

Tugas *Sentiment Analysis*

IF5153 Pemrosesan Bahasa Alami



Penulis:

Melvin Kent Jonathan

13521052

25 September 2024

1.Kode Program

Library

```
import pandas as pd
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.metrics import precision_score, accuracy_score, recall_score,
f1_score
from sklearn.metrics import classification_report
```

Program sentiment analysis ini menggunakan beberapa library sebagai berikut:

Libary	Kegunaan
pandas	Menampung dan melakukan pemrosesan sederhana terhadap data latih dan data uji
string	Melakukan <i>lowercase</i> pada <i>preprocessing</i>
nltk	Melakukan <i>preprocessing</i> seperti tokenisasi dan penghapusan <i>stop words</i> dan tanda baca
Sastrawi	Melakukan <i>preprocessing</i> seperti <i>stemming</i>
sklearn	Melakukan vektorisasi terhadap kata-kata dalam teks menjadi bag of words; Melakukan perhitungan metrik seperti akurasi, presisi, recall, dan f1 score; Menerapkan algoritme pembelajaran mesin tradisional, seperti <i>logistic regression</i> , <i>Support Vector Machine</i> (SVM), <i>k-Nearest Neighbors</i> (k-NN), dan Naive Bayes.

NLP Tools Initializastion

```
# create stemmer
```

```

stemmer_factory = StemmerFactory()
stemmer = stemmer_factory.create_stemmer()

# Download NLTK data for tokenization and stopwords removal
nltk.download('punkt')
nltk.download('stopwords')

# Load Indonesian stopwords from NLTK
stop_words = set(stopwords.words('indonesian'))

# Initialize the CountVectorizer (Bag of Words model)
binary_vectorizer = CountVectorizer(binary=True) # binary as value
tf_vectorizer = CountVectorizer() # count as value
tfidf_vectorizer = TfidfVectorizer() # tfidf as value

```

Bagian kode ini diperuntukkan sebagai tempat inisialisasi dari kaskas NLP yang digunakan pada program. Pada program dibutuhkan 3 jenis *vecotizer*, yakni binary, tf, dan tfidf yang akan dipakai untuk menerapkan variasi terhadap nilai fitur dari *bag of words*.

Preprocessing

```

# Define a function to preprocess the text
def preprocess_text(text):
    # 1. Convert text to lowercase
    text = text.lower()

    # 2. Tokenize the text
    tokens = word_tokenize(text)

    # 3. Remove stopwords and punctuation
    tokens = [word for word in tokens if word not in stop_words and word not
in string.punctuation]

    # 4. Stem the tokens
    tokens = [stemmer.stem(word) for word in tokens]

    # Rejoin the tokens into a single string
    processed_text = ' '.join(tokens)

    return processed_text

```

Bagian kode ini berisi fungsi yang dapat dipanggil untuk menerapkan *preprocessing* terhadap data latih dan data uji sebelum dimasukkan ke dalam model pembelajaran mesin. Adapun *preprocessing* yang dilakukan adalah:

1. Melakukan lowecasing terhadap teks.
2. Melakukan tokenisasi terhadap teks.
3. Melakukan penghapusan terhadap *stop words* dan tanda baca.
4. Melakukan *stemming* terhadap token

Metrics Function

```
# Define a function to display the metrics
def metrics(y_test, y_pred):
    # Calculate metrics
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred, average='macro')
    recall = recall_score(y_test, y_pred, average='macro')
    f1_score_value = f1_score(y_test, y_pred, average='macro')

    print(f'Accuracy : {accuracy:.4f}')
    print(f'Precision : {precision:.4f}')
    print(f'Recall : {recall:.4f}')
    print(f'F1-score : {f1_score_value:.4f}')

    # Generate and display classification report
    report = classification_report(y_test, y_pred, target_names=['Negative',
'Neutral', 'Positive'])
    print(report)
```

Bagian kode ini mendefinisikan fungsi *metrics* yang berfungsi sebagai evaluator terhadap kinerja prediksi dari model. Adapun evaluasi performa didasarkan kepada 4 metrik, yakni nilai akurasi, presisi, *recall*, dan *f1-score*.

Import Data for Training and Testing

```
headers = ['text', 'label']

# import train data
train = pd.read_csv("nlp_data/train_preprocess.tsv", sep='\t',
names=headers)

# import test data
test = pd.read_csv("nlp_data/test_preprocess.tsv", sep='\t',
names=headers)
```

```
# Preprocess the train data
train['preprocessed_text'] = train['text'].apply(preprocess_text)

# Preprocess the test data
test['preprocessed_text'] = test['text'].apply(preprocess_text)
```

Bagian kode ini melakukan *import* data latih dan data uji. Adapun data latih dan data uji diperoleh dari sumber berikut:

https://github.com/IndoNLP/indonlu/tree/master/dataset/smsa_doc-sentiment-prosa

Data latih dan data uji kemudian dikenakan fungsi `preprocess_text` guna memastikan data minim dari *noise* yang dapat mengurangi performa pembelajaran mesin.

Bag of Words using Binary, TF, TFxIDF

```
# Example preprocessed text data
train_preprocessed_texts = train['preprocessed_text'] # This column
contains the preprocessed text
test_preprocessed_texts = test['preprocessed_text']

y_train = train['label']
y_test = test['label']

# Fit and transform the preprocessed text data to generate the Bag of
Words matrix
## binary
X_train_binary = binary_vectorizer.fit_transform(train_preprocessed_texts)
X_test_binary = binary_vectorizer.transform(test_preprocessed_texts)

## TF
X_train_tf = tf_vectorizer.fit_transform(train_preprocessed_texts)
X_test_tf = tf_vectorizer.transform(test_preprocessed_texts)

## TFxIDF
X_train_tfidf = tfidf_vectorizer.fit_transform(train_preprocessed_texts)
X_test_tfidf = tfidf_vectorizer.transform(test_preprocessed_texts)
```

Bagian kode ini membuat data latih ke dalam representasi bag of words menggunakan *vectorizer* yang sudah diinisialisasi sebelumnya pada bagian **NLP Tools Initialization**. Adapun nilai dari fitur yang dipakai berupa nilai biner, Term Frequency (TF), dan Term Frequency-Inverse Document Frequency (TFxIDF).

Train and Evaluate Function

```
def train_and_evaluate(classifier, X_train_binary, X_train_tf,
X_train_tfidf, y_train, X_test_binary, X_test_tf, X_test_tfidf, y_test):
    # Train and make predictions
    ## Binary
    classifier.fit(X_train_binary, y_train)
    y_pred_binary = classifier.predict(X_test_binary)

    print("===== Binary =====")
    metrics(y_test, y_pred_binary)

    ## TF
    classifier.fit(X_train_tf, y_train)
    y_pred_tf = classifier.predict(X_test_tf)

    print("===== TF =====")
    metrics(y_test, y_pred_tf)

    ## TFxIDF
    classifier.fit(X_train_tfidf, y_train)
    y_pred_tfidf = classifier.predict(X_test_tfidf)

    print("===== TFxIDF =====")
    metrics(y_test, y_pred_tfidf)
```

Bagian kode ini berisi fungsi yang melakukan pelatihan menggunakan model yang diterima dari parameter fungsi. Pelatihan dilakukan untuk masing-masing jenis nilai dari fitur *bag of words*. Untuk setiap jenis, dipanggil pula fungsi metrics untuk memperlihatkan kinerja prediksi model.

Logistic Regression

```
from sklearn.linear_model import LogisticRegression

# Initialize the Logistic Regression classifier
classifier = LogisticRegression(max_iter=500)

# Train and Evaluate
train_and_evaluate(classifier, X_train_binary, X_train_tf, X_train_tfidf,
y_train, X_test_binary, X_test_tf, X_test_tfidf, y_test)
```

Support Vector Machines (SVM)

```
from sklearn.svm import SVC
```

```
# Initialize the SVM classifier
classifier = SVC(kernel='linear')

# Train and Evaluate
train_and_evaluate(classifier, X_train_binary, X_train_tf, X_train_tfidf,
y_train, X_test_binary, X_test_tf, X_test_tfidf, y_test)
```

k-Nearest Neighbors (k-NN)

```
from sklearn.neighbors import KNeighborsClassifier

# Initialize the k-NN classifier
classifier = KNeighborsClassifier(n_neighbors=3)

# Train and Evaluate
train_and_evaluate(classifier, X_train_binary, X_train_tf, X_train_tfidf,
y_train, X_test_binary, X_test_tf, X_test_tfidf, y_test)
```

Naive Bayes

```
from sklearn.naive_bayes import MultinomialNB

# Initialize the Naive Bayes classifier
classifier = MultinomialNB()

# Train and Evaluate
train_and_evaluate(classifier, X_train_binary, X_train_tf, X_train_tfidf,
y_train, X_test_binary, X_test_tf, X_test_tfidf, y_test)
```

Link github:

<https://github.com/melvinkj/Sentiment-Analysis-NLP>

2. Skenario Eksperimen

Skenario dari eksperimen yang dilakukan adalah dengan memilih 4 variasi model pembelajaran mesin tradisional, yakni *logistic regression*, *Support Vector Machine* (SVM), *k-Nearest Neighbors* (k-NN), dan Naive Bayes. Untuk setiap model pembelajaran, dilakukan pula variasi terhadap jenis nilai dari fitur bag of words, yakni biner biner, Term Frequency (TF), dan Term Frequency-Inverse Document Frequency (TFxIDF). Oleh karena itu, terdapat 12 kasus uji yang dijalankan.

3. Hasil Eksperimen

a. Logistic Regression

1) Binary

```
===== Binary =====
Accuracy : 0.7560
Precision : 0.7597
Recall : 0.7032
F1-score : 0.7181

      precision    recall  f1-score   support

Negative      0.73      0.88      0.80      204
Neutral       0.76      0.48      0.59       88
Positive      0.79      0.75      0.77      208

 accuracy
macro avg      0.76      0.70      0.72      500
weighted avg   0.76      0.76      0.75      500

Confusion Matrix
Predicted negative neutral positive All
Actual
negative      179      4      21  204
neutral        24     42      22   88
positive       42      9     157  208
All            245     55     200  500
```

2) TF

```
===== TF =====
Accuracy : 0.7440
Precision : 0.7325
Recall : 0.6934
F1-score : 0.7048

      precision    recall  f1-score   support

Negative      0.72      0.85      0.78      204
Neutral       0.69      0.48      0.56       88
Positive      0.79      0.75      0.77      208

 accuracy
macro avg      0.73      0.69      0.70      500
weighted avg   0.74      0.74      0.74      500

Confusion Matrix
Predicted negative neutral positive All
Actual
negative      173      7      24  204
neutral        27     42      19   88
positive       39     12     157  208
All            239     61     200  500
```


3) TFXIDF

===== TFXIDF =====				
Accuracy	: 0.7360			
Precision	: 0.7767			
Recall	: 0.6589			
F1-score	: 0.6732			
	precision	recall	f1-score	support
Negative	0.68	0.91	0.77	204
Neutral	0.85	0.33	0.48	88
Positive	0.80	0.74	0.77	208
accuracy			0.74	500
macro avg	0.78	0.66	0.67	500
weighted avg	0.76	0.74	0.72	500
Confusion Matrix				
Predicted	negative	neutral	positive	All
Actual				
negative	185	1	18	204
neutral	39	29	20	88
positive	50	4	154	208
All	274	34	192	500

b. Support Vector Machines (SVM)

1) Binary

===== Binary =====				
Accuracy	: 0.7060			
Precision	: 0.6927			
Recall	: 0.6669			
F1-score	: 0.6755			
	precision	recall	f1-score	support
Negative	0.69	0.79	0.74	204
Neutral	0.64	0.50	0.56	88
Positive	0.75	0.71	0.73	208
accuracy			0.71	500
macro avg	0.69	0.67	0.68	500
weighted avg	0.71	0.71	0.70	500
Confusion Matrix				
Predicted	negative	neutral	positive	All
Actual				
negative	162	15	27	204
neutral	23	44	21	88
positive	51	10	147	208
All	236	69	195	500

2) TF

```

===== TF =====
Accuracy : 0.7100
Precision : 0.6877
Recall : 0.6723
F1-score : 0.6776

              precision    recall  f1-score   support

   Negative      0.69      0.78      0.73      204
    Neutral      0.59      0.51      0.55       88
    Positive      0.78      0.72      0.75      208

   accuracy              0.71      500
  macro avg      0.69      0.67      0.68      500
 weighted avg      0.71      0.71      0.71      500

Confusion Matrix
Predicted negative neutral positive All
Actual
negative      160      17      27  204
neutral       28      45      15   88
positive      44      14     150  208
All           232      76     192  500

```

3) TFXIDF

```

===== TFXIDF =====
Accuracy : 0.7500
Precision : 0.7593
Recall : 0.6962
F1-score : 0.7116

              precision    recall  f1-score   support

   Negative      0.69      0.88      0.78      204
    Neutral      0.76      0.47      0.58       88
    Positive      0.82      0.74      0.78      208

   accuracy              0.75      500
  macro avg      0.76      0.70      0.71      500
 weighted avg      0.76      0.75      0.74      500

Confusion Matrix
Predicted negative neutral positive All
Actual
negative      180       3      21  204
neutral       35      41      12   88
positive      44      10     154  208
All           259      54     187  500

```

c. k-Nearest Neighbors (k-NN)

1) Binary

```
===== Binary =====
Accuracy : 0.5420
Precision : 0.5837
Recall : 0.4523
F1-score : 0.4329

              precision    recall  f1-score   support

   Negative      0.47      0.76      0.59      204
    Neutral      0.60      0.07      0.12       88
    Positive      0.68      0.52      0.59      208

   accuracy              0.54      500
  macro avg      0.58      0.45      0.43      500
 weighted avg      0.58      0.54      0.51      500

Confusion Matrix
Predicted negative neutral positive All
Actual
negative      156      3      45  204
neutral       75      6      7   88
positive      98      1     109  208
All          329     10     161  500
```

2) TF

```
===== TF =====
Accuracy : 0.5380
Precision : 0.5712
Recall : 0.4689
F1-score : 0.4654

              precision    recall  f1-score   support

   Negative      0.48      0.79      0.60      204
    Neutral      0.56      0.17      0.26       88
    Positive      0.68      0.45      0.54      208

   accuracy              0.54      500
  macro avg      0.57      0.47      0.47      500
 weighted avg      0.58      0.54      0.51      500

Confusion Matrix
Predicted negative neutral positive All
Actual
negative      161      6      37  204
neutral       66     15      7   88
positive     109      6     93  208
All          336     27    137  500
```

3) TFXIDF

```
===== TFXIDF =====
Accuracy : 0.4340
Precision : 0.3654
Recall : 0.3539
F1-score : 0.2530

              precision    recall  f1-score   support

   Negative      0.42      0.96      0.58       204
    Neutral      0.00      0.00      0.00        88
    Positive      0.68      0.10      0.18       208

   accuracy              0.43       500
  macro avg      0.37      0.35      0.25       500
 weighted avg      0.45      0.43      0.31       500

Confusion Matrix
Predicted negative neutral positive All
Actual
negative      196         0         8    204
neutral        86         0         2     88
positive      186         1        21    208
All           468         1        31    500
```

d. Naive Bayes

1) Binary

```
===== Binary =====
Accuracy : 0.6520
Precision : 0.6950
Recall : 0.5851
F1-score : 0.5917

              precision    recall  f1-score   support

   Negative      0.61      0.91      0.73       204
    Neutral      0.76      0.30      0.43        88
    Positive      0.71      0.55      0.62       208

   accuracy              0.65       500
  macro avg      0.70      0.59      0.59       500
 weighted avg      0.68      0.65      0.63       500

Confusion Matrix
Predicted negative neutral positive All
Actual
negative      186         0         18    204
neutral        34        26         28     88
positive        86         8        114    208
All           306        34        160    500
```

2) TF

```

===== TF =====
Accuracy : 0.6400
Precision : 0.6554
Recall : 0.5842
F1-score : 0.5874

              precision    recall  f1-score   support

   Negative      0.60      0.91      0.72      204
    Neutral      0.64      0.34      0.44       88
    Positive      0.73      0.50      0.60      208

   accuracy              0.64      500
  macro avg      0.66      0.58      0.59      500
weighted avg      0.66      0.64      0.62      500

Confusion Matrix
Predicted   negative   neutral   positive   All
Actual
negative      185         3         16      204
neutral        35        30         23       88
positive        89        14        105      208
All            309        47        144      500

```

3) TFxIDF

```

===== TFxIDF =====
Accuracy : 0.6660
Precision : 0.7763
Recall : 0.5453
F1-score : 0.5061

              precision    recall  f1-score   support

   Negative      0.68      0.79      0.73      204
    Neutral      1.00      0.03      0.07       88
    Positive      0.65      0.81      0.72      208

   accuracy              0.67      500
  macro avg      0.78      0.55      0.51      500
weighted avg      0.72      0.67      0.61      500

Confusion Matrix
Predicted   negative   neutral   positive   All
Actual
negative      162         0         42      204
neutral        37         3         48       88
positive        40         0        168      208
All            239         3        258      500

```

4. Error Analysis

a. Logistic Regression

Untuk Binary, model cenderung dapat memprediksi sentimen dengan label negative.
Untuk TF, model cenderung dapat memprediksi sentimen dengan label negative.
Untuk TFxIDF, model cenderung dapat memprediksi sentimen dengan label negative.

Pada TFxIDF, pelabelan negative cenderung lebih banyak ketimbang dengan jenis lainnya. Apabila dianalisis lebih lanjut, penambahan pelabelan negative berasal dari data-data dengan yang sebenarnya berlabel neutral.

b. Support Vector Machines (SVM)

Untuk Binary, model cenderung dapat memprediksi sentimen dengan label negative.
Untuk TF, model cenderung dapat memprediksi sentimen dengan label negative.
Untuk TFxIDF, model cenderung dapat memprediksi sentimen dengan label negative.

Pada TFxIDF, pelabelan negative cenderung lebih banyak ketimbang dengan jenis lainnya. Apabila dianalisis lebih lanjut, penambahan pelabelan negative berasal dari data-data dengan yang sebenarnya berlabel neutral.

c. k-Nearest Neighbors (k-NN)

Untuk Binary, model cenderung dapat memprediksi sentimen dengan label negative.
Untuk TF, model cenderung dapat memprediksi sentimen dengan label negative.
Untuk TFxIDF, model cenderung dapat memprediksi sentimen dengan label negative.

Pada TFxIDF, pelabelan negative mendominasi label lainnya secara signifikan. Proporsi pelabelan positive maupun neutral menjadi sangat berkurang, bahkan mendekati nol.

d. Naive Bayes

Untuk Binary, model cenderung dapat memprediksi sentimen dengan label negative.
Untuk TF, model cenderung dapat memprediksi sentimen dengan label negative.
Untuk TFxIDF, model cenderung dapat memprediksi sentimen dengan label positif.

Pada TFxIDF, pelabelan positive cenderung lebih banyak ketimbang dengan jenis lainnya. Apabila dianalisis lebih lanjut, penambahan pelabelan positive berasal dari data-data dengan yang sebenarnya berlabel neutral.

e. Analisis Menyeluruh

Secara keseluruhan, prediksi dari masing-masing model cenderung bias ke arah label negatif. Hal ini mungkin disebabkan oleh tahap *preprocessing* yang masih kurang optimal, terutama dalam memberikan konteks konotasi negatif yang biasanya diawali kata “tidak”, “ga”, ataupun “kurang”. Perbaikan yang dapat dilakukan adalah dengan menambah teknik *entity masking* serta *spelling correction* pada tahap *preprocessing*.

Link Github : <https://github.com/melvinkj/Sentiment-Analysis-NLP>

Sumber Data :

https://github.com/IndoNLP/indonlu/tree/master/dataset/smsa_doc-sentiment-prosa