LAMPIRAN

Lampiran 1 Pengambilan Data

```
from google_play_scraper import Sort, reviews

result, continuation_token = reviews(
'id.go.kpu.sirekap2024',
lang='id',
country='id',
sort=Sort.MOST_RELEVANT,
count=10000000,
filter_score_with=None
)
```

Scraping Data

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.array(result),columns=['review'])
df = df.join(pd.DataFrame(df.pop('review').tolist()))

df.head()
```

Dataframe Pandas

```
df = df[['userName', 'score', 'at', 'content',
'thumbsUpCount']]
df.sort_values(by='at', ascending=False)
df.head()
```

Filtering dan *Sorting* Kolom

Lampiran 2 Analisis Data

```
import pandas as pd

file_csv = 'scrapped_data.csv'

df = pd.read_csv(file_csv, sep='|')

print(df)
```

Input Data File CSV

```
deskripsi_statistik = df.describe()

mean_rating = df['Rating'].mean()
median_rating = df['Rating'].median()
mode_rating = df['Rating'].mode()[0]

print("Statistik Deskriptif untuk Dataframe berdasarkan
kolom:")
print(deskripsi_statistik)

print("\nStatistik Deskriptif untuk Kolom 'Rating':")
print(f"Mean Rating: {mean_rating}")
print(f"Median Rating: {median_rating}")
print(f"Mode Rating: {mode_rating}")
```

Statistik Desktiptif

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
ratings, counts =
zip(*sorted(dict(df['Rating'].value_counts()).items()))
plt.bar(ratings, counts, alpha=0.7, edgecolor='black',
align='center')
plt.title('Distribusi Rating')
plt.xlabel('Rating')
plt.ylabel('Frekuensi')
plt.xticks(ratings)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()

rating_counts = df['Rating'].value_counts().sort_index()

for rating, count in rating_counts.items():
print(f"Rating {rating}: {count} kali")
```

Distribusi *Rating*

```
import seaborn as sns

df['Panjang Komentar'] = df['Komentar'].apply(lambda x:
len(x.split()))
```

Relasi *Rating* Dengan Panjang Komentar

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Rating', y='Panjang Komentar', data=df,
color='blue', alpha=0.7)
plt.title('Hubungan antara Rating dan Panjang Komentar')
plt.xlabel('Rating')
plt.ylabel('Panjang Komentar')
plt.show()
```

Scatter Plot

```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Rating', y='Panjang Komentar', data=df,
color='blue')
plt.title('Distribusi Panjang Komentar untuk Setiap Rating')
plt.xlabel('Rating')
plt.ylabel('Panjang Komentar')
plt.show()
```

Box Plot

```
info_shortest_comments =
df.loc[df.groupby('Rating')['Panjang
Komentar'].idxmin()][['Rating', 'Panjang Komentar',
'Komentar']]
info_longest_comments = df.loc[df.groupby('Rating')['Panjang
Komentar'].idxmax()][['Rating', 'Panjang Komentar',
'Komentar']]

for index, row in info_shortest_comments.iterrows():
print(f"Rating {row['Rating']}: Komentar Paling Pendek -
Index: {index}, Panjang: {row['Panjang Komentar']},
Komentar: {row['Komentar']}")

print("\n")

for index, row in info_longest_comments.iterrows():
```

```
print(f"Rating {row['Rating']}: Komentar Paling Panjang -
   Index: {index}, Panjang: {row['Panjang Komentar']},
   Komentar: {row['Komentar']}")
```

Kalimat Terpanjang dan Terpendek Setiap *Rating*

```
from wordcloud import WordCloud

def generate_wordcloud(text, title):
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(text)

plt.figure(figsize=(15, 10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title(title)
plt.show()

sorted_ratings = df['Rating'].unique()[::-1]

for rating in sorted_ratings:
comments = ' '.join(df[df['Rating'] == rating]['Komentar'])
generate_wordcloud(comments, f'Word Cloud Rating {rating}')
```

Word Cloud Setiap *Rating*

Lampiran 3 Pra Proses

```
import pandas as pd

file_csv = 'scrapped_data.csv'

df = pd.read_csv(file_csv, sep='|')

print(df)
```

Input Data CSV

```
digit_count = len(str(len(df)))

df.insert(0, 'ID', df.index.map(lambda x: 'd' +
str(x+1).zfill(digit_count)))

print(df)
```

Memberikan *ID* Disetiap Data

```
import time

def lowercase(text):
return text.lower()

start_time = time.time()
df['komentar_lowercase'] = df['Komentar'].apply(lowercase)
end_time = time.time()
time_taken = end_time - start_time

print(df[['Komentar', 'komentar_lowercase']])
print("Waktu proses:", time_taken, "detik")
```

Lower Case

```
import re

```

```
def remove_unnecessary_char(text):
text =
re.sub('((www\.[^\s]+)|(https?://[^\s]+)|(http?://[^\s]+))','
',text)
text = re.sub('\n',' ',text)
text = re.sub('\r',' ',text)
text = re.sub(r'\\x..',' ',text)
text = re.sub('  +', ' ', text)
text = re.sub('[^0-9a-zA-Z]+', ' ', text)
return text

start_time = time.time()
df['komentar_remove_char'] =
df['komentar_lowercase'].apply(remove_unnecessary_char)
end_time = time.time()
time_taken = end_time - start_time


print(df[['komentar_lowercase', 'komentar_remove_char']])
print("Waktu proses:", time_taken, "detik")
```

Remove Unecessary Character

```
alay_dict = pd.read_csv('kamus_tidak_baku.csv', sep=";",
encoding='latin-1', header=None)
alay_dict = alay_dict.rename(columns={0: 'original',
                                      1. 1: 'replacement'})

alay_dict_map = dict(zip(alay_dict['original'],
alay_dict['replacement']))
```

Input Kamus Tidak Baku

```
df['komentar_remove_char'] =
df['komentar_remove_char'].astype(str)
def normalize_alay(text):
return ' '.join([str(alay_dict_map.get(word, word)) for word
in text.split (' ')])

start_time = time.time()
df['komentar_spellchecker'] =
df['komentar_remove_char'].apply (normalize_alay)
end_time = time.time()
time_taken = end_time - start_time

print(df[['komentar_remove_char', 'komentar_spellchecker']])
print("Waktu proses:", time_taken, "detik")
```

Proses *Spellchecker*

```
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
factory = StemmerFactory()
stemmer = factory.create_stemmer()

def stemming(text):
return stemmer.stem(text)

start_time = time.time()

df['komentar_stemming'] =
df['komentar_spellchecker'].apply(stemming)

end_time = time.time()
time_taken = end_time - start_time
```

```
13
14 print(df[['komentar_spellchecker', 'komentar_stemming']])
15 print("Waktu proses:", time_taken, "detik")
```

Stemming Sastrawi

```
1  import numpy as np
2
3  df['komentar_stemming'] = df['komentar_stemming'].replace('',
   np.nan)
4
5  baris_nan = df[df['komentar_stemming'].isnull()]
6  jumlah_nan = df['komentar_stemming'].isnull().sum()
7  print(f"Jumlah NaN pada kolom 'komentar_stemming':
   {jumlah_nan}")
8  print(baris_nan)
9  #print("ID dengan 'komentar_stemming' NaN:",
   baris_nan['ID'].tolist())
```

Cek Data Kosong

```
1  df_cleaned = df.dropna(subset=['komentar_stemming'])
2  df_cleaned
```

Hapus Data Kosong

Lampiran 4 *WordNet*

```
1  import pandas as pd
2
3  file_csv = 'output_praproses.csv'
4  3
5  df = pd.read_csv(file_csv, sep='|')
6
7  print(df)
```

Input Data

```
1  from deep_translator import GoogleTranslator
2  import time
3
4  def translate_komentar(text):
5  text_translated = GoogleTranslator(source='id',
   target='en').translate(text)
6  return text_translated
7
8  from tqdm import tqdm
9
10 tqdm.pandas()
11
12 start_time = time.time()
13 df['komentar_stemming'] = df['komentar_stemming'].fillna('')
14
15 df['Translated'] =
   df['komentar_stemming'].progress_apply(translate_komentar)
16 end_time = time.time()
17 time_taken = end_time - start_time
18
19 print(df[['komentar_stemming', 'Translated']])
20 print("Waktu proses:", time_taken, "detik")
```

Fungsi *Translated*

```
1  from textblob import TextBlob
2
3  def scoring sentiment(text):
```

```
    analysis = TextBlob(text)
    sentiment_score = analysis.sentiment.polarity
    # print("Sentiment polarity:", sentiment_score)
    score_wordnet = 0

    if sentiment_score > 0.5:
            score_wordnet = 5
    elif 0.2 <= sentiment_score <= 0.5:
            score_wordnet = 4
    elif -0.2 < sentiment_score < 0.2:
            score_wordnet = 3
    elif -0.5 <= sentiment_score <= -0.2:
           score_wordnet = 2
    else:
           score_wordnet = 1

    return score_wordnet, sentiment_score
```

Hitung *Score*

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, f1_score
from matplotlib.ticker import FixedLocator

cm = confusion_matrix(df['Rating'], df['Score_Wordnet'])

macro_f1 = f1_score(df['Rating'], df['Score_Wordnet'],
average='macro')
print(f"Macro F1-Score: {macro_f1}")

fig, ax = plt.subplots(figsize=(10, 8), dpi=100)
cax = ax.matshow(cm, cmap=plt.cm.Blues)
fig.colorbar(cax)

tick_marks = np.arange(len(np.unique(df['Rating'])))
ax.set_xticks(tick_marks)
ax.set_yticks(tick_marks)
ax.set_xticklabels(list(range(1, 6)))
ax.set_yticklabels(list(range(1, 6)))

plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix WordNet')

for i in range(len(cm)):
for j in range(len(cm[i])):
ax.text(j, i, str(cm[i][j]), va='center', ha='center',
color='black')

plt.show()
```

Confusion Matrix F1-Score

Lampiran 5 Klasifikasi

```
import pandas as pd

file_csv = 'output_praproses.csv'

df = pd.read_csv(file_csv, sep='|')
```

```
6
7 print(df)
```

Input Data

```
1 X = df['komentar_stemming']
2 y = df['Rating']
```

Menentukan *Input* dan Kelas

```
1 from sklearn.feature_extraction.text import TfidfVectorizer
2
3 X.fillna('', inplace=True)
4 vectorizer = TfidfVectorizer()
5 features = vectorizer.fit_transform(X)
```

Ekstraksi Fitur TF-IDF

```
1  from sklearn.model_selection import KFold
2  from sklearn.model_selection import cross_val_predict,
   StratifiedKFold
3  from sklearn.metrics import accuracy_score, precision_score,
   recall_score, f1_score, confusion_matrix
4  import numpy as np
5  import matplotlib.pyplot as plt
6
7  def cross_validation(model, _X, _y, _cv):
8  kf = KFold(n_splits=_cv, shuffle=True)
9  fold_count = 1
10 results = {}
11 fold_predictions = []
12 overall_cm = np.zeros((len(np.unique(_y)),
   len(np.unique(_y))), dtype=int)
13
14 for train_index, test_index in kf.split(_X):
15 X_train, X_test = _X[train_index], _X[test_index]
16 y_train, y_test = _y[train_index], _y[test_index]
17
18 model.fit(X_train, y_train)
19 y_pred = model.predict(X_test)
20 fold_predictions.append(y_pred)
21 cm = confusion_matrix(y_test, y_pred)
22
23 results[f"Fold {fold_count}"] = {
24 "Confusion Matrix": cm,
25 "Precision": precision_score(y_test, y_pred,
   average='macro'),
26 "Recall": recall_score(y_test, y_pred, average='macro'),
27 "F1 Score": f1_score(y_test, y_pred, average='macro')
28 }
29 overall_cm += cm
30 fold_count += 1
31
32 for key, value in results.items():
33 print(key + ":")
34 print("Confusion Matrix:")
35 print(value["Confusion Matrix"])
36 print("Precision:", value["Precision"])
37 print("Recall:", value["Recall"])
38 print("F1 Score:", value["F1 Score"])
39 print()
40
41 fig, ax = plt.subplots(figsize=(10, 8), dpi=100)
```

```
    im = ax.imshow(value["Confusion Matrix"],
    interpolation='nearest', cmap=plt.cm.Blues)
    ax.figure.colorbar(im, ax=ax)

    class_names = np.unique(_y)
    ax.set(xticks=np.arange(len(class_names)),
    yticks=np.arange(len(class_names)),
    xticklabels=class_names, yticklabels=class_names,
    title=f'Confusion Matrix - {key}',
    ylabel='True label',
    xlabel='Predicted label')

    thresh = value["Confusion Matrix"].max() / 2.
    for i in range(value["Confusion Matrix"].shape[0]):
    for j in range(value["Confusion Matrix"].shape[1]):
    ax.text(j, i, format(value["Confusion Matrix"][i, j], 'd'),
    ha="center", va="center",
    color="white" if value["Confusion Matrix"][i, j] > thresh
    else "black")

    plt.show()

    print("Overall Confusion Matrix:")
    print(overall_cm)
    print()

    overall_precision = np.diag(overall_cm).sum() /
    overall_cm.sum(axis=0).sum()
    overall_recall = np.diag(overall_cm).sum() /
    overall_cm.sum(axis=1).sum()
    overall_f1_score = 2 * (overall_precision * overall_recall) /
    (overall_precision + overall_recall)

    #print("Overall Evaluation:")
    #print("Precision:", overall_precision)
    #print("Recall:", overall_recall)
    #print("F1 Score:", overall_f1_score)

    y_pred = cross_val_predict(model, _X, _y, cv=_cv)

    cm_overall = confusion_matrix(_y, y_pred)
    #print("Confusion Matrix:")
    #print(cm_overall)

    fig, ax = plt.subplots(figsize=(10, 8), dpi=100)
    cax = ax.matshow(cm_overall, cmap=plt.cm.Blues)
    fig.colorbar(cax)

    tick_marks = np.arange(len(np.unique(_y)))
    ax.set_xticks(tick_marks)
    ax.set_yticks(tick_marks)
    ax.set_xticklabels(list(range(1, 6)))
    ax.set_yticklabels(list(range(1, 6)))

    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.title('Confusion Matrix Overall')

    thresh = overall_cm.max()/2.

    for i in range(cm_overall.shape[0]):
```

```
for j in range(cm_overall.shape[1]):
ax.text(j, i, format(overall_cm[i][j], 'd'), va='center',
   ha='center', color='black' if overall_cm[i,j] > thresh else
   "black")

      plt.show()

      precision_macro = precision_score(y, y_pred,
   average='macro')
      print(f"\nPrecision Macro: {precision_macro}")

      recall_macro = recall_score(y, y_pred, average='macro')
      print(f"Recall Macro: {recall_macro}")

      f1_score_macro = f1_score(y, y_pred, average='macro')
      print(f"F1-Score Macro: {f1_score_macro}")


      return results, fold_predictions
```

Cross Validation 10 Fold

```

kfold = KFold(n_splits=10, shuffle=False)
for fold, (train_index, test_index) in
enumerate(kfold.split(X, y), 1):
if fold == 10:

X_train, X_test = X[train_index], X[test_index]
y_train, y_test = y[train_index], y[test_index]

print(f'Fold {fold}:')
print(f'  - Train data: {len(X_train)} samples')
print(f'  - Test data: {len(X_test)} samples')
print(f'  - Train Index: {train_index}')
print(f'  - Test Index: {test_index}')
```

Menampilkan Isi Data Setiap Pembagian *Fold*

```
from sklearn.neighbors import KNeighborsClassifier
# from sklearn.neighbors import NearestNeighbors
knn model = KNeighborsClassifier()
```

Klasifikasi K-NN

```
import time

start_time = time.time()
cv_results, fold_predictions = cross_validation(knn_model,
features, y, _cv=10)

end_time = time.time()
time_taken = end_time - start_time
print("Waktu proses:", time_taken, "detik")
```

Menjalankansss *Cross Validation*

Lampiran 6 Kartu Kendali Bimbingan

KARTU KENDALI BIMBINGAN LAPORAN KARYA ILMIAH

Nama	:	Emyzar Haflida Tanjung
NIM	:	2011102441240
Nama Dosen Pembimbing	:	Naufal Azmi Verdikha, S.Kom., M.Eng.
Judul Penelitian	:	Perbandingan Analisis *Wordnet* dan *K-Nearest Neighbor* Pada Ulasan Aplikasi Sirekap 2024.

No	Tanggal	Uraian Pembimbingan	Paraf Dosen
1	27 Januari 2024	penentuan algoritma yang akan digunakan dan pembuatan judul besar.	
2	29 januari 2024	arahan proses codingan awal dan pembuatan kamus tidak baku	
3	31 Januari 2024	1. koreksi pembuatan kamus tidak baku 2. penjelasan codingan dan mencari alasan mengapa harus melakukan klasifikasi sentimen.	
4	1 Febuari 2024	arahan point penting yang harus ditulis diproposal.	
5	18 Febuari 2024	1. koreksi penulisan latar belakang 2. pembuatan tahap alur penelitian (metodologi)	
6	19 Febuari 2024	koreksi bab 1 dan bab 2 proposal	
7	21 Februari 2024	bimbingan koreksi revisi proposal	
8	11 Maret 2024	arahan membuat kerangka codingan	
9	25 Juni 2024	koreksi revisi proposal	
10	13 Juni 2024	revisi bab 3	
11	14 Juni 2024	1. koreksi penyesuaian penulisan bab 3 2. koreksi revisian bab 3	
12	22 Juni 2024	koreksi revisian bab 3	

Dosen Pembimbing

(.....)
Naufal Azmi Verdikha, S.Kom., M.Eng
NIDN. 1114048801

Mengetahui
Ketua Program Studi

(.....)
NIDN. 1118019203

Lampiran 7 Hasil Uji Turnitin

SKRIPSI EMYZAR HAFILDA TANJUNG

by Teknik Informatika Universitas Muhammadiyah Kalimantan Timur

Submission date: 25-Jul-2024 02:23PM (UTC+0800)
Submission ID: 2422166292
File name: SKRIPSI_EMYZAR_HAFILDA_TANJUNG.docx (1.95M)
Word count: 5788
Character count: 36142

SKRIPSI EMYZAR HAFILDA TANJUNG

ORIGINALITY REPORT

18%	17%	7%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	repository.ub.ac.id Internet Source	2%
2	e-journal.hamzanwadi.ac.id Internet Source	1%
3	media.neliti.com Internet Source	1%
4	core.ac.uk Internet Source	1%
5	docplayer.info Internet Source	1%
6	dspace.umkt.ac.id Internet Source	1%
7	scholar.unand.ac.id Internet Source	1%
8	digilib.polban.ac.id Internet Source	1%
9	repo.iain-tulungagung.ac.id Internet Source	<1%

Lampiran 8 Surat Ijin Penelitian

UMKT
Program Studi
Teknik Informatika
Fakultas Sains dan Teknologi
UNIVERSITAS MUHAMMADIYAH Kalimantan Timur
Berkarakter | Berwawasan | Berkemajuan
Telp. 0541-748511 Fax. 0541-766832
Website http://informatika.umkt.ac.id
email: informatika@umkt.ac.id

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Nomor : 055-001/KET/FST.1/A/2024
Lampiran : -
Perihal : **Keterangan Pengambilan Data Sekunder**

Assalamu'alaikum Warrahmatullahi Wabarrakatuh

Puji Syukur kepada Allah Subhanahu wa ta'ala yang senantiasa melimpahkan Rahmat-Nya kepada kita sekalian. Amin.

Dengan surat ini, kami menerangkan bahwa mahasiswa berikut:

No	Nama	NIM
1	Emyzar Haflida Tanjung	2011102441240
2	Lilis Sagita	2011102441198
3	Sri Ramadani	2011102441177
4	Muhammad Fariz Ijlal Rafi	2011102441124
5	Nurlita	2011102441070

Melakukan penelitian dengan pengambilan data sekunder di Google Playstore data yang diambil yaitu data ulasan aplikasi "Sirekap" dari rating 1-5.

Demikian hal ini disampaikan, atas kerjasamanya kami ucapkan terima kasih.

Wassalamu'alaikum Warrahmatullahi Wabarrakatuh

Samarinda, 18 Dzulhijjah 1445 H
25 Juni 2024 M

Ketua Program Studi S1 Teknik Informatika

Arbansyah, S.Kom., M.TI
NIDN. 1118019203

RIWAYAT HIDUP

Penulis bernama lengkap Emyzar Haflida Tanjung dilahirkan di Kisaran 23 Agustus 2001, dan merupakan anak ketiga dari 4 bersaudara dari pasangan Ghazali Tanjung,.SE dan Paridah. Mengawali pendidikan formal di Pendidikan Sekolah Dasar di MIN Tanah Paser (2007-2013) dan melanjutkan Pendidikan Sekolah Menengah Pertama di Mts Bina Islam Tanah Paser (2013-2016). Penulis melanjutkan jenjang pendidikan formal Sekolah Menengah Atas di SMKN 1 Tanah Paser (2016-2019). Penulis masuk di Fakultas Sains dan Teknologi Universitas Muhammadiyah Kalimantan Timur pada tahun 2020.

Untuk menyelesaikan studi di Fakultas Sains dan Teknologi Jurusan Teknik Informatika UMKT, penulis melakukan penelitian dengan judul "**Perbandingan Analisis *Wordnet* Dan *K-Nearest Neighbor* Pada Ulasan Aplikasi Sirekap 2024"** sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer.