fake-news-n

April 2, 2024

[41]:
```
import pandas as pd
import numpy as np
import itertools
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer,␣
↪HashingVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
import matplotlib.pyplot as plt
```

[42]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib

C:\Users\Pooja Jayaprakash\Anaconda3\lib\site-
packages\IPython\core\magics\pylab.py:160: UserWarning: pylab import has
clobbered these variables: ['clf', 'cm']
`%matplotlib` prevents importing * from pylab and numpy
  "\n`%matplotlib` prevents importing * from pylab and numpy"
```

[43]:
```
df = pd.read_csv('fake_or_real_news.csv')
```

[44]:
```
df.shape
```

[44]:
```
(6335, 4)
```

[45]:
```
df = df.set_index('Unnamed: 0')
```

[7]:
```
df.head()
```

[7]:

Unnamed: 0	title	\
8476	You Can Smell Hillary's Fear	
10294	Watch The Exact Moment Paul Ryan Committed Pol…	
3608	Kerry to go to Paris in gesture of sympathy	
10142	Bernie supporters on Twitter erupt in anger ag…	
875	The Battle of New York: Why This Primary Matters	

```
                                                        text label
Unnamed: 0
8476        Daniel Greenfield, a Shillman Journalism Fello…  FAKE
10294       Google Pinterest Digg Linkedin Reddit Stumbleu…  FAKE
3608        U.S. Secretary of State John F. Kerry said Mon…  REAL
10142       - Kaydee King (@KaydeeKing) November 9, 2016 T…  FAKE
875         It's primary day in New York and front-runners…  REAL
```

[8]:
```
y = df.label
```

[9]:
```
df = df.drop('label', axis=1)
```

[10]:
```
X_train, X_test, y_train, y_test = train_test_split(df['text'], y, test_size=0.
↪33, random_state=53)
```

[11]:
```
count_vectorizer = CountVectorizer(stop_words='english')
count_train = count_vectorizer.fit_transform(X_train)
count_test = count_vectorizer.transform(X_test)
```

[12]:
```
tfidf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
tfidf_train = tfidf_vectorizer.fit_transform(X_train)
tfidf_test = tfidf_vectorizer.transform(X_test)
```

[13]:
```
tfidf_vectorizer.get_feature_names()[-10:]
```

[13]:
```
[' ', '  ', ' ', ' ', ' ', '    ', ' ', '  ', '    ', ' ade']
```

[14]:
```
count_vectorizer.get_feature_names()[:10]
```

[14]:
```
['00',
 '000',
 '0000',
 '00000031',
 '000035',
 '00006',
 '0001',
 '0001pt',
 '000ft',
 '000km']
```

[15]:
```
count_df = pd.DataFrame(count_train.A, columns=count_vectorizer.
↪get_feature_names())
```

[16]:
```
tfidf_df = pd.DataFrame(tfidf_train.A, columns=tfidf_vectorizer.
↪get_feature_names())
```

[17]:
```
difference = set(count_df.columns) - set(tfidf_df.columns)
difference
```

[17]:
```
set()
```

[18]:
```
print(count_df.equals(tfidf_df))
```

```
False
```

[19]:
```
count_df.head()
```

[19]:

	00	000	0000	00000031	000035	00006	0001	0001pt	000ft	000km	…	\
0	0	0	0	0	0	0	0	0	0	0	…	
1	0	0	0	0	0	0	0	0	0	0	…	
2	0	0	0	0	0	0	0	0	0	0	…	
3	0	0	0	0	0	0	0	0	0	0	…	
4	0	0	0	0	0	0	0	0	0	0	…	

							ade		
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0

```
[5 rows x 56922 columns]
```

[20]:
```
tfidf_df.head()
```

[20]:

	00	000	0000	00000031	000035	00006	0001	0001pt	000ft	000km	…	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	…	

							ade		
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```
[5 rows x 56922 columns]
```

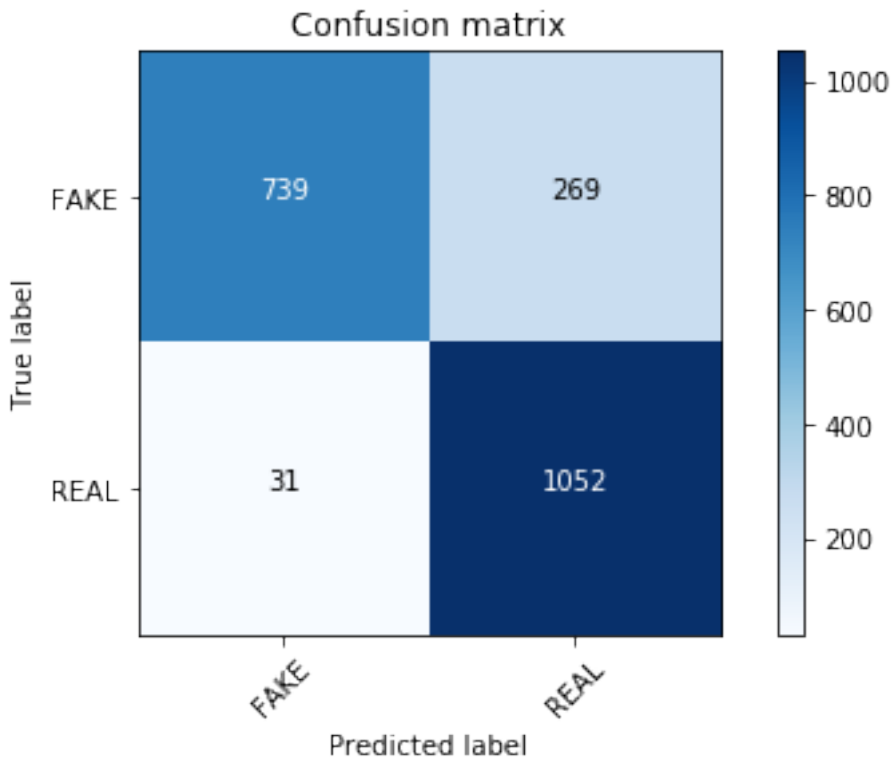
[21]:
```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
```

```
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    See full source and example:
    http://scikit-learn.org/stable/auto_examples/model_selection/
↪plot_confusion_matrix.html

    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    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, without normalization')

    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')
```

[22]:
```
clf = MultinomialNB()
```

[23]:
```
clf.fit(tfidf_train, y_train)
pred = clf.predict(tfidf_test)
score = metrics.accuracy_score(y_test, pred)
print("accuracy:   %0.3f" % score)
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

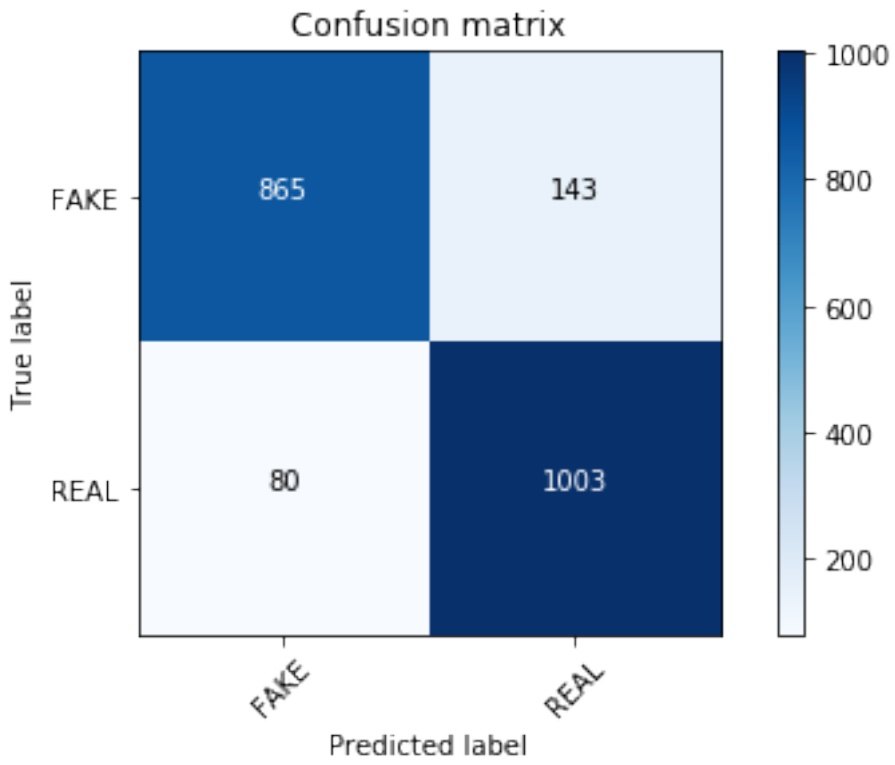
```
accuracy:   0.857
Confusion matrix, without normalization
```



[24]:
```
clf = MultinomialNB()
```

[25]:
```
clf.fit(count_train, y_train)
pred = clf.predict(count_test)
score = metrics.accuracy_score(y_test, pred)
print("accuracy:   %0.3f" % score)
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

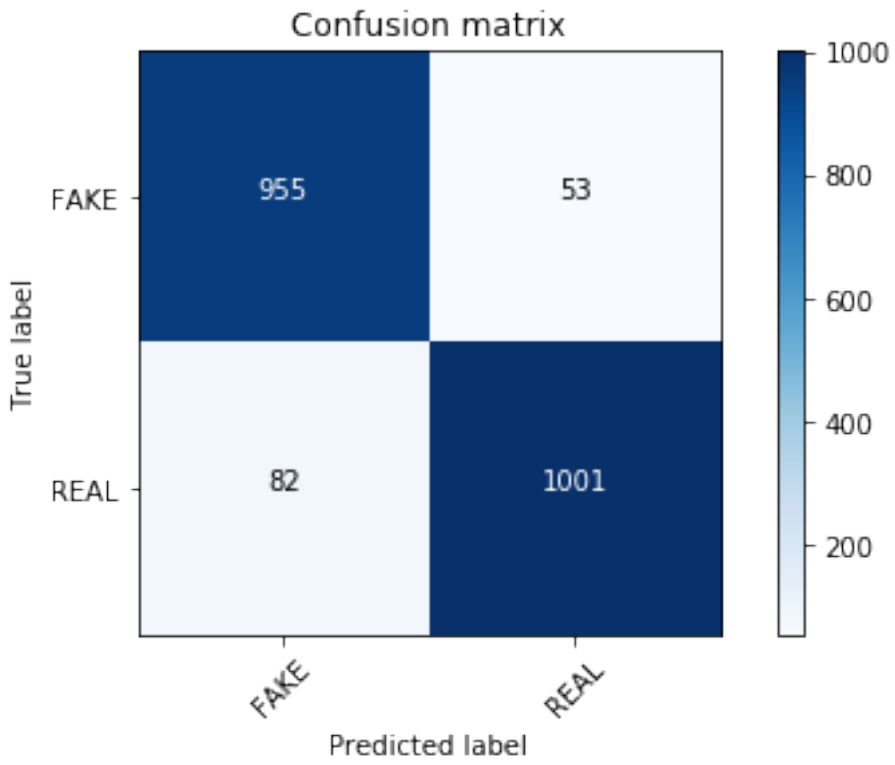
```
accuracy:   0.893
Confusion matrix, without normalization
```



[26]:
```
linear_clf = PassiveAggressiveClassifier(n_iter=50)
```

[27]:
```
linear_clf.fit(tfidf_train, y_train)
pred = linear_clf.predict(tfidf_test)
score = metrics.accuracy_score(y_test, pred)
print("accuracy:   %0.3f" % score)
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
C:\Users\Pooja Jayaprakash\Anaconda3\lib\site-
packages\sklearn\linear_model\stochastic_gradient.py:117: DeprecationWarning:
n_iter parameter is deprecated in 0.19 and will be removed in 0.21. Use max_iter
and tol instead.
  DeprecationWarning)

accuracy:   0.935
Confusion matrix, without normalization
```



[28]:
```
clf = MultinomialNB(alpha=0.1)
```

[29]:
```
last_score = 0
for alpha in np.arange(0,1,.1):
    nb_classifier = MultinomialNB(alpha=alpha)
    nb_classifier.fit(tfidf_train, y_train)
    pred = nb_classifier.predict(tfidf_test)
    score = metrics.accuracy_score(y_test, pred)
    if score > last_score:
        clf = nb_classifier
    print("Alpha: {:.2f} Score: {:.5f}".format(alpha, score))
```

```
C:\Users\Pooja Jayaprakash\Anaconda3\lib\site-
packages\sklearn\naive_bayes.py:472: UserWarning: alpha too small will result in
numeric errors, setting alpha = 1.0e-10
  'setting alpha = %.1e' % _ALPHA_MIN)

Alpha: 0.00 Score: 0.88140
Alpha: 0.10 Score: 0.89766
Alpha: 0.20 Score: 0.89383
Alpha: 0.30 Score: 0.89000
Alpha: 0.40 Score: 0.88570
Alpha: 0.50 Score: 0.88427
```

```
Alpha: 0.60 Score: 0.87470
Alpha: 0.70 Score: 0.87040
Alpha: 0.80 Score: 0.86609
Alpha: 0.90 Score: 0.85892
```

[30]:
```
def most_informative_feature_for_binary_classification(vectorizer, classifier,␣
↪n=100):
    """
    See: https://stackoverflow.com/a/26980472

    Identify most important features if given a vectorizer and binary␣
↪classifier. Set n to the number
    of weighted features you would like to show. (Note: current implementation␣
↪merely prints and does not
    return top classes.)
    """

    class_labels = classifier.classes_
    feature_names = vectorizer.get_feature_names()
    topn_class1 = sorted(zip(classifier.coef_[0], feature_names))[:n]
    topn_class2 = sorted(zip(classifier.coef_[0], feature_names))[-n:]

    for coef, feat in topn_class1:
        print(class_labels[0], coef, feat)

    print()

    for coef, feat in reversed(topn_class2):
        print(class_labels[1], coef, feat)


most_informative_feature_for_binary_classification(tfidf_vectorizer,␣
↪linear_clf, n=30)
```

```
FAKE -4.813872712383007 2016
FAKE -4.145150815798882 october
FAKE -4.037175642797679 hillary
FAKE -3.223408201528488 share
FAKE -2.918377251500853 november
FAKE -2.839059379503755 article
FAKE -2.6530274513128873 oct
FAKE -2.578051148084088 email
FAKE -2.520684443484875 print
FAKE -2.361054401362197 advertisement
FAKE -2.2007739326437634 election
FAKE -2.1902852431199142 source
FAKE -2.1893058315152247 mosul
```

```
FAKE -2.1644566242225296 war
FAKE -2.112554953242476 establishment
FAKE -2.066493430558853 nov
FAKE -2.013963713705571 wikileaks
FAKE -1.9576621542543828 podesta
FAKE -1.8111242184874567 daesh
FAKE -1.7687374574892525 com
FAKE -1.7258989873412311 corporate
FAKE -1.7082195249744814 ayotte
FAKE -1.7007681851430538 donald
FAKE -1.6851638363473116 26
FAKE -1.6828402797207036 uk
FAKE -1.6819821101727255 jewish
FAKE -1.6700399497428244 pipeline
FAKE -1.65913294709671 entire
FAKE -1.6185071490964729 snip
FAKE -1.598491423780342 watch

REAL 4.66264417474318 said
REAL 2.7236184701762043 tuesday
REAL 2.422159253048804 says
REAL 2.4161452443224043 gop
REAL 2.4121443992557134 friday
REAL 2.3390902541708223 islamic
REAL 2.2358422547834964 cruz
REAL 2.2171559835981967 conservative
REAL 2.166688638558029 marriage
REAL 2.1300010884950273 attacks
REAL 2.1246127655276164 debate
REAL 2.081116860656193 continue
REAL 2.0417942526773785 jobs
REAL 2.039143800771422 monday
REAL 1.9751301041758564 candidates
REAL 1.9690414425698783 rush
REAL 1.9559211971904231 presumptive
REAL 1.8772508327777875 sen
REAL 1.845206487187319 deal
REAL 1.8446433974022158 march
REAL 1.7901959994838128 sunday
REAL 1.7554442274646074 group
REAL 1.7534652842882525 convention
REAL 1.733605605200332 campaign
REAL 1.7194893083324563 recounts
REAL 1.6844655851617245 conservatives
REAL 1.6741186017499758 paris
REAL 1.6715649424212113 fox
REAL 1.6682588488059968 week
REAL 1.5936862914014653 attack
```

[31]:
```
feature_names = tfidf_vectorizer.get_feature_names()
```

[32]:
```
### Most real
sorted(zip(clf.coef_[0], feature_names), reverse=True)[:20]
```

[32]:
```
[(-6.257361214701582, 'trump'),
 (-6.494453094312678, 'said'),
 (-6.6539784739838845, 'clinton'),
 (-7.037944662867073, 'obama'),
 (-7.146539983381228, 'sanders'),
 (-7.215376008647511, 'president'),
 (-7.266562805741617, 'campaign'),
 (-7.2875931446681514, 'republican'),
 (-7.341118458599064, 'state'),
 (-7.341357110247905, 'cruz'),
 (-7.378312441985425, 'party'),
 (-7.44688067245789, 'new'),
 (-7.476288801154588, 'people'),
 (-7.547225599514773, 'percent'),
 (-7.5553074094582335, 'bush'),
 (-7.580150633909893, 'republicans'),
 (-7.5855405012652435, 'house'),
 (-7.634478172520314, 'voters'),
 (-7.648482443695299, 'rubio'),
 (-7.6734836186463795, 'states')]
```

[33]:
```
### Most fake
sorted(zip(clf.coef_[0], feature_names))[:20]
```

[33]:
```
[(-11.349866225220305, '0000'),
 (-11.349866225220305, '000035'),
 (-11.349866225220305, '0001'),
 (-11.349866225220305, '0001pt'),
 (-11.349866225220305, '000km'),
 (-11.349866225220305, '0011'),
 (-11.349866225220305, '006s'),
 (-11.349866225220305, '007'),
 (-11.349866225220305, '007s'),
 (-11.349866225220305, '008s'),
 (-11.349866225220305, '0099'),
 (-11.349866225220305, '00am'),
 (-11.349866225220305, '00p'),
 (-11.349866225220305, '00pm'),
 (-11.349866225220305, '014'),
 (-11.349866225220305, '015'),
 (-11.349866225220305, '018'),
 (-11.349866225220305, '01am'),
```

```
 (-11.349866225220305, '020'),
 (-11.349866225220305, '023')]
```

[34]:
```
tokens_with_weights = sorted(list(zip(feature_names, clf.coef_[0])))
```

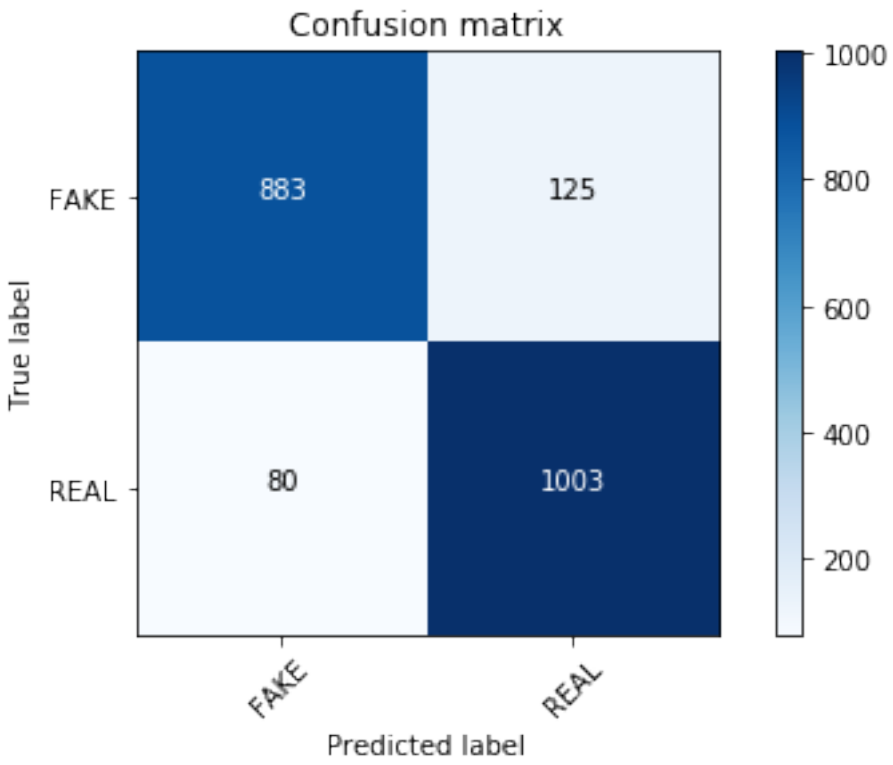
[35]:
```
hash_vectorizer = HashingVectorizer(stop_words='english', non_negative=True)
hash_train = hash_vectorizer.fit_transform(X_train)
hash_test = hash_vectorizer.transform(X_test)
```

```
C:\Users\Pooja Jayaprakash\Anaconda3\lib\site-
packages\sklearn\feature_extraction\hashing.py:94: DeprecationWarning: the
option non_negative=True has been deprecated in 0.19 and will be removed in
version 0.21.
  " in version 0.21.", DeprecationWarning)
C:\Users\Pooja Jayaprakash\Anaconda3\lib\site-
packages\sklearn\feature_extraction\hashing.py:94: DeprecationWarning: the
option non_negative=True has been deprecated in 0.19 and will be removed in
version 0.21.
  " in version 0.21.", DeprecationWarning)
C:\Users\Pooja Jayaprakash\Anaconda3\lib\site-
packages\sklearn\feature_extraction\hashing.py:94: DeprecationWarning: the
option non_negative=True has been deprecated in 0.19 and will be removed in
version 0.21.
  " in version 0.21.", DeprecationWarning)
```

[36]:
```
clf = MultinomialNB(alpha=.01)
```

[37]:
```
clf.fit(hash_train, y_train)
pred = clf.predict(hash_test)
score = metrics.accuracy_score(y_test, pred)
print("accuracy:   %0.3f" % score)
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

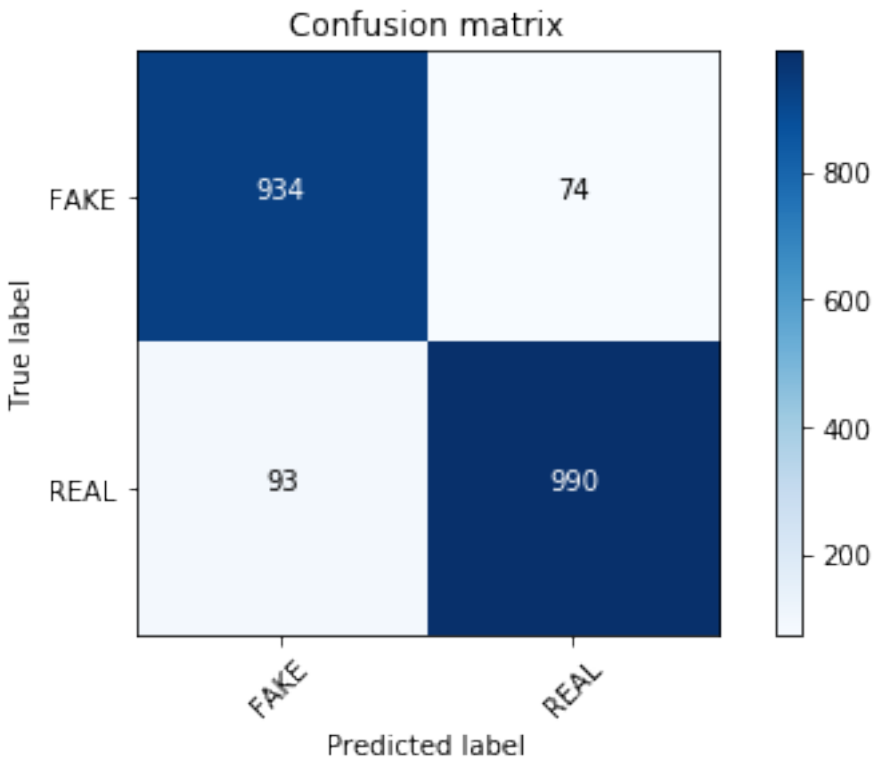
```
accuracy:   0.902
Confusion matrix, without normalization
```



[38]:
```
clf = PassiveAggressiveClassifier(n_iter=50)
```

[39]:
```
clf.fit(hash_train, y_train)
pred = clf.predict(hash_test)
score = metrics.accuracy_score(y_test, pred)
print("accuracy:   %0.3f" % score)
cm = metrics.confusion_matrix(y_test, pred, labels=['FAKE', 'REAL'])
plot_confusion_matrix(cm, classes=['FAKE', 'REAL'])
```

```
C:\Users\Pooja Jayaprakash\Anaconda3\lib\site-
packages\sklearn\linear_model\stochastic_gradient.py:117: DeprecationWarning:
n_iter parameter is deprecated in 0.19 and will be removed in 0.21. Use max_iter
and tol instead.
  DeprecationWarning)

accuracy:   0.920
Confusion matrix, without normalization
```


Confusion matrix
934
74
93
990
FAKE
REAL
True label
FAKE
REAL
Predicted label
800
600
400
200

[ ]: