BAB 8 – TEXT CLASSIFICATION

TUJUAN BELAJAR

1. Memahami dan menerapkan *Text Classification* menggunakan Python

DASAR TEORI

- Text classification merupakan istilah untuk pengkategorian data teks berlabel berdasarkan ketentuan yang telah diberikan
- Data teks dapat memiliki struktur semi maupun teks bebas
- Contoh data yang memiliki struktur semi yaitu html, json, iklan koran, daftar Pustaka, dsb
- contoh data teks bebas yaitu esai, berita, buku, dsb.
- Dalam python, toolkit yang dapat digunakan untuk melakukan text mining salah satunya adalah NLTK
- Untuk menginstall NLTK, cukup lakukan perintah pip install nltk dan lakukan import
- Tokenisasi merupakan langkah awal dalam pemrosesan teks, langkah ini mengubah sebuah teks menjadi list yang berisi elemen berupa setiap kata dari teks tersebut.
- Untuk melakukan tokenisasi, perintah string.split() dapat dilakukan, atau menggunakan NLTK

```
import nltk


text = "In Brazil they drive on the right-hand side of the
road. has a large coastline on the eastern side of South
America"

from nltk.tokenize import word_tokenize
token = word_tokenize(text)
token
```

```
['In','Brazil','they','drive', 'on','the', 'right hand',
'side', 'of', 'the', 'road', '.', 'Brazil', 'has', 'a',
'large', 'coastline', 'on', 'the', 'eastern', 'side', 'of',
'South', 'America']
```

- Setelah tokenisasi, kita dapat mengetahui berapa kali sebuah kata muncul dalam teks tersebut dengan mencari frekuensinya
- Frekuensi kata yang muncul dalam sebuah teks dapat menjadi tanda awal tingkat kepentingan kata tersebut dalam korpus
- Untuk mencari frekuensi kata, dapat menggunakan fungsi FreqDist dari NLTK

```
from nltk.probability import FreqDist
fdist = FreqDist(token)
fdist

output:
"FreqDist({'the': 3, 'Brazil': 2, 'on': 2, 'side': 2, 'of':
2, 'In': 1, 'they': 1, 'drive': 1, 'right-hand': 1, 'road':
1, ...})"
```

- Atau, dengan menggunakan method most_common untuk mencari beberapa kata dengan frekuensi tertinggi dalam korpus

```
from nltk.probability import FreqDist
fdist = FreqDist(token)
fdist1 = fdist.most_common(10)
fdist1

output:
[('the', 3),
 ('Brazil', 2),
 ('on', 2),
 ('side', 2),
 ('of', 2),
 ('In', 1),
 ('they', 1),
 ('drive', 1),
 ('right-hand', 1),
 ('road', 1)]
```

- Untuk Bahasa Indonesia, terdapat library stemming dan lemma dengan nama PySastrawi

In [138]:
```
# import Sastrawi package
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

# create stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# token without stopword
list_tokens = tokens_without_stopword

# stem
output   = [(token + " : " + stemmer.stem(token)) for token in list_tokens]

output
```

Out[138]:
```
['positif : positif',
 'virus : virus',
 'corona : corona',
 'april : april',
 'orang : orang',
 'pasien : pasien',
 'sembuh : sembuh',
 'ri : ri',
 'meninggal : tinggal']
```

- Stopwords merupakan kata yang paling banyak muncul dalam sebuah teks. Biasanya tidak memiliki makna tertentu, contoh “the” “an” “a” “on” “is”.

```
from nltk.corpus import stopwords

# tokenize text
freq_tokens

# get Indonesian stopword
list_stopwords = set(stopwords.words('indonesian'))


#remove stopword pada list token
tokens_without_stopword = [word for word in
freq_tokens if not word in list_stopwords]


print(tokens_without_stopword)
```

```
from nltk import word_tokenize
from nltk.corpus import stopwords
a = set(stopwords.words('english'))
text = "Cristiano Ronaldo was born on February 5,
1985, in Funchal, Madeira, Portugal."

text1 = word_tokenize(text.lower())

print(text1)stopwords = [x for x in text1 if x not in
a]
print(stopwords)

output:
['cristiano', 'ronaldo', 'born', 'february', '5',
'1985',  'funchal' , 'madeira', 'portugal']
```

- Setelah proses lemmatisasi dan penghilangan stopwords dilakukan, hal yang dapat dilakukan selanjutnya yaitu proses vektorisasi
- Vektorisasi kata ada berbagai metode, yakni TF-IDF untuk pembobotan kata berdasarkan relevansi kata pada kumpulan dokumen, dan Word Embedding untuk vektorisasi kata berdasarkan makna relative sebuah kata dimana kata tersebut digunakan dalam sebuah kalimat.
- Untuk mengetahui performa klasifikasi, jangan lupa lakukan train test split agar kita mempunyai data untuk di test
-
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(
    df['text'], df['encoded_cat'], test_size=.2, stratify=df['category'], random_state=42)
```
- Menggunakan TF IDF, jalankan kode berikut
-
```
from sklearn.feature_extraction.text import TfidfVectorizer
# initialize the vectorizer
vectorizer = TfidfVectorizer(sublinear_tf=True, min_df=5, max_df=0.95)
# fit_transform applies TF-IDF to clean texts - we save the array of vectors in X
X = vectorizer.fit_transform(df['cleaned'])
```
- Untuk menggunakan random forest, jalankan kode berikut
-
```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier()
clf.fit(X, Y)
pred = clf.predict(vectorizer.transform(X_test))
print(metrics.classification_report(test_y, pred))
```
- Untuk MultinomialNB, jalankan kode berikut

```
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics

naive_bayes_classifier = MultinomialNB()
naive_bayes_classifier.fit(X_train_tf, train_y)
y_pred = naive_bayes_classifier.predict(X_test_tf)

print(metrics.classification_report(test_y, y_pred))
```

- Untuk menggunakan neural network, gunakan tensorflow / pytorch untuk mendefinisikan konfigurasi tiap layernya. Seluruh metode diatas dengan asumsi bahwa data teks sudah direpresentasikan dengan vector

```
import tensorflow as tf
import pandas as pd
import numpy as np
import scipy
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split

model = tf.keras.Sequential([
   tf.keras.layers.Dense(48, activation='relu', input_shape=.    (x_train.shape[1],)),
   tf.keras.layers.Dropout(0.2),
   tf.keras.layers.Dense(24, activation='relu'),
   tf.keras.layers.Dropout(0.1),
   tf.keras.layers.Dense(df['category'].nunique(),      activation='softmax')
])
```

-

```
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()

history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test), verbose=2)
```

-

LATIHAN DAN TUGAS PRAKTIKUM

1. Download data teks dari halaman berikut: https://raw.githubusercontent.com/ruzcmc/ClickbaitIndo-textclassifier/master/all_agree.csv
2. Lakukan classification menggunakan metode MultinomialNB, RandomForest, dan metode klasifikasi pilihan kalian (terserah) dengan menggunakan fitur TF-IDF
3. Buat perbandingan performa ketiga metode supervised learning tersebut berdasarkan performanya (poin plus: lakukan cross validation)
4. Dokumentasikan setiap langkah diatas dan tulis laporannya
5. Kumpulkan dalam format pdf