IF4072 Pemrosesan Teks dan Suara Bahasa Alami

Anggota Kelompok

1. 13515021 Dewita Sonya Tarabunga
2. 13515057 Erick Wijaya
3. 13515107 Roland Hartanto

Import

In [1]:

```
import os, stat, string, re
import pandas as pd
import numpy as np
import nltk
import math

from collections import Counter
from nltk import word_tokenize, NaiveBayesClassifier, DecisionTreeClassifier, MaxentClassifier,
classify
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk.corpus import wordnet, stopwords
```

Load Data

In [2]:

```
def load_data(src, size):
    emails = ['' for _ in range(size)]
    if not os.path.exists(src):
        os.makedirs(src)
    files = os.listdir(src)
    for file in files:
        idx = file.replace('TRAIN_','').replace('.eml','').replace('TEST_', '')
        path = os.path.join(src, file)
        info = os.stat(path)
        if not stat.S_ISDIR(info.st_mode):
            fp = open(path, encoding='utf-8', errors='ignore')
            body = fp.read()
            fp.close()

            emails[int(idx) - 1] = body
    return emails

labels = pd.read_csv('spam-mail.tr.label')
labels = labels['Prediction'].tolist()
emails = load_data('TR-extracted', 2500)
test = load_data('TT-extracted', 1827)
```

Preprocessing

Remove Punctuations

In [3]:

```
trans = str.maketrans(string.punctuation, ' ' * len(string.punctuation))
emails = [email.translate(trans) for email in emails]
test = [email.translate(trans) for email in test]
```

Tokenize

In [4]:

```
tokens = [word_tokenize(email.lower()) for email in emails]
test = [word_tokenize(email.lower()) for email in test]
```

Lemmatization

In [5]:

```
tagged = [nltk.pos_tag(token) for token in tokens]
test_tagged = [nltk.pos_tag(token) for token in test]
```

In [6]:

```
def get_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

lemmatizer = WordNetLemmatizer()
tokens = [[lemmatizer.lemmatize(word[0], pos=get_pos(word[1])) for word in token] for token in tagg
ed]
test = [[lemmatizer.lemmatize(word[0], pos=get_pos(word[1])) for word in token] for token in test_t
agged]
```

Feature Selection

Stop Words Elimination

In [7]:

```
stop_words = set(stopwords.words('english'))
tokens = [[word for word in token if not word in stop_words] for token in tokens]
test = [[word for word in token if not word in stop_words] for token in test]
```

Convert to Trainable Data

In [8]:

```
def get_features(text):
    return {word: True for word in text}

train_data = []
idx = 0
for email in tokens:
    train_data.append((get_features(email), labels[idx]))
    idx += 1

test_data = []
idx = 0
for email in test:
    test_data.append(get_features(email))
    idx += 1
```

Training Model

Naive Bayes

In [9]:

```
classifierNB = NaiveBayesClassifier.train(train_data)
```

Decision Tree

In [10]:

```
classifierDT = DecisionTreeClassifier.train(train_data, depth_cutoff=30)
```

Maximum Entropy

In [11]:

```
classifierMaxentGIS = MaxentClassifier.train(train_data, 'GIS', trace=0, max_iter=500)
```

Classification

Naive Bayes

In [12]:

```
fp = open('answerNB.csv', 'w')
fp.write('Id,Prediction\n')

idx = 1
for data in test_data:
    ans = classifierNB.classify(data)
    fp.write(str(idx)+','+str(ans)+'\n')
    idx += 1

fp.close()
```

In [13]:

```
classify.accuracy(classifierNB, train_data)
```

Out[13]:

```
0.9636
```

Decision Tree

In [14]:

```
fp = open('answerDT.csv', 'w')
fp.write('Id,Prediction\n')

idx = 1
for data in test_data:
    ans = classifierDT.classify(data)
    fp.write(str(idx)+','+str(ans)+'\n')
    idx += 1

fp.close()
```

In [15]:

```
classify.accuracy(classifierDT, train_data)
```

Out[15]:

0.9744

Maximum Entropy

In [16]:

```
fp = open('answerMaxentGIS.csv', 'w')
fp.write('Id,Prediction\n')

idx = 1
for data in test_data:
    ans = classifierMaxentGIS.classify(data)
    fp.write(str(idx)+','+str(ans)+'\n')
    idx += 1

fp.close()
```

In [17]:

```
classify.accuracy(classifierMaxentGIS, train_data)
```

Out[17]:

0.95