Sentimen analisis dengan Algoritma Naïve Bayes

Hendri Kurniawan Prakosa[#]

[#]*S2 Ilmu Komputer, Universitas Gadjah Mada*
Yogyakarta
[1]hendri.kurniawan.p@gmail.com

Algoritma naïve bayes adalah salah satu algoritma *machine learning* yang digunakan untuk memprediksi suatu kejadian di masa yang akan datang. Naïve bayes termasuk pada kategori *supervised learning* yang membagi proses pengolahan data menjadi *data training* dan *data testing* pada data yang sudah berlabel.

I. LANDASAN TEORI

Analisis sentimen atau opinion mining yakni proses memahami, mengekstrak dan mengolah data tekstual untuk memperoleh informasi yang terdapat dalam suatu kalimat pendapat atau opini. Analisis sentimen dilakukan untuk melihat pendapat atau kecendrungan opini terhadap suatu masalah atau objek oleh seseorang, apakah cenderung punya pemikiran atau beropini positif atau negative.

Naive Bayes (Naive Bayes Classifier) adalah algoritma yang sangat efektif dalam permasalahan klasifikasi atau penggolongan. Algoritma ini bekerja berdasarkan probabilitas yang sudah ada untuk menentukan probabilitas yang akan datang. Metode Bayesian classification digunakan untuk menganalisis dalam membantu tercapainya pengambilan keputusan terbaik pada suatu permasalahan dari sejumlah alternatif. Kaitan antara Naïve Bayes dengan klasifikasi, korelasi hipotesis, dan bukti dengan klasifikasi adalah hipotesis dalam teorema Bayes merupakan label kelas yang menjadi target pemetaan dalam klasifikasi, sedangkan bukti merupakan fitur fitur yang menjadi masukan dalam model klasifikasi.

II. IMPLEMENTASI

Implementasi kali ini menggunakan data *social media* twitter tentang ojek online yang sudah dilabeli dengan angka 1 dan 0 secara manual. 1 untuk sentimen positif dan 0 untuk sentimen negatif. Data dapat dilihat pada gambar 1.

Untuk melakukan sentiment analisis beberapa tahapan yang perlu dilakukan adalah *preprocessing data*, ekstraksi fitur, training and testing data menggunakan algoritma naïve bayes dan evaluasi.

A. Pre processing Data

Preprocessing data adalah langkah awal yang dilakukan pada datamining untuk mengubah format data mentah menjadi data terformat yang siap diolah.

Untuk melakukan preprocessing data, data dibagi menjadi 2 bagian yaitu data training dan data testing sebanyak 80:20. Data training digunakan pada model klasifikasi untuk mengekstrak informasi yang ada pada data tersebut yaitu berupa relasi antara tweet dan sentimennya. Data testing digunakan untuk mengukur model klasifikasi dengan data training yang ada apakah sudah tepat atau belum.

Data yang akan diolah (sebelum dilakukan *preprocessing*) ditunjukkan pada gambar 1.

	tweet	sentimen
0	Saya juga mau vouchee @gojekindonesia https:/...	1
1	download gojek duluuu uwuwu	1
2	Aminnn...#orderan goride mhn di lancar kan.all...	1
3	Tq @gojekindonesia @golifeindonesia ð ¤. Haru...	1
4	Semoga Twitter panjang umur. Berkomunikasi den...	1
5	Semoga di tahun yang baru ini, kita senantiasa...	1
6	Sejauh ini menurut saya UI paling nyaman dari ...	1
7	Thank you @gojekindonesia pic.twitter.com/pbZ...	1
8	Hai, ada yang bisa kami bantu mengenai layanan...	1
9	Full Week Feeling Great With You Guys @gojekin...	1
10	Happy Sunday everyone â ¤â ¤\nMinggu santai me...	1
11	You can donate via @gojekindonesia application...	1
12	Jogja terasa sangat indah pagi ini dengan MOTO...	1

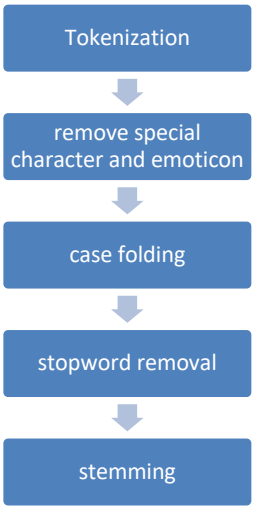
Gambar 1 data sentiment twitter

Langkah-langkah *preprocessing*-nya adalah sebagai berikut:

1. *Tokenization* atau tokenisasi, proses penghilangan tanda baca pada kalimat yang ada dalam dokumen sehingga menghasilkan kata - kata yang berdiri sendiri - sendiri.
2. *Cleansing*, menghapus karakter special dan khusus seperti ?, $, &, *, %, @, (,), dan ~ serta *emoticon*.
3. *Case Folding*, merubah semua kata menjadi *lowercase* (huruf kecil) atau *uppercase* (huruf besar)
4. *Stopword removal*
   Menghapus *stopword* seperti di, ke, dari, dan, atau, berikan, kalau, akan, dan lain – lain.
5. *Stemming*
   Proses penghilangan imbuhan, akhiran dan sisipan.

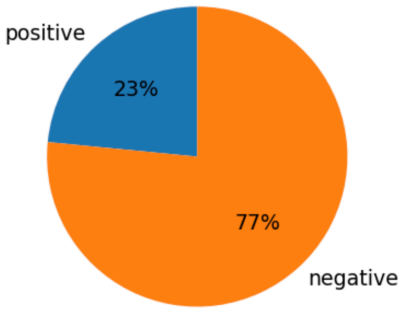
Stemming merupakan proses untuk mengembalikan kata kerja yang berimbuhan menjadi kata dasarnya.

Proses *stemming* dan *stopword removal* menggunakan algoritma sastrawi, yaitu algoritma yang menggunakan *library* berbahasa Indonesia. Algoritma sastrawi dibangun dari algoritma Nazief dan Adriani (NA) yang memiliki komputasi tinggi sehingga menyebabkan waktu proses stemming menjadi lebih lama.



Gambar 2 Langkah-langkah *preprocessing data*

Dalam proses *preprocessing* terdapat ketidakseimbangan data yang ditunjukkan pada gambar 3. Yaitu banyaknya dataset tweet positif dengan tweet negatif adalah hampir 1:3.



Gambar 3 Dataset yang tidak imbang

Data tidak seimbang merupakan suatu keadaan dimana distribusi kelas data tidak seimbang. Jumlah kelas data (*instance*) yang satu lebih sedikit atau lebih banyak dibanding dengan jumlah kelas data lainnya. Kelompok kelas data yang lebih sedikit dikenal dengan kelompok minoritas (*minority*), kelompok kelas data yang lainnya disebut dengan kelompok mayoritas (*mayority*).

Terdapat berbagai *treatment* untuk menangani *imbalance data* salah satunya adalah metode SMOTE. Metode *Synthetic Minority Over-sampling Technique* (SMOTE) merupakan metode yang populer diterapkan dalam menangani ketidak seimbangan kelas. Teknik ini mensintesis sampel baru dari kelas minoritas untuk menyeimbangkan dataset dengan cara sampling ulang pada sampel kelas minoritas.

Namun pada kasus ini kondisi *imbalance data* tidak dibahas (diabaikan) sehingga bisa saja akan berdampak pada model klasifikasi dimana kelas minoritas sering disalah klasifikasikan sebagai kelas mayoritas.

B. Ekstraksi Fitur

Proses ekstraksi fitur menggunakan metode pembobotan kata yaitu TF-IDF (Term Frequency Inverse Document Frecquency). Berikut persamaannya:

$$idf_j = log \frac{N}{df_j} \quad \text{persamaan (1)}$$

$$w_{ij} = tf_{ij} \times idf_j \quad \text{persamaan (2)}$$

Jika tf_{ij} menyatakan frekuensi dari term *i* yang muncul pada dokumen *j*, idf_j menyatakan inverse document frequency dari suatu term *j* terhadap keseluruhan dokumen, N menyatakan jumlah keseluruhan dokumen dan df_j menyatakan jumlah dokumen yang mengandung term, untuk menghitung bobot dari term w_{ij} digunakan persamaan (2).

C. Data training dan data testing

Langkah selanjutnya adalah membagi data menjadi data training dan data testing sebanyak 80% untuk data training dan 20% untuk data testing.

```
train_test_split(X,y,test_size=0.20, random_state=5)
```

Gambar 3 Kode program pembagian data testing

D. Pemodelan *Naïve Bayes*

Klasifikasi sentimen dengan naïve bayes dilakukan dengan cara membandingkan nilai (bobot) kata pada dokumen dalam dataset. Bila nilai (bobot) kata peluang (probabilitas) yang berkategori positif lebih banyak maka hasil sentimennya positif, bila nilai (bobot) kata peluang (probabilitas) yang berkategori negatif maka hasil sentimennya negatif Semua dokumen dataset akan diklasifikasi bila ditemukan nilai (bobot) pada tiap kata di dokumen dataset, dan data tidak terklasifikasi bila tidak ditemukan nilai (bobot) pada tiap kata di dokumen dataset.

Multinomial naïve bayes adalah implementasi dari naïve bayes untuk data yang multinomial terdistribusi dan diklasifikasikan dengan klasifikasi teks. Biasanya

membutuhkan perhitungan frekuensi kata dalam bentuk integer, seperti TF-IDF. Persamaan matematisnya sebagai berikut:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

persamaan (3)

Nyi adalah banyaknya fitur i yang muncul pada y, yaitu pada kasus ini adalah jumlah TF-IDF untuk term *i* pada kelas *y*. ***Ny*** adalah total dari kelas *y,* yaitu jika pada kasus ini adalah jumlah TF-IDF pada kelas y. α adalah nilai *smoothing prior* dimana $\alpha > 0$ untuk mencegah probability yang 0.

```
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.20, random_state=5)

# feature extraction with TF-IDF
vec = TfidfVectorizer(min_df=5, max_df=0.95, sublinear_tf
= True,use_idf = True,ngram_range=(1, 2))

X_train_vec = vec.fit_transform(X_train)
# modelling with multinomial naïve bayes
nb = MultinomialNB()
nb.fit(X_train_vec,y_train)

X_test_vec = vec.transform(X_test)
pred = nb.predict(X_test_vec)
```

Gambar 4 Kode program klasifikasi training set dengan multinomial naïve bayes

E. Evaluasi

Untuk mengetahui akurasi dari model yang telah dibangun digunakan matrik konfusion. *Confusion matrix* dapat diartikan sebagai suatu alat yang memiliki fungsi untuk melakukan analisis apakah *classifier* tersebut baik dalam mengenali tuple dari kelas yang berbeda. Nilai dari *True-Positive* dan *True-Negative* memberikan informasi ketika *classifier* dalam melakukan klasifikasi data bernilai benar, sedangkan *False-Positive* dan *False-Negative* memberikan informasi ketika *classifier* salah dalam melakukan klasifikasi data (Han dan Kamber, 2011).

		Predicted class		
		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	*P'*	*N'*	$P + N$

Gambar 5 Confusion matrix menampilkan total *positive* dan *negative tuple*

TP *(True Positive)* adalah jumlah data dengan nilai sebenarnya positif dan nilai prediksi positif. FP *(False Positive)* adalah jumlah data dengan nilai sebenarnya negatif dan nilai prediksi positif . FN *(False Negative)* adalah jumlah data dengan nilai sebenarnya positif dan nilai prediksi negatif TN *(True Negative)* adalah jumlah data dengan nilai sebenarnya negatif dan nilai prediksi negatif.

Sehingga akurasinya dapat dihitung sebagai berikut:

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN}$$

persamaan (4)

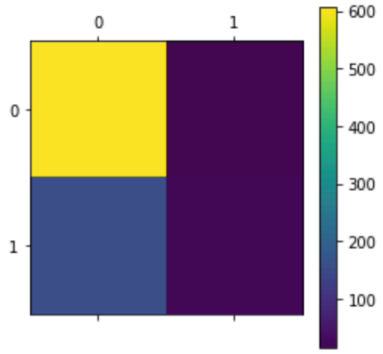
Dari pemodelan yang sudah dilakukan dan dengan menggunakan *library* yang sudah disediakan python (pada gambar 6) dihasilkan konfusi matrik sebagai berikut:

```
[[607  14]
 [156  23]]
```

```
print(metrics.confusion_matrix(y_test, pred))
```

Gambar 6 Kode program confusion matrix dengan python

Dengan menggunakan persamaan (4) akurasi yang dihasilkan adalah 78%. Visualisasi matrik konfusi dapat dilihat pada gambar 7.



Gambar 7 Diagram plot *confusion matrix*

Berikut adalah hasil precision, recall, f1-score dan support

	precision	recall	f1-score	support
0	0.80	0.98	0.88	621
1	0.62	0.13	0.21	179

III. HASIL DAN PEMBAHASAN

Akurasi yang hanya mencapai 78% dapat dipengaruhi oleh beberapa faktor, yaitu: preprocessing data dan adanya ketidak seimbangan dataset yang tersedia.

Berikut perbandingan akurasi data dengan tahapan preprocessing data:

Tabel 1 Pengaruh preprocessing data terhadap akurasi

Cleaning	Stopword Removal	Stemming	Akurasi
√	√	√	0.78
√	√	-	0.79
√	-	-	0.79

Berdasarkan tabel 1 maka proses *proprocessing data* tidak mempengaruhi akurasi secara signifikan. Hal tersebut bisa saja karena faktor dataset yang tidak seimbang *(imbalance)* atau *library* stopword dan stemming yang kurang kaya.

IV. KESIMPULAN

Algoritma multinomial naïve bayes dapat digunakan untuk melakukan pemodelan sentimen analisis menghasilkan akurasi hingga 78% dengan ekstraksi fitur menggunakan algoritma TF-IDF. Preprocessing data yang dilakukan adalah *data cleaning*, tokenisasi, *stopword removal* dan *stemming* menggunakan algoritma sastrawi dengan mengabaikan ketidakseimbangan dataset.

PUSTAKA

[1] Yasid M. & Junaedi L. Analisis Sentimen Maskapai Citilink Pada Twitter Dengan Metode Naïve Bayes. Jurnal Ilmiah Informatika Vol. 07 No. 02. 2019.

[2] Syahputra H., dkk. Sentiment Analysis of Public Opinion on The Go-Jek Indonesia Through Twitter Using Algoritma Support Vector Machine. Journal of Physics: Conf. Series 1462. 2020.

[3] Han, J. dan M. Kamber. 2001. Data Mining: Concepts and Techniques . *Tutorial*. Morgan Kaufman Publisher. San Francisco

[4] Agastya I Made A., dkk. Pengaruh Stemmer Bahasa Indonesia Terhadap Performa Analisis Sentimen Terjemahan Ulasan Film. Jurnal TEKNOKOMPAK Vol 12 No. 1. 2918

[5] Siringoringo R. Klasifikasi Data Tidak Seimbang Menggunakan Algoritma SMOTE dan k-Nearest Neighbor. Jurnal ISD Vol. 03 No.1. Januari-Juni 2018.

[6] https://github.com/shivanisoman/Twitter-Sentiment-Analysis. April 2019.

LAMPIRAN KODE PROGRAM

In [1]:
```
import nltk
import re
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
import pylab as pl

from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory, StopWordRemover, ArrayDictionary
```

In [2]:
```
def preprocess(tweet):

    #Convert www.* or https?://* to URL
    tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))','URL',tweet)

    #Convert @username to AT_USER
    tweet = re.sub('@[^\s]+','AT_USER',tweet)

    #Replace #word with word
    tweet = re.sub(r'#([^\s]+)', r'\1', tweet)

    #trim
    tweet = tweet.strip('\'"')

    # Repeating words like hellloooo
    repeat_char = re.compile(r"(.)\1{1,}", re.IGNORECASE)
    tweet = repeat_char.sub(r"\1\1", tweet)

    #Emoticons
    emoticons = \
    [
     ('__positive__',[ ':-)', ':)', '(:', '(-:', \
                       ':-D', ':D', 'X-D', 'XD', 'xD', \
                       '<3', ':\*', ';-)', ';)', ';-D', ';D', '(;', '(-;', ] ),\
     ('__negative__', [':-(', ':(', '(:', '(-:', ':,(',\
                       ':\'(', ':"(', ':((','D:' ] ),\
    ]

    def replace_parenthesis(arr):
       return [text.replace(')', '[)}\]]').replace('(', '[({\[') for text in arr]

    def join_parenthesis(arr):
        return '(' + '|'.join( arr ) + ')'

    emoticons_regex = [ (repl, re.compile(join_parenthesis(replace_parenthesis(regx))) ) \
            for (repl, regx) in emoticons ]

    for (repl, regx) in emoticons_regex :
        tweet = re.sub(regx, ' '+repl+' ', tweet)

    # Convert to lower case (Case folding)
    tweet = tweet.lower()

    return tweet
```

In [3]:
```
#start getStopWordList
def getStopWordList():
    #read the stopwords file and build a list
    stopWords = []

    fp = open('trainingandtestdata/stopwordsID.txt', 'r')
    line = fp.readline()
    while line:
        word = line.strip()
        stopWords.append(word)
        line = fp.readline()
    fp.close()
    return stopWords
```

In [4]:
```
#Stemming of Tweets

def stem(tweet):
        # create stemmer
        factory = StemmerFactory()
        stemmer = factory.create_stemmer() #bahasa
        # stemmer = nltk.stem.PorterStemmer() # english
        tweet_stem = ''
        words = [word if(word[0:2]=='__') else word.lower() \
                    for word in tweet.split() \
                    if len(word) >= 3]
        words = [stemmer.stem(w) for w in words]
        tweet_stem = ' '.join(words)

        return tweet_stem
```

In [5]:
```
def stopWordRemoval(tweet):
    # Get default stopword
    stop_factory = StopWordRemoverFactory().get_stop_words()
    more_stopword = getStopWordList()

    # Merge stopword
    data = stop_factory + more_stopword

    dictionary = ArrayDictionary(data)
    str = StopWordRemover(dictionary)

    return str.remove(tweet)
```

In [6]:
```
dataset = pd.read_csv('trainingandtestdata/gojek_twitter_dataset.csv',encoding='ISO-8859-1')
dataset
```

Out[6]:

	tweet	sentimen
0	Saya juga mau vouchee @gojekindonesia https:/...	1
1	download gojek duluuu uwuwu	1
2	Aminnn...#orderan goride mhn di lancar kan.all...	1
3	Tq @gojekindonesia @golifeindonesia ð ¤. Haru...	1
4	Semoga Twitter panjang umur. Berkomunikasi den...	1
5	Semoga di tahun yang baru ini, kita senantiasa...	1
6	Sejauh ini menurut saya UI paling nyaman dari ...	1
7	Thank you @gojekindonesia pic.twitter.com/pbZ...	1
8	Hai, ada yang bisa kami bantu mengenai layanan...	1
9	Full Week Feeling Great With You Guys @gojekin...	1
10	Happy Sunday everyone â ¤â ¤\nMinggu santai me...	1
11	You can donate via @gojekindonesia application...	1
12	Jogja terasa sangat indah pagi ini dengan MOTO...	1
13	Satu kesuksesan pelatih adalah di saat pemain ...	1
14	Engga sih, soalnya gua udah instal aplikasi @g...	1
15	Tgl 28 kemaren baru d tf bpk 200rb, sekarang u...	1
16	Pasti selalu Ada jalannya tenang aja ye gak @g...	1
17	wkwkwk fakta 5 thn terakhir	1
18	Ucapan "Selamat Pagi" dari abang gocar pun bis...	1

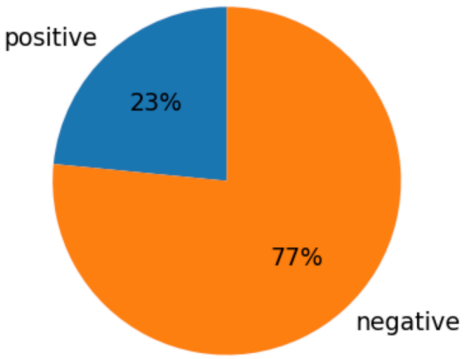
In [7]:
```
dataset.sentimen.value_counts()
```

Out[7]:
```
0    3062
1     938
Name: sentimen, dtype: int64
```

In [8]:
```
import matplotlib.pyplot as plt

amount_of_positive = dataset.sentimen.value_counts()[1]
amount_of_negative = dataset.sentimen.value_counts()[0]
category_names = ['positive', 'negative']
sizes = [amount_of_positive, amount_of_negative]

plt.figure(figsize=(2, 2), dpi=227)
plt.pie(sizes, labels=category_names, textprops={'fontsize': 6}, startangle=90,
       autopct='%1.0f%%')
plt.show()
```



In [9]:
```
X=dataset.iloc[:,0].values
X=pd.Series(X)
y=dataset.iloc[:,1].values

X
```

Out[9]:
```
0       Saya juga mau vouchee @gojekindonesia  https:/...
1                           download gojek duluuu uwuwu
2       Aminnn...#orderan goride mhn di lancar kan.all...
3       Tq @gojekindonesia @golifeindonesia ð    ¤. Haru...
4       Semoga Twitter panjang umur. Berkomunikasi den...
5       Semoga di tahun yang baru ini, kita senantiasa...
6       Sejauh ini menurut saya UI paling nyaman dari ...
7       Thank you @gojekindonesia  pic.twitter.com/pbZ...
8       Hai, ada yang bisa kami bantu mengenai layanan...
9       Full Week Feeling Great With You Guys @gojekin...
10      Happy Sunday everyone â  ¤â  ¤\nMinggu santai me...
11      You can donate via @gojekindonesia application...
12      Jogja terasa sangat indah pagi ini dengan MOTO...
13      Satu kesuksesan pelatih adalah di saat pemain ...
14      Engga sih, soalnya gua udah instal aplikasi @g...
15      Tgl 28 kemaren baru d tf bpk 200rb, sekarang u...
16      Pasti selalu Ada jalannya tenang aja ye gak @g...
17                          wkwkwk fakta 5 thn terakhir
18      Ucapan "Selamat Pagi" dari abang gocar pun bis...
19      dear @gojekindonesia ada ide baru nih buat nga...
```

In [10]:
```
# preprocessing
X = [preprocess(tweet) for tweet in X]
# stop word removal
X = [stopWordRemoval(tweet) for tweet in X]
# stemming
X = [stem(tweet) for tweet in X]
X
```

Out[10]:
```
['vouchee url ',
 'download gojek duluu uwuwu',
 'aminn order goride mhn lancar kan all driver kecuali pakai mod tuyul',
 '   5 0 khusus daerah bandung therapist hubungin nomor kartu pic twitter com 5jmnmnmnhq',
 'moga twitter umur komunikasi sedia jasa indonesia never been this easy just mentioned once tanggap and fast respons
e twitter long live twitter',
 'moga ini senantiasa jalan lurus jalan baik jalan tuntun pintu surga selamat islam muharam 1441 islamicnewyear tahun
baruislam muharram1441 pastiadajalan pic twitter com qpabdmyzim',
 'nyaman app asli indonesia punya',
 'thank you pic twitter com pbzbg8lkf7',
 'hai bantu layan gojek terima kasih yun',
 'full week feeling great with you guys terima kasih terima kasih gojek grab driver for this week  moga penuh berkah
kirim tulus halau url ',
 'happy sunday everyone  minggu santai jelang makan siang yuk order gofood powered at user available ikan bakar goren
g nila mas mujaer bawal lele  ikan bakar biduri url',
 'you can donate via application help those need using bts jungkook name at user at user pic twitter com 15pkll2yqs',
 'jogja indah pagi motoranwaeovo thankyou again saing',
 'sukses latih main main maksimal latih kalah menang tetapi main main maksimal keluar mampu menang tanding thx at use
r pastiadajalan septemberceria',
```

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.20, random_state=5)


vec = TfidfVectorizer(min_df=5, max_df=0.95, sublinear_tf = True,use_idf = True,ngram_range=(1, 2))
X_train_vec = vec.fit_transform(X_train)
nb = MultinomialNB()
nb.fit(X_train_vec,y_train)
X_test_vec = vec.transform(X_test)
pred = nb.predict(X_test_vec)

print(metrics.accuracy_score(y_test, pred))
```

```
0.7875
```

In [12]:
```
y_test.mean()
```

Out[12]:
```
0.22375
```

In [13]:
```
1-y_test.mean()
```

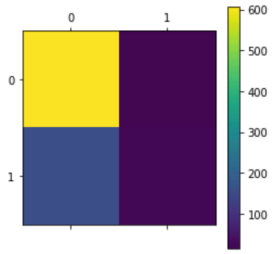
Out[13]:
```
0.77625
```

In [14]:
```
print(metrics.confusion_matrix(y_test, pred))
```

```
[[607  14]
 [156  23]]
```

In [15]:
```
cm = metrics.confusion_matrix(y_test, pred)
pl.matshow(cm)
#pl.title('Confusion matrix of the classifier')
pl.colorbar()
pl.show()

print(metrics.classification_report(y_test, pred))
```




```
              precision    recall  f1-score   support

           0       0.80      0.98      0.88       621
           1       0.62      0.13      0.21       179

   micro avg       0.79      0.79      0.79       800
   macro avg       0.71      0.55      0.55       800
weighted avg       0.76      0.79      0.73       800
```