In [274]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import sqlite3
%matplotlib inline
```

In [275]:
```
#import previous df from sqlite
con = sqlite3.connect('twitter_hate.db')
sql = """
SELECT * FROM tweets_nlp
"""
with sqlite3.connect('twitter_hate.db') as con:
    df = pd.read_sql_query(sql, con)
```

In [276]:
```
tweets = df['tweet_clean']

mentions = []
urls = []
hashtags = []
i = 0
for tweet in tweets:
    tweet = tweet.split()
    mentions.append(tweet.count('mentionhere')+tweet.count('mentionhere:')+
    urls.append(tweet.count('urlhere'))
    hashtags.append(tweet.count('hashtaghere'))
    tweet = [token for token in tweet if token not in [';&','']]
    tweet = [token for token in tweet if token not in ['&#;mentionhere:','m
    tweet = " ".join(tweet)
    tweets[i] = tweet
    i += 1

df['tweet_no_others'] = tweets
df['mention_count'] = mentions
df['url_count'] = urls
df['hashtag_count'] = hashtags
```

```
C:\Users\seanx\anaconda3\lib\site-packages\ipykernel_launcher.py:15: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://p
andas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-vi
ew-versus-a-copy)
  from ipykernel import kernelapp as app
```

In [277]:
```
df.head()
```

Out[277]:

	index	count	hate_speech	offensive_language	neither	class	tweet	tweet_
0	17	3	1	2	0	1	" bitch who do you love "	bitc
1	23	3	0	3	0	1	" fuck no that bitch dont even suck dick " &#1...	fuck don suc vide
2	38	3	0	2	1	1	" lames crying over hoes thats tears of a clown "	lames hoes tears
3	59	3	0	3	0	1	"..All I wanna do is get money and fuck model ...	all i v get r fuck b ru
4	62	3	0	3	0	1	"@ARIZZLEINDACUT: Females think dating a pussy...	fe think puss nov m

In [278]:
```
corpus = df['tweet_no_others']
```

Bag of Words Features

In [279]:
```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(min_df=0., max_df=1.)
cv_matrix = cv.fit_transform(corpus)
cv_matrix = cv_matrix.toarray()
cv_matrix
```

Out[279]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [280]:
```
# get all unique words in the corpus
vocab = cv.get_feature_names()
# show document feature vectors
df_BOW = pd.DataFrame(cv_matrix, columns=vocab)
df_BOW['class'] = df['class']
df_BOW
```

Out[280]:

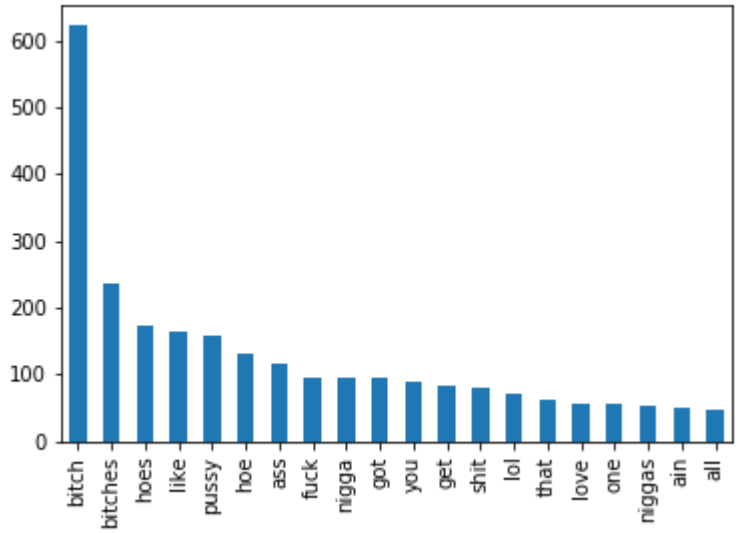
	aa	aaaaaaaaand	aap	aaron	aaronmacgruder	ab	ability	abortion	about	abraham	..
0	0	0	0	0	0	0	0	0	0	0	..
1	0	0	0	0	0	0	0	0	0	0	..
2	0	0	0	0	0	0	0	0	0	0	..
3	0	0	0	0	0	0	0	0	0	0	..
4	0	0	0	0	0	0	0	0	0	0	..
...	...	...	...	...	...	...	...	...	...	...	..
2855	0	0	0	0	0	0	0	0	0	0	..
2856	0	0	0	0	0	0	0	0	0	0	..
2857	0	0	0	0	0	0	0	0	0	0	..
2858	0	0	0	0	0	0	0	0	0	0	..
2859	0	0	0	0	0	0	0	0	0	0	..

2860 rows × 5139 columns

In [281]:
```
BOW_offensive = df_BOW[df_BOW['class'] == 1]
BOW_hate = df_BOW[df_BOW['class'] == 0]
BOW_offensive = BOW_offensive.drop(columns=['class'])
BOW_hate = BOW_hate.drop(columns=['class'])
```

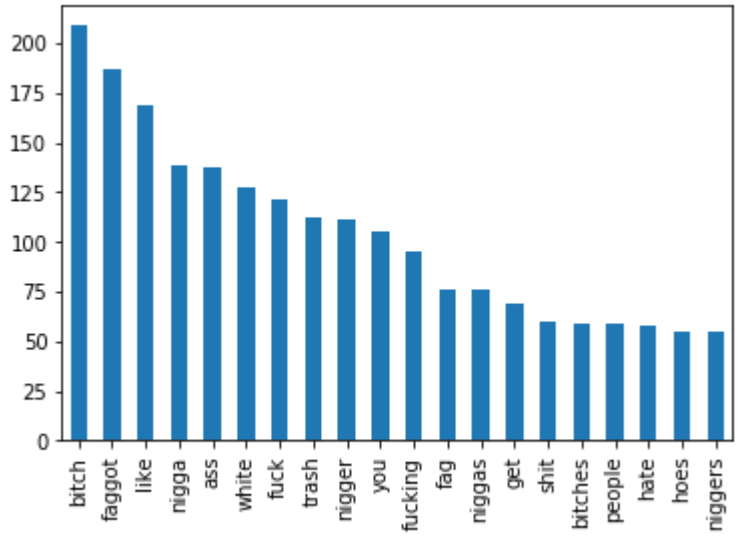
In [282]:
```
BOW_count_off=BOW_offensive.sum()
BOW_off_largest = BOW_count_off.nlargest(20)
BOW_off_largest.plot(kind='bar')
```

Out[282]: <matplotlib.axes._subplots.AxesSubplot at 0x2ed8c621d88>



In [283]:
```
BOW_count_hate=BOW_hate.sum()
BOW_hate_largest = BOW_count_hate.nlargest(20)
BOW_hate_largest.plot(kind='bar')
```

Out[283]: <matplotlib.axes._subplots.AxesSubplot at 0x2ed8c61b908>



Training BOW with Logistic Regression and Decision Tree

In [284]:
```
X = pd.concat([df_BOW.drop(columns = ['class']), df[['mention_count', 'url_
y = df['class'].astype(int)
```

In [285]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

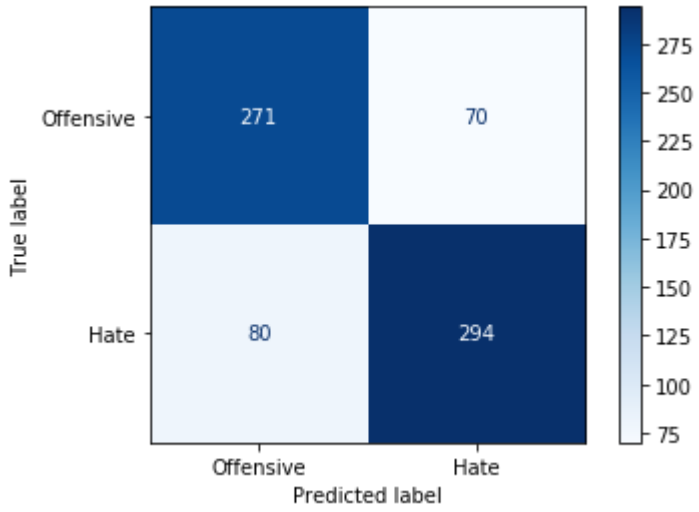
In [286]:
```
from sklearn.pipeline import Pipeline
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.metrics import classification_report
```

In [287]:
```
param_grid = [{}]
lg = GridSearchCV(LogisticRegression(),
                            param_grid,
                            cv=KFold(n_splits=5,
                                                random_state=42).split(X_trai
                            verbose=2)
y_preds_lg = lg.fit(X_train, y_train).predict(X_test)
```

```
C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection\_split.p
y:296: FutureWarning: Setting a random_state has no effect since shuffle is
False. This will raise an error in 0.24. You should leave random_state to i
ts default (None), or set shuffle=True.
  FutureWarning
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits
[CV]  ................................................................
[CV] ................................................. , total=   0.6s
[CV]  ................................................................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.5s remaining:
0.0s

[CV] ................................................. , total=   0.6s
[CV]  ................................................................
[CV] ................................................. , total=   0.6s
[CV]  ................................................................
[CV] ................................................. , total=   0.6s
[CV]  ................................................................
[CV] ................................................. , total=   0.6s

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    3.0s finished
```

In [288]:
```
from sklearn.metrics import plot_confusion_matrix
class_names = ['Offensive', 'Hate']
plot_confusion_matrix(lg, X_test, y_test, cmap=plt.cm.Blues, display_labels
```

Out[288]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2edb527
19c8>
```



In [289]:
```
from sklearn.metrics import classification_report
report_lg_BOW = classification_report( y_test, y_preds_lg)
print(report_lg_BOW)
```

```
              precision    recall  f1-score   support

           0       0.77      0.79      0.78       341
           1       0.81      0.79      0.80       374

    accuracy                           0.79       715
   macro avg       0.79      0.79      0.79       715
weighted avg       0.79      0.79      0.79       715
```

In [290]:
```
importance_logreg = lg.best_estimator_.coef_.tolist()[0]

features = list(df_BOW.columns)
feature_importance_logreg = pd.DataFrame(list(zip(features,importance_logre
feature_importance_logreg = feature_importance_logreg.sort_values(by='impor
feature_importance_logreg.head(20)
```

Out[290]:

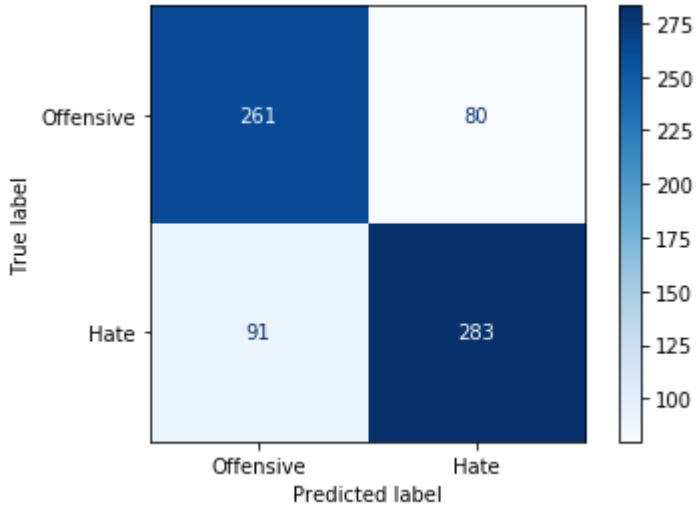
	features	importance
1455	faggit	-2.108559
3030	niggaz	-2.079960
3027	niggahs	-1.998250
1456	faggot	-1.949334
3033	niggerous	-1.639718
1732	gave	-1.594995
3023	nigerian	-1.555284
2411	kike	-1.518631
4949	whistle	-1.385182
3546	queen	-1.284641
444	black	-1.274186
1305	dwn	-1.250660
1458	fagjo	-1.218483
1453	facts	-1.196915
911	cool	-1.137583
3283	pennsylvanians	-1.103863
4160	sperm	-1.076972
851	comfortable	-1.046262
348	beaner	-1.045217
4061	smfh	-1.037146

In [291]:
```
tree = GridSearchCV(DecisionTreeClassifier(),
                            param_grid,
                            cv=KFold(n_splits=5,
                                     random_state=42).split(X_trai
                            verbose=2)

y_preds_tree = tree.fit(X_train, y_train).predict(X_test)
```

```
C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection\_split.p
y:296: FutureWarning: Setting a random_state has no effect since shuffle is
False. This will raise an error in 0.24. You should leave random_state to i
ts default (None), or set shuffle=True.
  FutureWarning
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits
[CV]  ................................................................
[CV] ................................................. , total=   0.9s
[CV]  ................................................................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.8s remaining:
0.0s

[CV] ................................................. , total=   0.9s
[CV]  ................................................................
[CV] ................................................. , total=   1.0s
[CV]  ................................................................
[CV] ................................................. , total=   0.9s
[CV]  ................................................................
[CV] ................................................. , total=   0.9s

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    4.5s finished
```

In [292]:
```
plot_confusion_matrix(tree, X_test, y_test, cmap=plt.cm.Blues, display_labe
```

Out[292]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2ed8fd6
c608>
```



In [293]:
```
report_tree = classification_report( y_test, y_preds_tree)
print(report_tree)
```

```
              precision    recall  f1-score   support

           0       0.74      0.77      0.75       341
           1       0.78      0.76      0.77       374

    accuracy                           0.76       715
   macro avg       0.76      0.76      0.76       715
weighted avg       0.76      0.76      0.76       715
```

In [294]:
```
importance_tree = tree.best_estimator_.feature_importances_.tolist()

features = list(df_BOW.columns)
feature_importance_logreg = pd.DataFrame(list(zip(features,importance_tree)
feature_importance_logreg = feature_importance_logreg.sort_values(by='impor
feature_importance_logreg.head(20)
```

Out[294]:

	features	importance
432	bitch	0.090740
434	bitches	0.071617
3535	pussies	0.069759
2046	hoeing	0.066966
2044	hockey	0.054338
3027	niggahs	0.025053
3023	nigerian	0.023572
5138	zzzzzz	0.014328
3025	niggaa	0.008869
1667	fuccing	0.008625
224	ass	0.008191
4949	whistle	0.007725
3937	shirts	0.007261
1676	fuckin	0.006865
4683	tv	0.006569
423	bird	0.006502
1942	hat	0.006423
1305	dwn	0.005905
4716	ugliest	0.005897
1157	dice	0.005507

Word2vec embedding

In [295]:
```
from gensim.models import word2vec
import nltk
```

In [296]:
```
feature_size = 100    # Word vector dimensionality
window_context = 30          # Context window size
min_word_count = 1   # Minimum word count
sample = 1e-3   # Downsample setting for frequent words

wpt = nltk.WordPunctTokenizer()
tokenized_corpus = [wpt.tokenize(document) for document in corpus]

# Set values for various parameters
feature_size = 100    # Word vector dimensionality
window_context = 30          # Context window size
min_word_count = 1   # Minimum word count
sample = 1e-3   # Downsample setting for frequent words

w2v_model = word2vec.Word2Vec(tokenized_corpus, size=feature_size,
                          window=window_context, min_count=min_word_count,
                          sample=sample, iter=50)

# view similar words based on gensim's model
similar_words = {search_term: [item[0] for item in w2v_model.wv.most_simila
                  for search_term in ['faggot', 'nigger', 'nigga','white','
similar_words
```

Out[296]:
```
{'faggot': ['tear', 'fag', 'powered', 'nicest', 'overly'],
 'nigger': ['traditions', 'honor', 'hoodrats', 'tyler', 'republicans'],
 'nigga': ['lame', 'scary', 'twin', 'yah', 'yo'],
 'white': ['trash', 'texarkana', 'trailer', 'lid', 'brainwash'],
 'queer': ['pathetic', 'sweet', 'fotos', 'obama', 'lbum'],
 'gay': ['sucks', 'welcome', 'blacklisted', 'episode', 'cuba'],
 'coon': ['tweets', 'ban', 'its', 'picks', 'work'],
 'kill': ['panthers', 'presser', 'hijack', 'route', 'wr'],
 'hate': ['goddamit', 'cripples', 'rally', 'kkk', 'dairy'],
 'trash': ['white', 'trailer', 'texarkana', 'slave', 'brainwash'],
 'black': ['faves', 'the', 'people', 'pack', 'preventing'],
 'retard': ['guinea', 'fucktards', 'republican', 'armed', 'mom'],
 'beaner': ['opinion', 'goddam', 'pussyed', 'homeless', 'quit'],
 'wetback': ['chinatown', 'slant', 'towelhead', 'bulldozed', 'kraut'],
 'homosexual': ['infiltration', 'gear', 'priesthood', 'sandwiches', 'shelb
y'],
 'gays': ['wholesome', 'perm', 'fool', 'humble', 'stars']}
```

In [297]:
```
def average_word_vectors(words, model, vocabulary, num_features):

    feature_vector = np.zeros((num_features,),dtype="float64")
    nwords = 0.

    for word in words:
        if word in vocabulary:
            nwords = nwords + 1.
            feature_vector = np.add(feature_vector, model[word])

    if nwords:
        feature_vector = np.divide(feature_vector, nwords)

    return feature_vector

def averaged_word_vectorizer(corpus, model, num_features):
    vocabulary = set(model.wv.index2word)
    features = [average_word_vectors(tokenized_sentence, model, vocabulary,
                    for tokenized_sentence in corpus]
    return np.array(features)

w2v_feature_array = averaged_word_vectorizer(corpus=tokenized_corpus, model
                                             num_features=feature_size)
pd.DataFrame(w2v_feature_array)
```

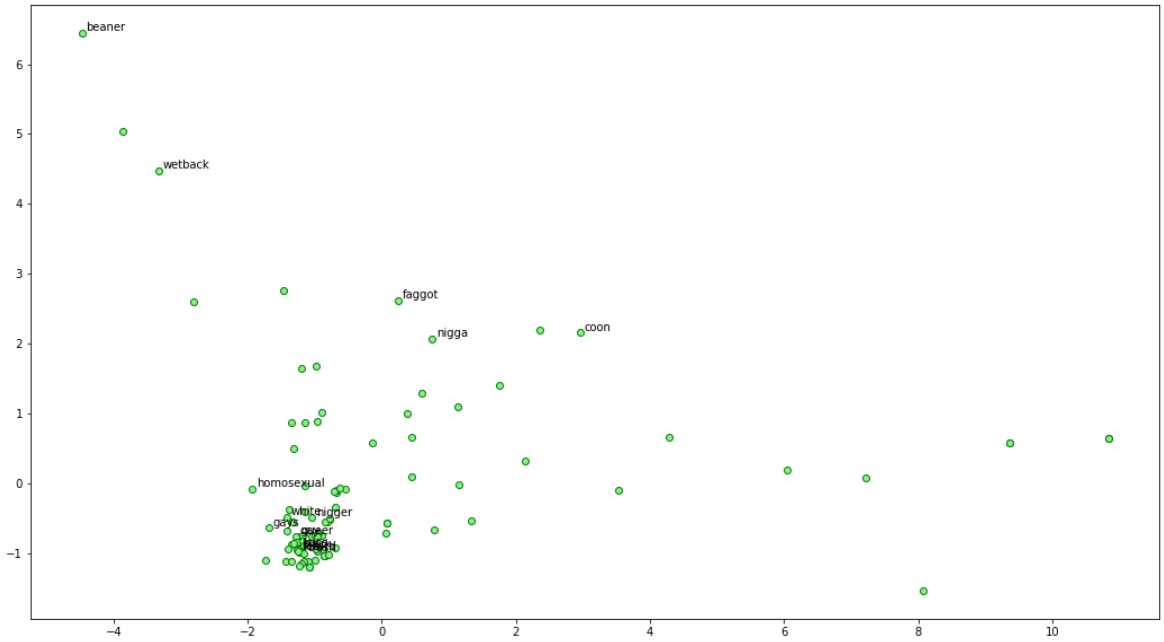
```
C:\Users\seanx\anaconda3\lib\site-packages\ipykernel_launcher.py:9: Depreca
tionWarning: Call to deprecated `__getitem__` (Method will be removed in 4.
0.0, use self.wv.__getitem__() instead).
  if __name__ == '__main__':
```

Out[297]:

	0	1	2	3	4	5	6	7	
0	0.506105	-0.229800	-0.363152	-0.535886	-0.026520	-0.957140	-0.865238	-0.185917	0.
1	0.249636	-0.258571	-0.349168	0.017806	-0.128595	-0.586897	-0.438253	0.146007	0.
2	0.135101	-0.165665	-0.148459	-0.236594	0.089040	-0.499186	-0.260654	-0.167711	0.
3	0.508545	-0.509254	-0.414899	-0.506056	0.074721	-0.791285	-0.530319	-0.158569	0.
4	0.265640	-0.226422	-0.075378	-0.251172	-0.232360	-0.473984	-0.222964	-0.225917	0.
...	...	...	...	...	...	...	...	...	
2855	0.384639	-0.419814	-0.089338	0.005814	-0.628487	-0.527565	-0.220804	0.108152	-0.
2856	0.483009	-0.686711	0.018624	-0.483838	-0.405409	-0.536809	-0.245619	-0.310263	-0.
2857	0.253021	-0.326742	-0.202440	0.106835	-0.229036	-0.103237	-0.055520	0.019371	-0
2858	0.341643	0.076964	-0.259367	-0.311987	-0.277611	-0.759661	-0.291327	0.325615	0.
2859	0.232542	-0.141576	-0.107217	-0.379523	0.102264	-0.501098	-0.331836	-0.146911	0.

2860 rows × 100 columns

In [244]:
```
from sklearn.decomposition import PCA

words = sum([[k] + v for k, v in similar_words.items()], [])
wvs = w2v_model.wv[words]

pca = PCA(n_components=2)
np.set_printoptions(suppress=True)
P = pca.fit_transform(wvs)
labels = ['faggot', 'nigger', 'nigga','white','queer','gay','coon','kill',

plt.figure(figsize=(18, 10))
plt.scatter(P[:, 0], P[:, 1], c='lightgreen', edgecolors='g')
for label, x, y in zip(labels, P[:, 0], P[:, 1]):
    plt.annotate(label, xy=(x+0.06, y+0.03), xytext=(0, 0), textcoords='off
```



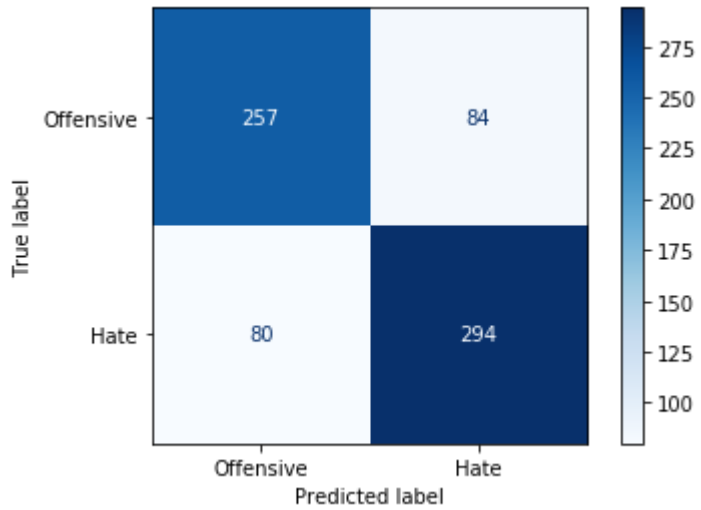
In [110]:
```
X_w2v = pd.DataFrame(w2v_feature_array)
y = df['class'].astype(int)
X_train_w2v, X_test_w2v, y_train_w2v, y_test_w2v = train_test_split(X_w2v,
```

In [111]:
```
lg_w2v = GridSearchCV(LogisticRegression(max_iter = 1000),
                      param_grid,
                      cv=KFold(n_splits=5,
                               random_state=42).split(X_trai
                      verbose=2)
y_preds_w2v_lg = lg_w2v.fit(X_train_w2v, y_train_w2v).predict(X_test_w2v)
```

```
C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection\_split.p
y:296: FutureWarning: Setting a random_state has no effect since shuffle is
False. This will raise an error in 0.24. You should leave random_state to i
ts default (None), or set shuffle=True.
  FutureWarning
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:
0.0s

Fitting 5 folds for each of 1 candidates, totalling 5 fits
[CV]  ................................................................
[CV] ................................................. , total=   0.1s
[CV]  ................................................................
[CV] ................................................. , total=   0.1s
[CV]  ................................................................
[CV] ................................................. , total=   0.1s
[CV]  ................................................................
[CV] ................................................. , total=   0.1s
[CV]  ................................................................
[CV] ................................................. , total=   0.1s

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.2s finished
```

In [112]:
```
plot_confusion_matrix(lg_w2v, X_test_w2v, y_test_w2v, cmap=plt.cm.Blues, di
```

Out[112]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2ed9859
9f88>
```



In [113]:
```
report_w2v = classification_report( y_test_w2v, y_preds_w2v_lg)
print(report_w2v)
```

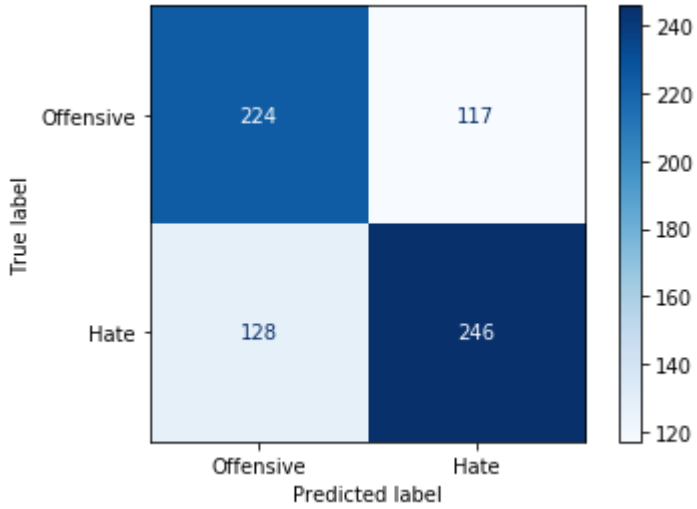
```
              precision    recall  f1-score   support

           0       0.76      0.75      0.76       341
           1       0.78      0.79      0.78       374

    accuracy                           0.77       715
   macro avg       0.77      0.77      0.77       715
weighted avg       0.77      0.77      0.77       715
```

In [114]:
```
tree_w2v = GridSearchCV(DecisionTreeClassifier(),
                        param_grid,
                        cv=KFold(n_splits=5,
                                 random_state=42).split(X_trai
                        verbose=2)
y_preds_w2v_tree = tree_w2v.fit(X_train_w2v, y_train_w2v).predict(X_test_w2
```

```
C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection\_split.p
y:296: FutureWarning: Setting a random_state has no effect since shuffle is
False. This will raise an error in 0.24. You should leave random_state to i
ts default (None), or set shuffle=True.
  FutureWarning
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits
[CV]  ................................................................
[CV] ................................................. , total=   0.2s
[CV]  ................................................................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.1s remaining:
0.0s

[CV] ................................................. , total=   0.3s
[CV]  ................................................................
[CV] ................................................. , total=   0.2s
[CV]  ................................................................
[CV] ................................................. , total=   0.2s
[CV]  ................................................................
[CV] ................................................. , total=   0.2s

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    1.0s finished
```

In [115]:
```
plot_confusion_matrix(tree_w2v, X_test_w2v, y_test_w2v, cmap=plt.cm.Blues,
```

Out[115]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2ed8df3
c9c8>
```



In [116]:
```
report_w2v_tree = classification_report( y_test_w2v, y_preds_w2v_tree)
print(report_w2v_tree)
```

```
              precision    recall  f1-score   support

           0       0.64      0.66      0.65       341
           1       0.68      0.66      0.67       374

    accuracy                           0.66       715
   macro avg       0.66      0.66      0.66       715
weighted avg       0.66      0.66      0.66       715
```

Combining BOW and W2V

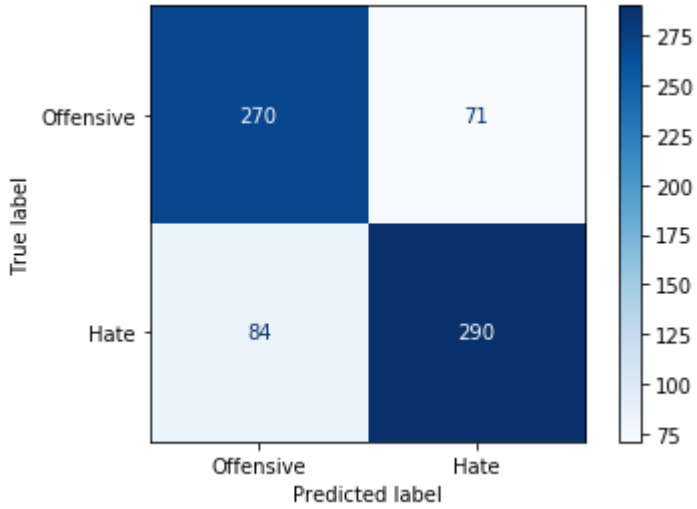
In [117]:
```
X_mixed = pd.concat([pd.DataFrame(w2v_feature_array), df_BOW.drop(columns=[
y = df['class'].astype(int)
X_train_mixed, X_test_mixed, y_train_mixed, y_test_mixed = train_test_split
```

In [118]:
```
lg_mixed = GridSearchCV(LogisticRegression(max_iter = 1000),
                        param_grid,
                        cv=KFold(n_splits=5,
                                 random_state=42).split(X_trai
                        verbose=2)
y_preds_mixed = lg_mixed.fit(X_train_mixed, y_train_mixed).predict(X_test_m
```

```
C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection\_split.p
y:296: FutureWarning: Setting a random_state has no effect since shuffle is
False. This will raise an error in 0.24. You should leave random_state to i
ts default (None), or set shuffle=True.
  FutureWarning
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits
[CV]  ................................................................
[CV] ................................................. , total=   1.4s
[CV]  ................................................................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.3s remaining:
0.0s

[CV] ................................................. , total=   1.6s
[CV]  ................................................................
[CV] ................................................. , total=   1.7s
[CV]  ................................................................
[CV] ................................................. , total=   1.7s
[CV]  ................................................................
[CV] ................................................. , total=   1.4s

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    7.7s finished
```

In [119]:
```
plot_confusion_matrix(lg_mixed, X_test_mixed, y_test_mixed, cmap=plt.cm.Blu
```

Out[119]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2ed8df3
c148>
```



In [120]:
```
report_lg_mixed = classification_report( y_test_mixed, y_preds_mixed)
print(report_lg_mixed)
```

```
              precision    recall  f1-score   support

           0       0.76      0.79      0.78       341
           1       0.80      0.78      0.79       374

    accuracy                           0.78       715
   macro avg       0.78      0.78      0.78       715
weighted avg       0.78      0.78      0.78       715
```

In [121]:
```
features = list(X_mixed.columns)
feature_importance_logreg = pd.DataFrame(list(zip(features,importance_logre
feature_importance_logreg = feature_importance_logreg.sort_values(by='impor
feature_importance_logreg.head(20)
```

Out[121]:

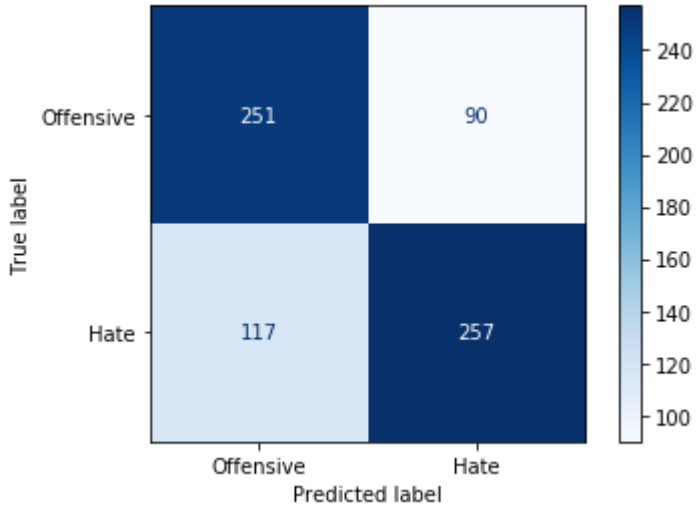
	features	importance
1455	emoji	-2.108559
3030	muslims	-2.079960
3027	murdered	-1.998250
1456	emojis	-1.949334
3033	muthafucka	-1.639718
1732	found	-1.594995
3023	muhhfuckin	-1.555284
2411	jezzy	-1.518631
4949	wannabe	-1.385182
3546	pray	-1.284641
444	bc	-1.274186
1305	dm	-1.250660
1458	encrusted	-1.218483
1453	emm	-1.196915
911	closer	-1.137583
3283	otter	-1.103863
4160	smfh	-1.076972
851	chill	-1.046262
348	aunt	-1.045217
4061	shout	-1.037146

In [122]:
```
tree_mixed = GridSearchCV(DecisionTreeClassifier(),
                          param_grid,
                          cv=KFold(n_splits=5,
                                   random_state=42).split(X_trai
                          verbose=2)
y_preds_mixed_tree = tree_mixed.fit(X_train_mixed, y_train_mixed).predict(X
```

```
C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection\_split.p
y:296: FutureWarning: Setting a random_state has no effect since shuffle is
False. This will raise an error in 0.24. You should leave random_state to i
ts default (None), or set shuffle=True.
  FutureWarning
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.

Fitting 5 folds for each of 1 candidates, totalling 5 fits
[CV]  ................................................................
[CV] ................................................. , total=   0.9s
[CV]  ................................................................

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.8s remaining:
0.0s

[CV] ................................................. , total=   0.7s
[CV]  ................................................................
[CV] ................................................. , total=   0.8s
[CV]  ................................................................
[CV] ................................................. , total=   0.9s
[CV]  ................................................................
[CV] ................................................. , total=   1.0s

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    4.2s finished
```

In [123]:
```
plot_confusion_matrix(tree_mixed, X_test_mixed, y_test_mixed, cmap=plt.cm.B
```

Out[123]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2ed9890
7d88>
```



In [124]:
```
report_tree_mixed = classification_report( y_test_mixed, y_preds_mixed_tree
print(report_tree_mixed)
```

```
              precision    recall  f1-score   support

           0       0.68      0.74      0.71       341
           1       0.74      0.69      0.71       374

    accuracy                           0.71       715
   macro avg       0.71      0.71      0.71       715
weighted avg       0.71      0.71      0.71       715
```

Using LIME to interpret predictions

In [299]:

```
import lime
import lime.lime_tabular

i = np.random.randint(0, X_test.shape[0])

explainer = lime.lime_tabular.LimeTabularExplainer(training_data = X_train.
                                                   mode = 'classification',
                                                   feature_names = features
                                                  class_names = ['Hate', 'O

exp = explainer.explain_instance(data_row = X_test.iloc[i].to_numpy(),
                                 predict_fn = lg.predict_proba)
actual = df_BOW['class'][i]

if actual == 0:
    actual = 'Hate'
else:
    actual = 'Offensive'

print(f'Actual classification: {actual}')
exp.show_in_notebook()
```

```
---------------------------------------------------------------------------
-----
IndexError                                Traceback (most recent call
 last)
<ipython-input-299-fa11010aa5e8> in <module>
      7                                                    mode = 'cla
ssification',
      8                                                    feature_nam
es = features,
----> 9                                                   class_names
 = ['Hate', 'Offensive'])
     10
     11 exp = explainer.explain_instance(data_row = X_test.iloc[i].to_
numpy(),

~\anaconda3\lib\site-packages\lime\lime_tabular.py in __init__(self, t
raining_data, mode, training_labels, feature_names, categorical_featur
es, categorical_names, kernel_width, kernel, verbose, class_names, fea
ture_selection, discretize_continuous, discretizer, sample_around_inst
ance, random_state, training_data_stats)
    216                             training_data, self.categorical_featur
es,
    217                             self.feature_names, labels=training_la
bels,
--> 218                             random_state=self.random_state)
    219             elif discretizer == 'decile':
    220                 self.discretizer = DecileDiscretizer(

~\anaconda3\lib\site-packages\lime\discretize.py in __init__(self, dat
a, categorical_features, feature_names, labels, random_state)
    178         BaseDiscretizer.__init__(self, data, categorical_featu
res,
    179                                  feature_names, labels=labels,
```

```
--> 180                                    random_state=random_state)
    181
    182     def bins(self, data, labels):

~\anaconda3\lib\site-packages\lime\discretize.py in __init__(self, dat
a, categorical_features, feature_names, labels, random_state, data_sta
ts)
     62             n_bins = qts.shape[0]  # Actually number of border
s (= #bins-1)
     63             boundaries = np.min(data[:, feature]), np.max(data
[:, feature])
---> 64             name = feature_names[feature]
     65
     66             self.names[feature] = ['%s <= %.2f' % (name, qts[0
])]

IndexError: list index out of range
```

Training CBOW based on twitter dataset

In [10]:
```
from keras.preprocessing import text
from keras.utils import np_utils
from keras.preprocessing import sequence

tokenizer = text.Tokenizer()
tokenizer.fit_on_texts(corpus)
word2id = tokenizer.word_index

word2id['PAD'] = 0
id2word = {v:k for k, v in word2id.items()}
wids = [[word2id[w] for w in text.text_to_word_sequence(doc)] for doc in co

vocab_size = len(word2id)
embed_size = 100
window_size = 2

print('Vocabulary Size:', vocab_size)
print('Vocabulary Sample:', list(word2id.items())[:10])
```

```
Using TensorFlow backend.
C:\Users\seanx\anaconda3\lib\site-packages\tensorflow\python\framework
\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a synon
ym of type is deprecated; in a future version of numpy, it will be und
erstood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\seanx\anaconda3\lib\site-packages\tensorflow\python\framework
\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a synon
ym of type is deprecated; in a future version of numpy, it will be und
erstood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\seanx\anaconda3\lib\site-packages\tensorflow\python\framework
\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a synon
ym of type is deprecated; in a future version of numpy, it will be und
erstood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\seanx\anaconda3\lib\site-packages\tensorflow\python\framework
\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a synon
ym of type is deprecated; in a future version of numpy, it will be und
erstood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\seanx\anaconda3\lib\site-packages\tensorflow\python\framework
\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a synon
ym of type is deprecated; in a future version of numpy, it will be und
erstood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\seanx\anaconda3\lib\site-packages\tensorflow\python\framework
\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a synon
ym of type is deprecated; in a future version of numpy, it will be und
erstood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
C:\Users\seanx\anaconda3\lib\site-packages\tensorboard\compat\tensorfl
ow_stub\dtypes.py:541: FutureWarning: Passing (type, 1) or '1type' as
a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
```

```
C:\Users\seanx\anaconda3\lib\site-packages\tensorboard\compat\tensorfl
ow_stub\dtypes.py:542: FutureWarning: Passing (type, 1) or '1type' as
a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\seanx\anaconda3\lib\site-packages\tensorboard\compat\tensorfl
ow_stub\dtypes.py:543: FutureWarning: Passing (type, 1) or '1type' as
a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\seanx\anaconda3\lib\site-packages\tensorboard\compat\tensorfl
ow_stub\dtypes.py:544: FutureWarning: Passing (type, 1) or '1type' as
a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\seanx\anaconda3\lib\site-packages\tensorboard\compat\tensorfl
ow_stub\dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as
a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\seanx\anaconda3\lib\site-packages\tensorboard\compat\tensorfl
ow_stub\dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as
a synonym of type is deprecated; in a future version of numpy, it will
be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])

Vocabulary Size: 5168
Vocabulary Sample: [('i', 1), ('bitch', 2), ('like', 3), ('bitches', 4),
('ass', 5), ('s', 6), ('nigga', 7), ('hoes', 8), ('fuck', 9), ('pussy', 1
0)]
```

In [11]:

```
def generate_context_word_pairs(corpus, window_size, vocab_size):
    context_length = window_size*2
    for words in corpus:
        sentence_length = len(words)
        for index, word in enumerate(words):
            context_words = []
            label_word   = []
            start = index - window_size
            end = index + window_size + 1

            context_words.append([words[i]
                                  for i in range(start, end)
                                  if 0 <= i < sentence_length
                                  and i != index])
            label_word.append(word)

            x = sequence.pad_sequences(context_words, maxlen=context_length
            y = np_utils.to_categorical(label_word, vocab_size)
            yield (x, y)
```

In [12]:
```
i = 0
for x, y in generate_context_word_pairs(corpus=wids, window_size=window_siz
    if 0 not in x[0]:
        print('Context (X):', [id2word[w] for w in x[0]], '-> Target (Y):',

        if i == 10:
            break
        i += 1
```

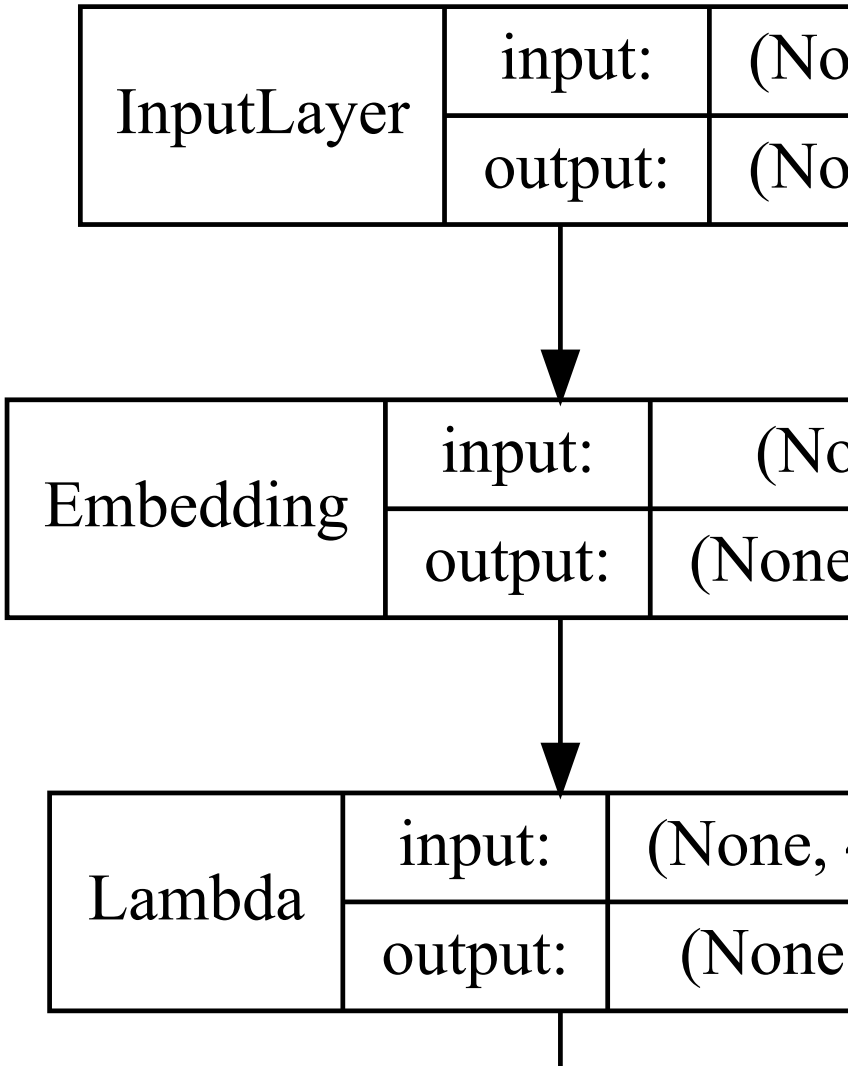
```
Context (X): ['fuck', 'bitch', 'even', 'suck'] -> Target (Y): dont
Context (X): ['bitch', 'dont', 'suck', 'dick'] -> Target (Y): even
Context (X): ['dont', 'even', 'dick', 'kermit'] -> Target (Y): suck
Context (X): ['even', 'suck', 'kermit', 'videos'] -> Target (Y): dick
Context (X): ['suck', 'dick', 'videos', 'bout'] -> Target (Y): kermit
Context (X): ['dick', 'kermit', 'bout', 'fuck'] -> Target (Y): videos
Context (X): ['kermit', 'videos', 'fuck', 'ig'] -> Target (Y): bout
Context (X): ['lames', 'crying', 'thats', 'tears'] -> Target (Y): hoes
Context (X): ['crying', 'hoes', 'tears', 'clown'] -> Target (Y): thats
Context (X): ['all', 'i', 'get', 'money'] -> Target (Y): wanna
Context (X): ['i', 'wanna', 'money', 'fuck'] -> Target (Y): get
```

In [13]:
```
import keras.backend as K
from keras.models import Sequential
from keras.layers import Dense, Embedding, Lambda

cbow = Sequential()
cbow.add(Embedding(input_dim=vocab_size, output_dim=embed_size, input_lengt
cbow.add(Lambda(lambda x: K.mean(x, axis=1), output_shape=(embed_size,)))
cbow.add(Dense(vocab_size, activation='softmax'))

cbow.compile(loss='categorical_crossentropy', optimizer='rmsprop')
print(cbow.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 4, 100)            516800
_________________________________________________________________
lambda_1 (Lambda)            (None, 100)               0
_________________________________________________________________
dense_1 (Dense)              (None, 5168)              521968
=================================================================
Total params: 1,038,768
Trainable params: 1,038,768
Non-trainable params: 0
_________________________________________________________________
None
```

In [14]:
```
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot

SVG(model_to_dot(cbow, show_shapes=True, show_layer_names=False,
                 rankdir='TB').create(prog='dot', format='svg'))
```

Out[14]:



In [103]:

```
for epoch in range(1, 6):
    loss = 0.
    i = 0
    for x, y in generate_context_word_pairs(corpus=wids, window_size=window
        i += 1
        loss += cbow.train_on_batch(x, y)
        if i % 100000 == 0:
            print('Processed {} (context, word) pairs'.format(i))

    print('Epoch:', epoch, '\tLoss:', loss)
    print()
```

```
Epoch: 1        Loss: 186800.52378857136

Epoch: 2        Loss: 224056.9714373313

Epoch: 3        Loss: 229746.46687418967

Epoch: 4        Loss: 231765.57582517853

Epoch: 5        Loss: 232720.49580033985
```

In [157]:
```
weights = cbow.get_weights()[0]
weights = weights[:5167]
print(weights.shape)
trained_embedding = pd.DataFrame(weights, index=list(id2word.values())[:516
trained_embedding
#trained_embedding.to_csv('trained_embedding.csv')
```

```
(5167, 100)
```

Out[157]:

	0	1	2	3	4	5	6	7
i	-0.222502	0.259196	-0.149339	0.365478	0.104890	-0.211132	0.214763	-0.217084
bitch	-0.374511	1.947970	-0.348361	-0.082477	0.320433	0.033585	0.143289	-0.180969
like	0.420282	0.267553	-0.226183	0.264617	0.179408	0.165740	0.691293	0.112533
bitches	0.168009	0.395935	-0.082524	0.197108	-0.037274	-0.238570	-0.355305	0.254387
ass	-0.203083	-0.177824	0.219084	0.174521	0.464803	-0.103060	-0.072733	0.006912
...	...	...	...	...	...	...	...	...
sugar	-0.030888	-0.021650	0.061245	0.057789	0.028779	-0.085339	0.035457	-0.080947
rush	-0.020139	-0.069486	0.022567	-0.007940	0.063400	0.011016	-0.058338	-0.061581
buck	0.012096	-0.053386	0.059186	0.022004	0.019282	-0.010376	0.041935	-0.071567
youu	-0.025845	-0.072485	0.038698	-0.000171	0.039669	-0.034721	0.006980	-0.041145
tellin	-0.059408	-0.053047	0.002504	-0.032275	0.022687	-0.011306	-0.006478	-0.020560

5167 rows × 100 columns

In [132]:
```
trained_embedding = pd.read_csv('trained_embedding.csv')
trained_embedding.set_index(trained_embedding['Unnamed: 0'], inplace = True
trained_embedding = trained_embedding.drop(columns = 'Unnamed: 0')
trained_embedding.head()
```

Out[132]:

Unnamed: 0	0	1	2	3	4	5	6	
i	-0.222502	0.259196	-0.149339	0.365478	0.104890	-0.211132	0.214763	-0.2170
bitch	-0.374511	1.947970	-0.348361	-0.082477	0.320433	0.033585	0.143289	-0.1809
like	0.420282	0.267553	-0.226183	0.264617	0.179408	0.165740	0.691293	0.1125
bitches	0.168009	0.395935	-0.082524	0.197108	-0.037274	-0.238570	-0.355305	0.2543
ass	-0.203083	-0.177824	0.219084	0.174521	0.464803	-0.103060	-0.072733	0.0069

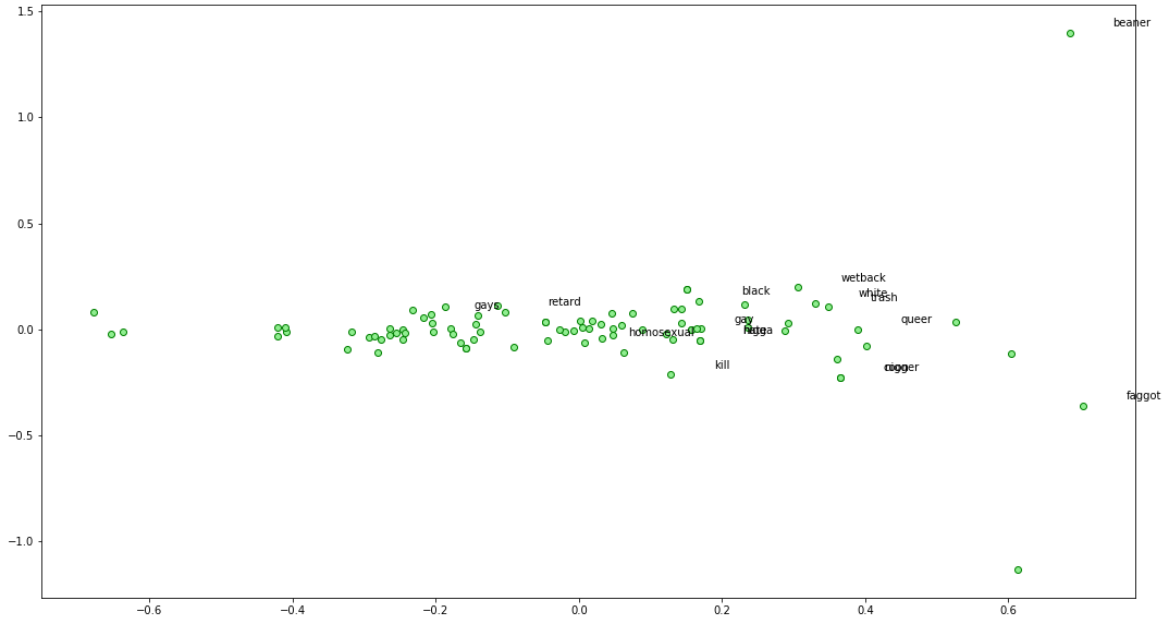
5 rows × 100 columns

In [246]:
```
from sklearn.metrics.pairwise import euclidean_distances

# compute pairwise distance matrix
distance_matrix = euclidean_distances(trained_embedding)
print(distance_matrix.shape)

# view contextually similar words
similar_words = {search_term: [id2word[idx] for idx in distance_matrix[word
                  for search_term in ['faggot', 'nigger', 'nigga','white',

similar_words
```

```
(5167, 5167)
```

Out[246]:
```
{'faggot': ['nigger', 'monkey', 'they', 'redneck', 'queer'],
 'nigger': ['niggers', 'monkey', 'ur', 'please', 'making'],
 'nigga': ['niggas', 'boss', 'might', 'broke', 'up'],
 'white': ['park', 'school', 'every', 'black', 'president'],
 'queer': ['his', 'lookin', 'check', 'fan', 'henny'],
 'gay': ['joe', 'fo', 'muzzie', 'looking', 'ol'],
 'coon': ['check', 'forget', 'took', 'for', 'water'],
 'kill': ['thug', 'street', 'cracker', 'internet', 'dirty'],
 'hate': ['live', 'd', 'swear', 'single', 'chinks'],
 'trash': ['people', 'man', 'faggots', 'jews', 'racist'],
 'black': ['probably', 'use', 'knows', 'filthy', 'followers'],
 'retard': ['wetbacks', 'throat', 'everyone', 'gook', 'muzzie'],
 'beaner': ['teabaggers', 'killing', 'niglet', 'inch', 'election'],
 'wetback': ['clearly', 'yelling', 'anti', 'dc', 'pops'],
 'homosexual': ['lgbtq', 'media', 'mgr', 'uwi', 'sideways'],
 'gays': ['jason', 'keri', 'texts', 'ona', 'arrested']}
```

In [135]:
```
#unsupervised from trained word embedding CBOW

from sklearn.decomposition import PCA
words = sum([[k] + v for k, v in similar_words.items()], [])
wvs = trained_embedding.loc[words]

pca = PCA(n_components=5)
np.set_printoptions(suppress=True)
P = pca.fit_transform(wvs)
labels =  ['faggot', 'nigger', 'nigga','white','queer','gay','coon','kill',

plt.figure(figsize=(18, 10))
plt.scatter(P[:, 0], P[:, 1], c='lightgreen', edgecolors='g')
for label, x, y in zip(labels, P[:, 0], P[:, 1]):
    plt.annotate(label, xy=(x+0.06, y+0.03), xytext=(0, 0), textcoords='off
```



In [227]:
```
tokenized_corpus = [wpt.tokenize(document) for document in corpus]

def average_word_vectors(words, model, vocabulary, num_features):

    feature_vector = np.zeros((num_features,),dtype="float64")
    nwords = 0.

    for word in words:
        if word in vocabulary:
            nwords = nwords + 1.
            feature_vector = np.add(feature_vector, model.loc[word])

    if nwords:
        feature_vector = np.divide(feature_vector, nwords)

    return feature_vector

features = [average_word_vectors(tokenized_sentence, trained_embedding, tra
                  for tokenized_sentence in tokenized_corpus]

twitter_feature_array = pd.DataFrame(np.array(features))
twitter_feature_array.to_csv('twitter_feature_array.csv')
```

In [242]:
```
twitter_feature_array
```

Out[242]:

	0	1	2	3	4	5	6	7	
0	-0.092240	0.655190	-0.054113	-0.001526	0.245873	-0.047122	-0.195120	-0.241070	-0.
1	-0.036190	0.052367	0.056565	0.040076	0.153519	-0.078184	-0.050946	-0.083378	-0.
2	-0.084097	0.030291	0.036536	0.039797	0.106109	0.002597	0.060696	-0.047483	-0.
3	0.007605	0.010492	0.032044	0.126457	0.047635	-0.069211	-0.054946	-0.011408	-0.
4	-0.100222	-0.060809	0.076636	-0.023275	0.134729	-0.025185	0.020398	-0.055950	-0.
...	...	...	...	...	...	...	...	...	
2855	0.082106	0.163869	0.087209	0.145215	0.068083	-0.082497	0.035079	0.005989	-0.
2856	-0.178984	0.105335	0.093642	0.143920	0.036653	-0.008155	-0.112179	-0.071968	-0.
2857	-0.014508	0.042220	0.106782	0.091900	0.062218	-0.030795	-0.054249	-0.053927	-0.
2858	0.001350	0.051687	0.038868	0.093754	0.076561	-0.059643	0.140547	-0.080700	-0.
2859	0.031242	0.092824	0.032311	0.049737	0.053893	-0.075962	-0.002365	0.022531	-0.

2860 rows × 100 columns

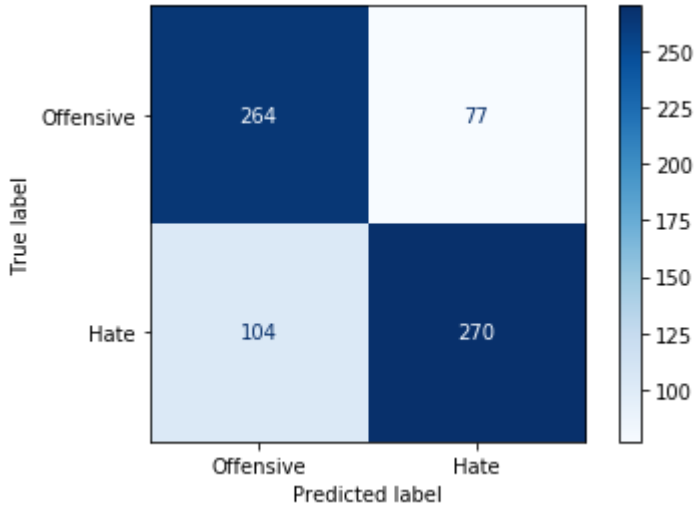
In [229]:
```
X_twit = pd.DataFrame(twitter_feature_array)
y = df['class'].astype(int)
X_train_twit, X_test_twit, y_train_twit, y_test_twit = train_test_split(X_t
```

In [230]:
```
lg_twit = GridSearchCV(LogisticRegression(max_iter = 1000),
                           param_grid,
                           cv=KFold(n_splits=5,
                                    random_state=42).split(X_trai
                           verbose=2)
y_preds_twit_lg = lg_twit.fit(X_train_twit, y_train_twit).predict(X_test_tw
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
[CV]  ................................................................
[CV] ................................................. , total=   0.0s
[CV]  ................................................................
[CV] ................................................. , total=   0.0s
[CV]  ................................................................
[CV] ................................................. , total=   0.0s
[CV]  ................................................................
[CV] ................................................. , total=   0.0s
[CV]  ................................................................
[CV] ................................................. , total=   0.0s

C:\Users\seanx\anaconda3\lib\site-packages\sklearn\model_selection\_split.p
y:296: FutureWarning: Setting a random_state has no effect since shuffle is
False. This will raise an error in 0.24. You should leave random_state to i
ts default (None), or set shuffle=True.
  FutureWarning
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:
0.0s
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.0s finished
```

In [231]:
```
plot_confusion_matrix(lg_twit, X_test_twit, y_test_twit, cmap=plt.cm.Blues,
```

Out[231]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2edb34a
c088>
```



In [300]:
```
report_lg_twit = classification_report( y_test_twit, y_preds_twit_lg)
print(report_lg_twit)
```

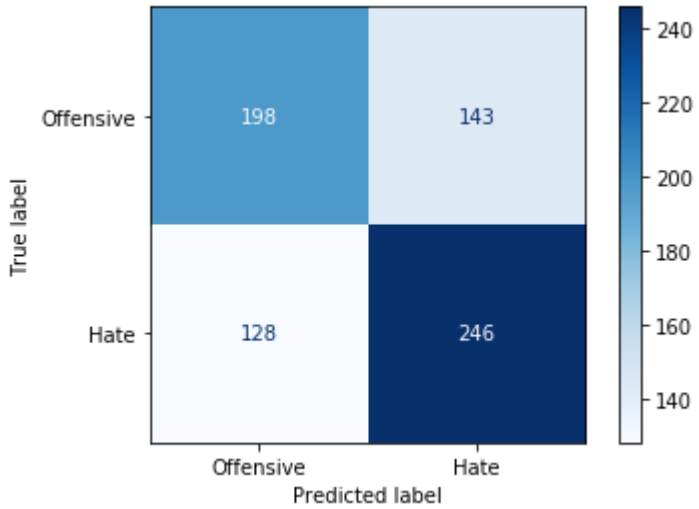
```
              precision    recall  f1-score   support

           0       0.72      0.77      0.74       341
           1       0.78      0.72      0.75       374

    accuracy                           0.75       715
   macro avg       0.75      0.75      0.75       715
weighted avg       0.75      0.75      0.75       715
```

In [236]:
```
tree_twit = GridSearchCV(DecisionTreeClassifier(),
                         param_grid,
                         cv=KFold(n_splits=5,
                                  random_state=42).split(X_trai
                         verbose=2)
y_preds_twit_tree = tree_twit.fit(X_train_twit, y_train_twit).predict(X_tes
```

```
Fitting 5 folds for each of 1 candidates, totalling 5 fits
[CV]  ................................................................
[CV] ................................................. , total=   0.1s
[CV]  ................................................................

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent wor
kers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:
0.0s

[CV] ................................................. , total=   0.1s
[CV]  ................................................................
[CV] ................................................. , total=   0.1s
[CV]  ................................................................
[CV] ................................................. , total=   0.1s
[CV]  ................................................................
[CV] ................................................. , total=   0.1s

[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.5s finished
```

In [237]:
```
plot_confusion_matrix(tree_twit, X_test_twit, y_test_twit, cmap=plt.cm.Blue
```

Out[237]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x2eda403
ed08>
```



In [239]:
```
report_tree_twit = classification_report( y_test_twit, y_preds_twit_tree)
print(report_tree_twit)
```

```
              precision    recall  f1-score   support

           0       0.61      0.58      0.59       341
           1       0.63      0.66      0.64       374

    accuracy                           0.62       715
   macro avg       0.62      0.62      0.62       715
weighted avg       0.62      0.62      0.62       715
```