#where diff_n_rows is the difference of the number of rows between Xa and Xb

Y = sp.hstack((d, Xb_ne))
```

[ ]:
```
clf = MultinomialNB()
#print(type(train), type(y_train.tolist()))
clf.fit(X, y_train)
pickle.dump(clf, open('pos_sem_nb', 'wb'))
pred = clf.predict(Y)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with Multinomial Naive Bayes:   %0.3f" % score)
```

[ ]:
```
clf = RandomForestClassifier()
clf.fit(X, y_train)
pickle.dump(clf, open('pos_sem_rf', 'wb'))
pred = clf.predict(Y)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with RandomForestClassifier:   %0.3f" % score)
```

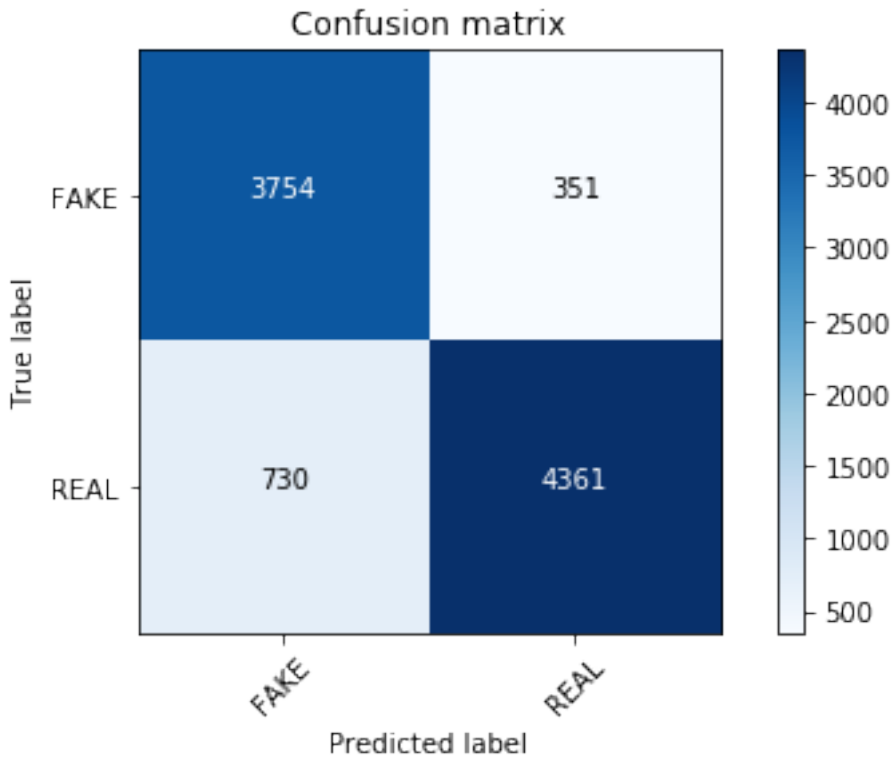
[ ]:
```
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

[174]:
```
clf = GradientBoostingClassifier()
clf.fit(X, y_train)
pickle.dump(clf, open('pos_sem_gb', 'wb'))
pred = clf.predict(Y)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with Gradient Boosting:   %0.3f" % score)
```

```
Accuracy with Gradient Boosting:   0.882
```

[175]:
```
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
Confusion matrix
```



[ ]:

[ ]:

[164]:
```
#Directly loading the final dateframe by loading the pickle file from the␣
 ↪previously saved pickle file
df = pd.read_pickle('Semantic.pkl')
print(df.columns)
print(df.shape)
```

```
Index(['Headline', 'Body', 'Label', 'headline_length', 'body_length',
       'Body_pos', 'Text_pos', 'Text', 'Semantic', 'Semantics'],
      dtype='object')
(27865, 10)
```

[165]:
```
y = df.Label
x_train, x_test, y_train, y_test = train_test_split(df,y, test_size=0.33)
```

[166]:
```
x_train_text = x_train['Text']
x_test_text = x_test['Text']
x_train_text_pos = x_train['Text_pos']
x_test_text_pos = x_test['Text_pos']
x_train_semantics = x_train['Semantics']
x_test_semantics = x_test['Semantics']
```

[167]:
```
#Tf-idf Bigrams
#Initialize the `tfidf_vectorizer`
tfidf_vectorizer = TfidfVectorizer(stop_words='english', ngram_range = (2,2),␣
 ↪max_features = 20000)

# Fit and transform the training data
tfidf1_train = tfidf_vectorizer.fit_transform(x_train_text.astype('str'))

# Transform the test set
tfidf1_test = tfidf_vectorizer.transform(x_test_text.astype('str'))

pickle.dump(tfidf1_train, open("tfidf1_train.pickle", "wb"))

pickle.dump(tfidf1_test, open("tfidf1_test.pickle", "wb"))
```

[168]:
```
#POS
#Initialize the `tfidf_vectorizer`
tfidf_vectorizer = TfidfVectorizer(stop_words='english', ngram_range = (2,2))

# Fit and transform the training data
tfidf_train = tfidf_vectorizer.fit_transform(x_train_text_pos.astype('str'))

# Transform the test set
tfidf_test = tfidf_vectorizer.transform(x_test_text_pos.astype('str'))

pickle.dump(tfidf_train, open("tfidf_train.pickle", "wb"))
```

```
pickle.dump(tfidf_test, open("tfidf_test.pickle", "wb"))
```

[169]:
```
#Initialize the `tfidf_vectorizer`
tfidf_vectorizer = TfidfVectorizer(stop_words='english', ngram_range = (1,1))

# Fit and transform the training data
tfidf2_train = tfidf_vectorizer.fit_transform(x_train_semantics.astype('str'))

# Transform the test set
tfidf2_test = tfidf_vectorizer.transform(x_test_semantics.astype('str'))

pickle.dump(tfidf2_train, open("tfidf_train.pickle", "wb"))

pickle.dump(tfidf2_test, open("tfidf_test.pickle", "wb"))
```

[170]:
```
ttf1_train = tfidf1_train
ttf1_test = tfidf1_test
ttf_train = tfidf_train
ttf_test = tfidf_test
ttf2_train = tfidf2_train
ttf2_test = tfidf2_test
```

[218]:
```
#Giving weights to each of the 3 feature vectors generated
big_w = 0.35
synt_w = 0.5
sem_w = 0.15
big_w *= 3
synt_w *= 3
sem_w *= 3
tfidf1_train = big_w*ttf1_train
tfidf1_test = big_w*ttf1_test
tfidf_train = synt_w*ttf_train
tfidf_test = synt_w*ttf_test
tfidf2_train = sem_w*ttf2_train
tfidf2_test = sem_w*ttf2_test
```

[219]:
```
import scipy.sparse as sp
# ui = sp.vstack(tfidf_train, tfidf1_train)
# yu = tfidf_train.data.tolist()
# yu.append(tfidf1_train.tolist())
# test = tfidf_test.data.tolist() + x_test.tolist()
#print(type(tfidf_train), tfidf_train.shape)
#print(type(tfidf1_train), tfidf1_train.shape)
# print(type(x_train), x_train.shape)
diff_n_rows = tfidf_train.shape[0] - tfidf1_train.shape[0]
```

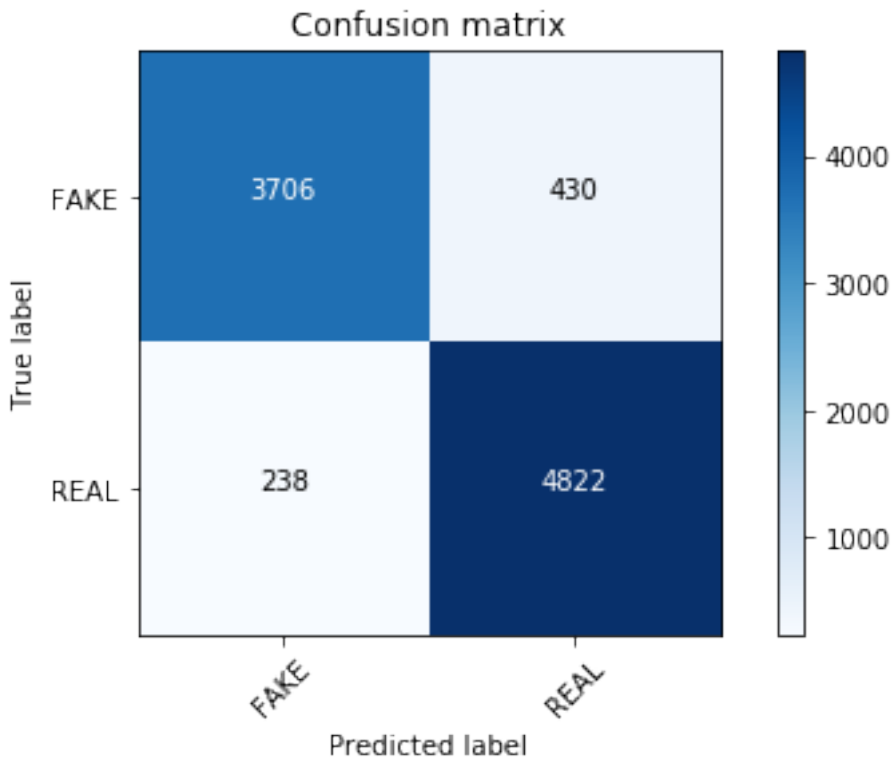
```
Xb_new = sp.vstack((tfidf1_train, sp.csr_matrix((diff_n_rows, tfidf1_train.
 ↪shape[1]))))
#where diff_n_rows is the difference of the number of rows between Xa and Xb

c = sp.hstack((tfidf_train, Xb_new))

diff_n_rows = c.shape[0] - tfidf2_train.shape[0]

Xb_new = sp.vstack((tfidf2_train, sp.csr_matrix((diff_n_rows, tfidf2_train.
 ↪shape[1]))))
#where diff_n_rows is the difference of the number of rows between Xa and Xb

X = sp.hstack((c, Xb_new))
X

dif_n_rows = tfidf_test.shape[0] - tfidf1_test.shape[0]

Xb_ne = sp.vstack((tfidf1_test, sp.csr_matrix((dif_n_rows, tfidf1_test.
 ↪shape[1]))))
#where diff_n_rows is the difference of the number of rows between Xa and Xb

d = sp.hstack((tfidf_test, Xb_ne))

dif_n_rows = d.shape[0] - tfidf2_test.shape[0]

Xb_ne = sp.vstack((tfidf2_test, sp.csr_matrix((dif_n_rows, tfidf2_test.
 ↪shape[1]))))
#where diff_n_rows is the difference of the number of rows between Xa and Xb

Y = sp.hstack((d, Xb_ne))
```

[220]:
```
clf = MultinomialNB()
#type(x_train.tolist())
clf.fit(X, y_train)
pickle.dump(clf, open('bi_pos_sem_nb', 'wb'))
pred = clf.predict(Y)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with Multinomial Naive Bayes:   %0.3f" % score)
```

```
Accuracy with Multinomial Naive Bayes:   0.927
```

[221]:
```
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
Confusion matrix
```

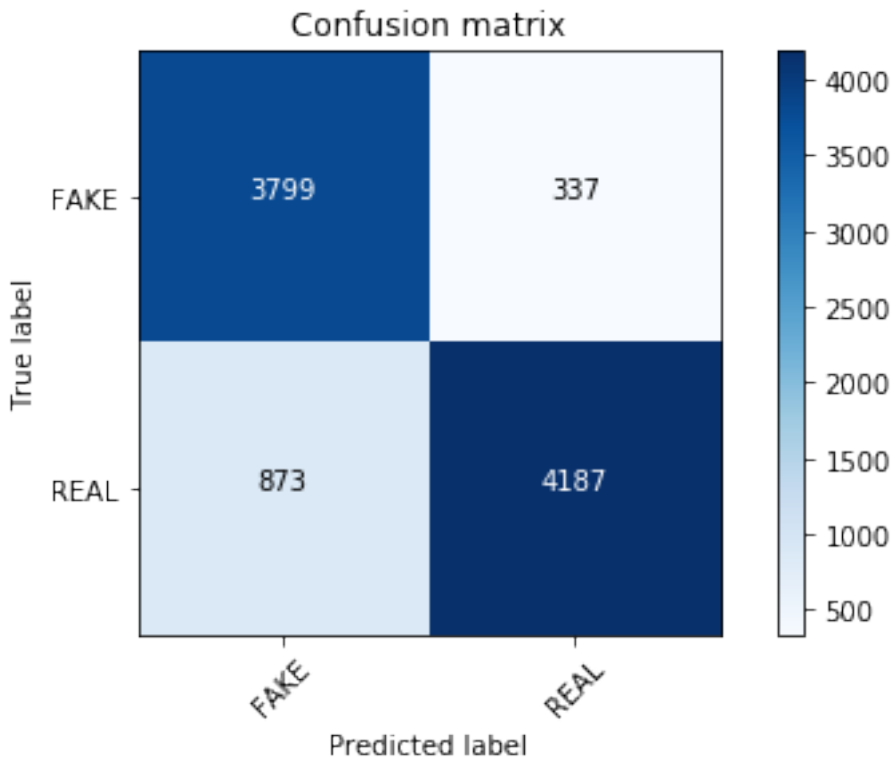


[222]:
```
clf = RandomForestClassifier()
clf.fit(X, y_train)
pickle.dump(clf, open('bi_pos_sem_rf', 'wb'))
pred = clf.predict(Y)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with RandomForestClassifier:   %0.3f" % score)
```

```
Accuracy with RandomForestClassifier:   0.868
```

[223]:
```
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
Confusion matrix
```

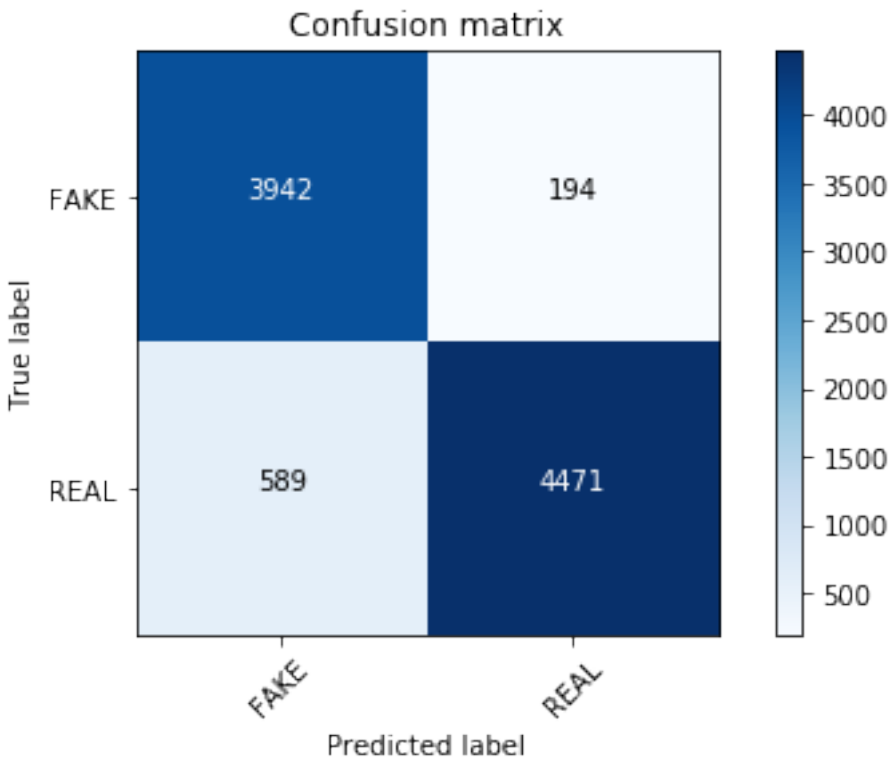


[214]:
```
clf = GradientBoostingClassifier()
clf.fit(X, y_train)
pickle.dump(clf, open('pos_gb', 'wb'))
pred = clf.predict(Y)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with Gradient Boosting:   %0.3f" % score)
```

```
Accuracy with Gradient Boosting:   0.915
```

[215]:
```
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
Confusion matrix
```



[ ]:

[ ]:

[ ]:
```
#For testing any new article
a = (open('a.txt'))
x_test = a.read()
```

[ ]:
```
#Tf-idf Bigrams
#Initialize the `tfidf_vectorizer`
tfidf_vectorizer = TfidfVectorizer(stop_words='english', ngram_range = (2,2),␣
 ↪max_features = 20000)

# Fit and transform the training data
tfidf1_train = tfidf_vectorizer.fit_transform(x_train_text.astype('str'))

# Transform the test set
tfidf1_test = tfidf_vectorizer.transform([x_test])
```

[ ]:
```
nlp = spacy.load('en_core_web_sm')
x = []
text_new = []
```

```
doc = nlp(x_test)
for token in doc:
    text_new.append(token.pos_)
txt = ' '.join(text_new)
txt
```

[ ]:
```
#Tf-idf Bigrams
#Initialize the `tfidf_vectorizer`
tfidf_vectorizer = TfidfVectorizer(stop_words='english', ngram_range = (2,2))

# Fit and transform the training data
tfidf_train = tfidf_vectorizer.fit_transform(x_train_text_pos.astype('str'))

# Transform the test set
tfidf_test = tfidf_vectorizer.transform([x_test])
```

[ ]:
```
categories = []
a = lexicon.analyze("")
for key, value in a.items():
    categories.append(key)
categories
lexicon = Empath()
semantic = []
cnt = 0
d = lexicon.analyze(x_test)
d
sem = []
for key,value in d.items() :
    sem.append(value)
a = []
for j in range(len(sem)):
    for k in range(int(sem[j])):
        a.append(categories[j])
    b = " ".join(a)
b
```

[ ]:
```
#Initialize the `tfidf_vectorizer`
tfidf_vectorizer = TfidfVectorizer(stop_words='english', ngram_range = (1,1))

# Fit and transform the training data
tfidf2_train = tfidf_vectorizer.fit_transform(x_train_semantics.astype('str'))

# Transform the test set
tfidf2_test = tfidf_vectorizer.transform([b])
```

[ ]:
```
import scipy.sparse as sp
# ui = sp.vstack(tfidf_train, tfidf1_train)
```

```
# yu = tfidf_train.data.tolist()
# yu.append(tfidf1_train.tolist())
# test = tfidf_test.data.tolist() + x_test.tolist()
#print(type(tfidf_train), tfidf_train.shape)
#print(type(tfidf1_train), tfidf1_train.shape)
# print(type(x_train), x_train.shape)
diff_n_rows = tfidf_train.shape[0] - tfidf1_train.shape[0]

Xb_new = sp.vstack((tfidf1_train, sp.csr_matrix((diff_n_rows, tfidf1_train.
 ↪shape[1]))))
#where diff_n_rows is the difference of the number of rows between Xa and Xb

c = sp.hstack((tfidf_train, Xb_new))

diff_n_rows = c.shape[0] - tfidf2_train.shape[0]

Xb_new = sp.vstack((tfidf2_train, sp.csr_matrix((diff_n_rows, tfidf2_train.
 ↪shape[1]))))
#where diff_n_rows is the difference of the number of rows between Xa and Xb

X = sp.hstack((c, Xb_new))
X

dif_n_rows = tfidf_test.shape[0] - tfidf1_test.shape[0]

Xb_ne = sp.vstack((tfidf1_test, sp.csr_matrix((dif_n_rows, tfidf1_test.
 ↪shape[1]))))
#where diff_n_rows is the difference of the number of rows between Xa and Xb

d = sp.hstack((tfidf_test, Xb_ne))

dif_n_rows = d.shape[0] - tfidf2_test.shape[0]

Xb_ne = sp.vstack((tfidf2_test, sp.csr_matrix((dif_n_rows, tfidf2_test.
 ↪shape[1]))))
#where diff_n_rows is the difference of the number of rows between Xa and Xb

Y = sp.hstack((d, Xb_ne))
```

[ ]:
```
clf = MultinomialNB()
#type(x_train.tolist())
clf.fit(X, y_train)
clf.predict(Y)
```

[ ]:
```

```

[ ]:
```

```

[ ]:

[ ]: