Sentiment Analysis - Preprocessing and Analysis

Data yang digunakan dalam sesi praktikum ini merupakan data yang telah berlabel dalam format csv yang tersedia secara publik dan dapat diunduh pada situs berikut https://www.figure-eight.com/data-for-everyone/. Untuk kepentingan praktikum ini tidak perlu mengunduh data masing-masing, gunakanlah data yang telah disediakan.

A. Menyusun features untuk analisis berdasarkan kata dengan frekwensi terbesar dari seluruh tweet

1. Import modul yang dibutuhkan

```
import nltk
import random
from sklearn.model_selection import train_test_split
from nltk.classify.scikitlearn import SklearnClassifier
import pickle
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.linear_model import LogisticRegression, SGDClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC
from nltk.classify import ClassifierI
from statistics import mode
from nltk.tokenize import word_tokenize
import re
import csv
from nltk.corpus import stopwords
import string
import emoji
```

2. Membaca dataset

```
with open('dataset/1377884570_tweet_global_warming.csv', 'r') as nodecsv: # Buka file
    csvreader = csv.reader(nodecsv) # membaca data
    # Menyusun data dalam list dan menghilangkan header data
    csv = [n for n in csvreader][1:]
```

3. Menyusun regular expression untuk tokenisasi tweet

```
emoticons_str = r"""
    (?:
        [:=;] # Eyes
        [oO\-]? # Nose (optional)
        [D\)\]\(\]/\\OpP] # Mouth
    )"""
regex_str = []
regex_str.append(emoticons_str)
regex_str.append(r'<[^>]+>')# HTML tags
regex_str.append(r'(?:@[\w_]+)')# @-mentions
regex_str.append(r'(?:&[\w_]+)')
regex_str.append(r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)") # hash-tags
regex_str.append(r'http[s]?://(?:[a-z]|[0-9]|[$-_@.&amp;+]|[!*\(\),]|(?:%[0-9a-f][0-9a-f]))+') # URLs
regex_str.append(r'(?:(?:\d+,?)+(?:\.?\d+)?)') # numbers
regex_str.append(r"(?:[a-z][a-z'\-_]+[a-z])") # words with - and '
regex_str.append(r'(?:[\w_]+)') # other words
regex_str.append(r'(?:\S)') # anything else

tokens_re = re.compile(r'('+'|'.join(regex_str)+')', re.VERBOSE | re.IGNORECASE)
emoticon_re = re.compile(r'^'+emoticons_str+'$', re.VERBOSE | re.IGNORECASE)
```

4. Menyiapkan pengenal untuk stopword dan tanda baca

```
punctuation = list(string.punctuation)
stop = stopwords.words('english') + punctuation + ['rt', 'via', '…','•']
```

5. Menyiapkan fungsi untuk tokenisasi tweet

```
def tokenize(s):
    tokens = tokens_re.findall(s)
    return tokens

def cleanTweet(token,regex):
    terms_all = [emoji.demojize(term) for term in token if term.lower() not in stop and not regex.match(term)]
    return terms_all
```

6. Tokenisasi dan menghitung frekwensi kata dari seluruh tweet

```
tokens = []
for c in csv:
    tokens.append(tokenize(c[0]))

print(tokens[:5])
```

```
import operator

exclude_str = []
exclude_str.append(emoticons_str)
exclude_str.append(r'<[^>]+>')# HTML tags
exclude_str.append(r'(?:@[\w_]+)')# @-mentions
exclude_str.append(r'(?:&[\w_]+)')
exclude_str.append(r"(?:\#+[\w_]+[\w\'_\-]*[\w_]+)") # hash-tags
exclude_str.append(r'http[s]?://(?:[a-z]|[0-9]|[$-_@.&amp;+]|[!*\(\),]|(?:%[0-9a-f][0-9a-f]))+') # URLs
exclude_str.append(r'(?:(?:\d+,?)+(?:\.?\d+)?)') # numbers

exclude_re = re.compile(r'('+'|'.join(exclude_str)+')', re.VERBOSE | re.IGNORECASE)
all_word=[]
from collections import Counter
count_all = Counter()
for token in tokens:
   cleanToken = cleanTweet(token,exclude_re)
   terms_all = [term.lower() for term in cleanToken]
   count_all.update(terms_all)
   for w in terms_all:
      all_word.append(w)
print(count_all.most_common(100))
```

7. Menyusun features dengan filter berdasarkan jenis kata (adjective, adverb, verb, atau noun)

```
nltk.download('averaged_perceptron_tagger')
pos = nltk.pos_tag(all_word)
```

```
#  j = adjective, r = adverb, n=noun and v = verb
allowed_word_types = ["N","J","V"]
all_adj=[w[0] for w in pos if w[1][0] in allowed_word_types]
```

```
#Menyusun distribusi kata berdasarkan kemunculannya
allwords = nltk.FreqDist(all_adj)

#Mengambil 5000 pertama dari distribusi kata berdasarkan frekwensi kemunculannya
word_features = list(allwords.keys())[:5000]

#Memeriksa word_features
print(word_features[:100])
```

B. Menyusun data training dan testing untuk analisis

1. Membentuk tuple yang berisi tweet dan labelnya, hanya data dengan label Yes, Y, No dan N yang dimasukan ke dalam tuple

```
document=[]
for w in csv:
   categ=""
   if w[1] in ['Yes','Y', '5']:
      categ="pos"
      document.append((w[0],categ))
   if w[1] in ['No','N', '1']:
      categ="neg"
      document.append((w[0],categ))
```

2. Membuat fungsi untuk mencari kemunculan feature word dalam setiap tweet

```
def find_features(document):
   words = word_tokenize(document)
   features = {}
   for w in word_features:
      features[w] = (w in words)
   return features
```

3. Menyusun featureset yang berisi feature word dan ada/tidaknya feature tersebut dalam tweet

```
featuresets = [(find_features(rev), category) for (rev, category) in document]
```

```
print(featuresets[:1])
```

4. Menyusun training data dan testing data dari featureset yang telah terbentuk (hitung dahulu jumlah featureset kemudian bagi featureset tersebut menjadi training dan testing fata)

```
# menyusun data training dan testing
trainsize = round(len(featuresets)*0.7)

random.shuffle(featuresets)
training_set = featuresets[:trainsize]
testing_set = featuresets[trainsize:]
```

5. Menyusun model klasifikasi dengan algoritma Naïve Bayesian dan mengukur akurasinya

```
classifier = nltk.NaiveBayesClassifier.train(training_set)

print("Classifier accuracy percent:",(nltk.classify.accuracy(classifier, testing_set))*100)
classifier.show_most_informative_features(25)
```

6. Menampilkan hasil prediksi dengan model naïve Bayesian

```
for (rev, category) in document[trainsize:]:
   result=classifier.classify(find_features(rev))
   print(rev,"-",category,"-",result)
```

C. Tugas

1. Lakukan pemodelan klasifikasi dengan algoritma SVC, kemudian ukur tingkat akurasinya;
2. Lakukan pemodelan klasifikasi dengan algoritma Logistic Regression, kemudian ukur tingkat akurasinya;
3. Bandingkan akurasi dari model dengan algoritma Naïve Bayesian, SVC, dan Logistic Regression, kemudian tentukan model yang paling baik berdasarkan tingkat akurasi.