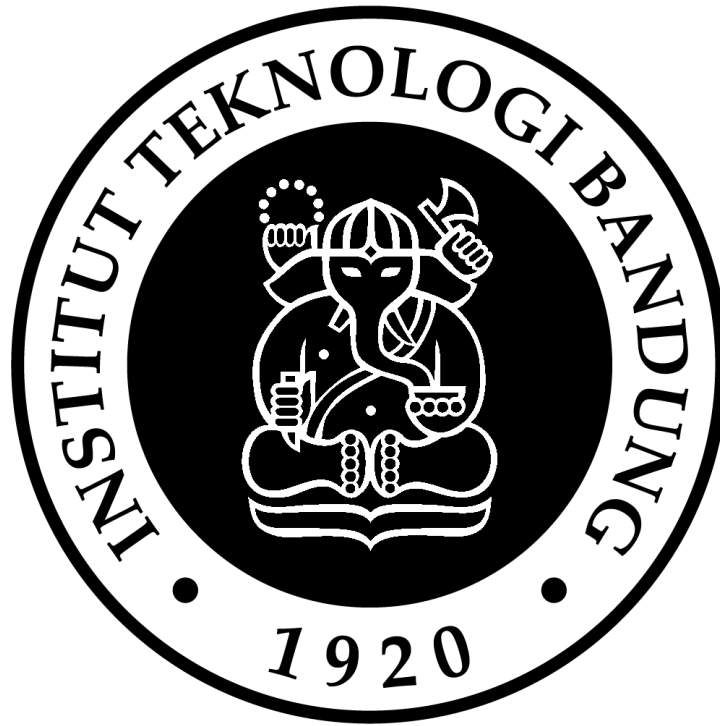


LAPORAN TUGAS PEMROGRAMAN IF5153
TEXT CLASSIFICATION MENGGUNAKAN SHALLOW MACHINE
LEARNING



Disusun Oleh:

13521131	Jeremya Dharmawan Raharjo
----------	---------------------------

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2024

DAFTAR ISI

DAFTAR ISI	2
DASAR TEORI	3
A. Sentiment Analysis	3
B. Data Preprocessing	3
C. Bag Of Words	4
D. Algoritma Machine Learning untuk Sentiment Analysis	5
EKSPERIMEN	6
Data Preprocessing	6
Modeling (Model Selection)	8
Model Testing	13
PEMBAHASAN	14
DAFTAR REFERENSI	15

DASAR TEORI

A. Sentiment Analysis

Sentiment analysis, yang merupakan bagian dari *Natural Language Processing* (NLP), sedang berkembang pesat seiring dengan meningkatnya kemampuan teknologi dalam memahami dan menginterpretasi bahasa manusia. Perkembangan ini memungkinkan *sentiment analysis* menjadi semakin akurat dan efisien dalam menganalisis berbagai jenis teks, mulai dari ulasan produk hingga interaksi media sosial. Saat ini, perusahaan memiliki sejumlah besar data teks, seperti email, transkrip obrolan layanan pelanggan, komentar media sosial, dan ulasan. Dengan bantuan metode *sentiment analysis*, teks-teks ini dapat dipindai secara otomatis untuk menentukan pandangan penulis terhadap suatu topik. Perusahaan memanfaatkan wawasan dari analisis ini untuk meningkatkan kualitas layanan pelanggan dan memperkuat citra merek mereka.

Sentiment analysis, yang juga dikenal sebagai *opinion mining*, merupakan alat penting dalam kecerdasan bisnis yang membantu perusahaan dalam menyempurnakan produk dan layanan mereka. Beberapa manfaat *sentiment analysis* antara lain:

- Menyediakan wawasan yang objektif
- Meningkatkan kualitas produk dan layanan
- Mampu menganalisis dalam skala besar
- Memberikan hasil secara real-time

B. Data Preprocessing

Pada task NLP, *data preprocessing* merupakan langkah penting untuk memastikan teks yang dianalisis siap digunakan oleh model *machine learning*. Proses ini melibatkan beberapa tahapan

untuk membersihkan dan menyederhanakan data teks agar dapat diproses dengan lebih efisien, yang meliputi:

- Lowercasing: Mengubah semua huruf menjadi huruf kecil agar perbedaan kapitalisasi tidak mempengaruhi analisis.
- Stopword Removal: Menghilangkan kata-kata umum seperti "dan", "atau", dan "dengan", yang tidak memiliki banyak arti dalam analisis sentimen.
- Tokenization: Memecah teks menjadi kata-kata atau token individu sehingga setiap kata dapat dianalisis secara terpisah.
- Stemming atau Lemmatization: Mengubah kata ke bentuk dasar atau akarnya (misalnya, "berlari" menjadi "lari") untuk mengurangi variasi kata dan fokus pada esensinya.
- Removing Punctuation and Special Characters: Menghapus tanda baca, angka, dan karakter khusus yang tidak relevan dalam analisis.

Data preprocessing memungkinkan model NLP untuk bekerja dengan teks yang lebih bersih dan lebih terstruktur yang dapat meningkatkan akurasi pada *task sentiment analysis*.

C. Bag Of Words

Matrix of Term x Document

Terms ↓	d1 ↓	d2 ↓	d3 ↓
a	1	1	1
arrived	0	1	1
damaged	1	0	0
delivery	0	1	0
fire	1	0	0
gold	1	0	1
in	1	1	1
of	1	1	1
shipment	1	0	1
silver	0	2	0
truck	0	1	1

Bag of Words merupakan metode representasi teks yang sering digunakan dalam sentiment analysis. Teknik ini mengabaikan struktur tata bahasa atau urutan kata, dan hanya fokus pada frekuensi kata dalam sebuah dokumen. Setiap kata dalam teks diubah menjadi fitur numerik, di mana setiap fitur mewakili apakah kata tersebut muncul atau tidak, serta seberapa sering kata tersebut muncul dalam dokumen. Dalam konteks sentiment analysis, pendekatan ini membantu model untuk memahami pola kata yang sering diasosiasikan dengan sentimen positif, negatif, atau netral.

Misalnya, jika sebuah dataset berisi ulasan produk atau cuitan dari twitter yang memiliki sentimen tertentu. Bag of Words tidak mempertimbangkan urutan kata, sehingga tidak selalu menangkap makna konteks kalimat, namun metode ini sangat efektif ketika digunakan dalam kombinasi dengan teknik lain seperti TF-ID.

D. Algoritma Machine Learning untuk Sentiment Analysis

Dalam *sentiment analysis*, setelah data teks diproses menjadi representasi numerik seperti Bag of Words, algoritma *machine learning* digunakan untuk memprediksi sentimen. Algoritma ini dikenal sebagai *shallow machine learning* karena model-modelnya relatif sederhana dibandingkan dengan *deep learning* seperti RNN atau Transformer. Beberapa algoritma yang sering digunakan dalam *sentiment analysis* antara lain seperti Logistic Regression, Decision Tree, Support Vector Machine, dan Naive Bayes. Masing-masing algoritma ini dapat disesuaikan dengan berbagai representasi teks dan jenis data, tergantung pada kebutuhan *task sentiment analysis*.

EKSPERIMEN

Data Preprocessing

Data yang diambil dari repositori GitHub IndoNLP-Prosa adalah dataset sentimen dari cuitan Twitter yang telah dilabeli, dengan tiga kelas sentimen: negatif, positif, dan netral. Dataset ini sudah tersedia dalam format .tsv yang terbagi menjadi set pelatihan (train), validasi (valid), dan pengujian (test), dan dibaca menggunakan library pandas. Berikut adalah cuplikan tahapan awal preprocessing untuk setiap dokumen (berformat string):

```
import string
import re
# using sastrawi stemmer for Bahasa
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

def preprocess(text : string):
    # lowering the text
    text = text.lower()

    # stemming using sastrawi ->
    """
    The process is very slow, so I skip this step.
    Assumed that the prosa dataset is already stemmed.
    """
    # factory = StemmerFactory()
    # stemmer = factory.create_stemmer()
    # text = stemmer.stem(text)

    # removing limited punctuation, including !, (), {}, [], ?, dan :
    text = re.sub(r"[,!'\?:\(\)\{\}\[\]]", '', text)

    return text
```

Fungsi di atas secara umum memproses dokumen dengan langkah-langkah sebagai berikut:

- **Mengubah semua huruf menjadi huruf kecil**
- **Melakukan stemming untuk mengubah kata menjadi bentuk dasar:** langkah ini tidak dilakukan karena dokumen mengandung kata dalam jumlah besar, dan penggunaan library Sastrawi memerlukan waktu yang cukup lama. Diasumsikan bahwa kata sudah cukup dasar karena dataset prosa yang *preprocessed*

- **Menghapus tanda baca** (simbol seperti "-" tidak dihapus karena dapat mengubah makna kata, misalnya, "kuda-kuda" dan "kuda" memiliki arti yang berbeda)

Langkah-langkah seperti penghapusan angka dan *stopwords* tidak dilakukan karena pada metode Bag of Words, konteks atau urutan kata tidak diperhatikan, sehingga *stopwords* dan angka bisa saja memiliki makna penting dalam dokumen. Meskipun penghapusan *stopwords* sering membantu mengurangi noise, dalam beberapa kasus, kata-kata tersebut justru berkontribusi pada sentimen atau makna keseluruhan, sehingga keputusan untuk tidak menghapusnya dibuat untuk menjaga informasi penting yang mungkin relevan dalam analisis. Fungsi di atas akan dimanfaatkan untuk pemrosesan tiap dokumen pada dataset.

```
def train_preprocessing(df : pd.DataFrame) -> np.array:
    X_train = df[0].apply(preprocess)
    y_train = df[1]

    pipeline = Pipeline([
        ('vectorizer', CountVectorizer()),
        ('label_encoder', LabelEncoder())
    ])

    # Fit and transform the data
    X_transformed =
pipeline['vectorizer'].fit_transform(X_train).toarray() # Bag of Words
matrix
    X_labels = pipeline['vectorizer'].get_feature_names_out()
    y_transformed = pipeline['label_encoder'].fit_transform(y_train)

    return X_transformed, X_labels, y_transformed, pipeline

# Valid preprocessing
def valid_preprocessing(df: pd.DataFrame, pipeline: Pipeline):
    X_valid = df[0].apply(preprocess)
    Y_valid = df[1]

    # Transform the validation data using the trained vectorizer
    X_valid_transformed =
pipeline['vectorizer'].transform(X_valid).toarray() # Bag of Words matrix
    Y_valid_transformed = pipeline['label_encoder'].transform(Y_valid)

    return X_valid_transformed, Y_valid_transformed
```

Langkah selanjutnya adalah melakukan pemrosesan data validasi dengan menggunakan *pipeline* yang telah dilatih pada data *training*. Fungsi `valid_preprocessing` memproses data validasi dengan cara yang sama seperti data *training*, namun tanpa fitting ulang komponen pipeline. Proses ini mencakup transformasi teks validasi menggunakan `CountVectorizer` (dari library `SkLearn`) yang telah dilatih sebelumnya untuk menghasilkan matriks Bag of Words, serta mengonversi label sentimen validasi menggunakan `LabelEncoder` yang sama. Hasil dari fungsi ini adalah data *training* maupun data validasi yang siap digunakan untuk membangun model klasifikasi.

Modeling (Model Selection)

```
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix, f1_score
# create description of model performance
def model_performance(y_true, y_pred):
    print("Accuracy: ", accuracy_score(y_true, y_pred))
    print("F1 Score: ", f1_score(y_true, y_pred, average='weighted'))
    print("Classification Report:\n", classification_report(y_true,
y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))
```

Akan didefinisikan fungsi untuk mencetak metrik yang dibutuhkan untuk mengevaluasi performa model berupa Akurasi, F1-score, serta *confusion matrix*. Untuk lebih lengkapnya per kelas, `sklearn` juga menyediakan *classification report*.

Berikut merupakan hasil eksperimen pemilihan model yang berhasil dilakukan:

✓

7 d

▶

```
# Create model
model = GaussianNB()
# Fit model
model.fit(X_train, y_train)
```

↔

GaussianNB ⓘ ?

GaussianNB()

✓

1 d

[19]

Kode yang disarankan mungkin tunduk kepada lisensi | Shashidher5716/Decision-Tree-f

```
# Evaluate X_valid
y_pred = model.predict(X_valid)
model_performance(y_valid, y_pred)
```

↔

Accuracy: 0.6515873015873016

F1 Score: 0.6604198809782014

Classification Report:

	precision	recall	f1-score	support
0	0.52	0.72	0.60	394
1	0.46	0.50	0.48	131
2	0.83	0.64	0.72	735
accuracy			0.65	1260
macro avg	0.60	0.62	0.60	1260
weighted avg	0.70	0.65	0.66	1260

Confusion Matrix:

```
[[285  39  70]
 [ 41  66  24]
 [227  38 470]]
```

Algoritma	Naive Bayes
Kelas	GaussianNB
Pustaka	Scikit-Learn
Skor Akurasi	65%
Skor F1	66%
Keterangan	Menggunakan parameter default

✓ SVM(Support Vector Machine) with Linear Kernel

✓
5 d

```
# Linear SVC
from sklearn.svm import LinearSVC
model = LinearSVC()
model.fit(X_train, y_train)
```



LinearSVC

LinearSVC() [Documentation for LinearSVC](#)

✓
0 d

```
[24] # evaluate X_valid
y_pred = model.predict(X_valid)
model_performance(y_valid, y_pred)
```



Accuracy: 0.85
F1 Score: 0.8494792873810929
Classification Report:

	precision	recall	f1-score	support
0	0.78	0.84	0.81	394
1	0.83	0.66	0.74	131
2	0.90	0.89	0.89	735
accuracy			0.85	1260
macro avg	0.83	0.80	0.81	1260
weighted avg	0.85	0.85	0.85	1260

Confusion Matrix:


```
[[331  8 55]
 [ 23 87 21]
 [ 72 10 653]]
```

Algoritma	Support Vector Machine
Kelas	LinearSVC
Pustaka	Scikit-Learn
Skor Akurasi	85%
Skor F1	84.9%
Keterangan	Menggunakan parameter default, kernel defaultnya adalah Linear Kernel

Logistic Classifier

```

✓ 1m [25] # logistic classifier
      from sklearn.linear_model import LogisticRegression
      model = LogisticRegression(max_iter=100)
      model.fit(X_train, y_train)
  
```




LogisticRegression ⓘ ?

LogisticRegression()

```

✓ 0 d [1] # evaluate logistic regressor for classify

      # Evaluate X_valid
      y_pred = model.predict(X_valid)
      model_performance(y_valid, y_pred)
  
```


 Accuracy: 0.8761904761904762
 F1 Score: 0.8757643043996104
 Classification Report:

	precision	recall	f1-score	support
0	0.81	0.88	0.84	394
1	0.83	0.69	0.76	131
2	0.92	0.91	0.91	735
accuracy			0.88	1260
macro avg	0.86	0.83	0.84	1260
weighted avg	0.88	0.88	0.88	1260

Confusion Matrix:
 [[346 6 42]
 [25 91 15]
 [56 12 667]]

Algoritma	Logistic Regression
Kelas	LogisticRegression
Pustaka	Scikit-Learn
Skor Akurasi	87.6%
Skor F1	87.5%
Keterangan	Menggunakan parameter max_iter(iterasi maksimal) = 100

✓ Extreme Gradient Boosting (Tree Model)

```
[27] # install XGBoost
!pip install xgboost
```

Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (2.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from

```
# Create xgboost model
from xgboost import XGBClassifier
model = XGBClassifier()
model.fit(X_train, y_train)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
               colsample_bylevel=None, colsample_bynode=None,
               colsample_bytree=None, device=None, early_stopping_rounds=None,
               enable_categorical=False, eval_metric=None, feature_types=None,
               gamma=None, grow_policy=None, importance_type=None,
               interaction_constraints=None, learning_rate=None, max_bin=None,
               max_cat_threshold=None, max_cat_to_onehot=None,
               max_delta_step=None, max_depth=None, max_leaves=None,
               min_child_weight=None, missing=nan, monotone_constraints=None,
               multi_strategy=None, n_estimators=None, n_jobs=None,
               num_parallel_tree=None, objective='multi:softprob', ...)
```

```
[29] # evaluate
y_pred = model.predict(X_valid)
model_performance(y_valid, y_pred)
```

Accuracy: 0.8523809523809524
F1 Score: 0.8527523288025913
Classification Report:

	precision	recall	f1-score	support
0	0.79	0.82	0.81	394
1	0.70	0.68	0.69	131
2	0.92	0.90	0.91	735
accuracy			0.85	1260
macro avg	0.80	0.80	0.80	1260
weighted avg	0.85	0.85	0.85	1260

Confusion Matrix:
[[325 18 51]
[32 89 10]
[55 20 660]]

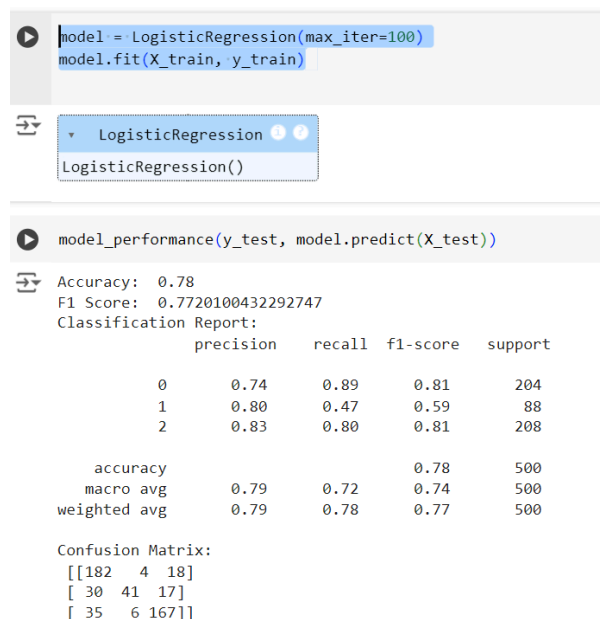
Algoritma	Decision Tree (Extreme Gradient Boosting)
Kelas	XGBClassifier
Pustaka	sgboost
Skor Akurasi	85.2%
Skor F1	85.2%
Keterangan	Menggunakan parameter default

Model Testing

Setelah berhasil menentukan model pengklasifikasi terbaik, dipilih model Logistic Regression untuk task klasifikasi sebagai model untuk menguji data set. Berikut cuplikan kodenya:

```
# concatenate df_train and df_valid
_df_train = pd.concat([df_train, df_valid], axis=0)
# using train preprocessor pipeline
X_train, X_labels, y_train, pipeline = train_preprocessing(df_train)
#preprocessing the test set
X_test, y_test = valid_preprocessing(df_test, pipeline)
model = LogisticRegression(max_iter=100)
model.fit(X_train, y_train)
```

Setelah memilih model Logistic Regression sebagai pengklasifikasi terbaik, langkah pertama adalah menggabungkan dataset *training* (`df_train`) dan dataset validasi (`df_valid`) menjadi satu DataFrame `_df_train`. Kemudian, fungsi `train_preprocessing` digunakan untuk memproses data *training* dan menghasilkan matriks fitur serta label target, sementara data uji (`df_test`) diproses menggunakan fungsi `valid_preprocessing` untuk memastikan konsistensi format. Setelah preprocessing, model Logistic Regression diinisialisasi dengan `max_iter=100` dan dilatih menggunakan data pelatihan yang telah diproses, sehingga model siap untuk melakukan prediksi pada dataset yang belum pernah dilihat sebelumnya pada dataset uji. Berikut merupakan hasil dari Model Testing yang berhasil dilakukan berdasarkan data uji prosa:



```
model = LogisticRegression(max_iter=100)
model.fit(X_train, y_train)
```

LogisticRegression

```
model_performance(y_test, model.predict(X_test))
```

Accuracy: 0.78
F1 Score: 0.7720100432292747
Classification Report:

	precision	recall	f1-score	support
0	0.74	0.89	0.81	204
1	0.80	0.47	0.59	88
2	0.83	0.80	0.81	208
accuracy			0.78	500
macro avg	0.79	0.72	0.74	500
weighted avg	0.79	0.78	0.77	500

Confusion Matrix:

```
[[182  4 18]
 [ 30 41 17]
 [ 35  6 167]]
```

PEMBAHASAN

Eksperimen yang dilakukan menunjukkan bahwa metode Bag of Words dapat memberikan hasil yang layak untuk klasifikasi sentimen, dengan akurasi 78% yang dicapai melalui model Logistic Regression. Meskipun akurasi ini cukup baik untuk aplikasi tertentu, ada potensi untuk peningkatan yang signifikan jika metode lain yang lebih canggih diterapkan. Metode selain Bag of Words, seperti *word embeddings* atau model berbasis konteks seperti *Transformers*, dapat menawarkan pemahaman yang lebih mendalam terhadap makna kata-kata dalam konteks kalimat, yang pada gilirannya dapat meningkatkan akurasi klasifikasi sentimen.

Keterbatasan dalam penerapan teknik seperti stemming disebabkan oleh faktor sumber daya komputasi dan waktu yang mungkin tidak memadai saat menggunakan platform seperti Google Colab. Ini menunjukkan bahwa meskipun metode sederhana dapat memberikan hasil awal yang baik, ada kebutuhan untuk menginvestasikan waktu dan sumber daya dalam pengembangan teknik yang lebih kompleks untuk mencapai hasil yang lebih akurat dan relevan. Dalam praktiknya, penggabungan metode *shallow* dan *deep learning* dapat menciptakan model yang lebih robust dan adaptif terhadap berbagai jenis data dan konteks, yang sangat penting dalam analisis sentimen yang efektif.

Terdapat cuplikan kode dimana beberapa sentiment misklasifikasi seperti pada gambar berikut:

```
# 2nd column is for result of classifier in test dataset
df_test_eval = df_test.copy()
df_test_eval[2] = y_final_pred_label
df_test_eval
```

		0	1	2
0	kemarin gue datang ke tempat makan baru yang a...	negative	negative	
1	kayak nya sih gue tidak akan mau balik lagi ke...	negative	negative	
2	kalau dipikir-pikir , sebenarnya tidak ada yan...	negative	negative	
3	ini pertama kalinya gua ke bank buat ngurusin ...	negative	negative	
4	waktu sampai dengan gue pernah disuruh ibu lat...	negative	negative	
...
495	kata nya tidur yang baik itu minimal enam jam ...	neutral	neutral	
496	indonesia itu ada di benua asia .	neutral	neutral	
497	salah satu kegemaran anak remaja indonesia sek...	neutral	negative	
498	melihat warna hijau bisa bikin mata jadi lebih...	positive	negative	
499	bondan winarno yang suka bilang maknyus sekara...	neutral	negative	

500 rows × 3 columns

DAFTAR REFERENSI

 [Complete Guide to EDA on Text Data | Kaggle](#)

[AI Center ITB](#)

[Vectorization: Bag of Words \(BoW\) | by Vaibhav Jayaswal | Towards Data Science](#)

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

LINK REPOSITORY

<https://github.com/jejejery/if5153-BoW> (Github)

https://colab.research.google.com/drive/10cXdGPDg7687U7PfG9b1Ibl_yYoAvzDE#scrollTo=2KEB648jikZw (Notebook)