LAPORAN FINAL PROJECT
TEXT EMOTION DETECTION

Disusun Guna Memenuhi Tugas

Mata Kuliah : Kecerdasan Buatan

Dosen Pengampu :

Disusun Oleh:

Meissy Irania Putri	2210631170030
Dinda Ismi Hanifah	2210631170116

PROGRAM STUDI INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS SINGAPERBANGSA KARAWANG

2024

1. Latar Belakang

Dalam era digital saat ini, komunikasi teks telah menjadi salah satu bentuk utama interaksi antar individu. Media sosial, email, dan platform pesan instan memungkinkan kita untuk berbagi perasaan, pikiran, dan pengalaman dalam bentuk teks. Namun, memahami emosi yang terkandung dalam teks tersebut tidak selalu mudah bagi manusia, terutama ketika volume data yang harus dianalisis sangat besar. Oleh karena itu, deteksi emosi dari teks menjadi topik penelitian yang penting dalam bidang pemrosesan bahasa alami (Natural Language Processing - NLP) dan kecerdasan buatan (Artificial Intelligence - AI).

2. Tujuan

Tujuan dari proyek ini adalah untuk mengembangkan sistem deteksi emosi berbasis teks menggunakan algoritma pembelajaran mesin. Sistem ini diharapkan mampu mengklasifikasikan teks ke dalam kategori emosi tertentu, seperti bahagia, sedih, marah, dan sebagainya. Dengan adanya sistem ini, berbagai aplikasi dapat dikembangkan, termasuk analisis sentimen di media sosial, layanan pelanggan otomatis, dan pemantauan kesejahteraan psikologis.

3. Metodologi

Proyek ini menggunakan metode pembelajaran mesin untuk membangun model deteksi emosi teks. Beberapa langkah utama dalam metodologi ini meliputi:

1. **Pengumpulan dan Pra-pemrosesan Data**: Mengumpulkan dataset teks dengan label emosi yang jelas dan membersihkannya dari kata-kata yang tidak relevan menggunakan teknik pra-pemrosesan seperti penghapusan stopwords.
2. **Transformasi Teks**: Menggunakan teknik TF-IDF (Term Frequency-Inverse Document Frequency) untuk mengonversi teks menjadi representasi numerik yang dapat dipahami oleh algoritma pembelajaran mesin.
3. **Pembelajaran Model**: Melatih model klasifikasi Naive Bayes menggunakan data yang telah diproses, serta melakukan pencarian parameter terbaik menggunakan GridSearchCV.
4. **Evaluasi Model**: Mengevaluasi performa model dengan metrik seperti akurasi dan laporan klasifikasi untuk memastikan keandalan prediksi emosi.
5. **Implementasi dan Pengujian**: Menyimpan model yang telah dilatih dan mengimplementasikan fungsi untuk memprediksi emosi dari teks baru.

4. Implementasi

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.pipeline import Pipeline
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import GridSearchCV
import joblib

nltk.download('stopwords')

def preprocess_text(text):
    stop_words = set(stopwords.words('english'))
    words = text.split()
    words = [word for word in words if word.lower() not in stop_words]
    return ' '.join(words)
```

```
data = {
    'text': [
        'I am so happy today!',
        'I feel sad and depressed.',
        'What a wonderful experience!',
        'I am so angry right now.',
        'Feeling excited and joyful!',
        'I am very happy with the results.'
        'I am feeling very sad and lonely.',
        'This is the best day ever!',
        'I am frustrated and upset.',
        'I am overjoyed with happiness.'
    ],
    'emotion': ['happy', 'sad', 'happy', 'angry', 'happy', 'happy', 'sad', 'happy', 'angry', 'happy']
}

df = pd.DataFrame(data)
df['clean_text'] = df['text'].apply(preprocess_text)
X_train, X_test, y_train, y_test = train_test_split(df['clean_text'], df['emotion'], test_size=0.2, random_state=42)

pipeline = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('clf', MultinomialNB())
])

parameters = {
    'tfidf__max_df': [0.8, 0.9, 1.0],
    'tfidf__ngram_range': [(1,1), (1,2)],
    'clf__alpha': [0.1, 0.5, 1.0]
}

grid_search = GridSearchCV(pipeline, parameters, cv=5, n_jobs=-1, verbose=1)
grid_search.fit(X_train, y_train)

best_pipeline = grid_search.best_estimator_
best_pipeline.fit(X_train, y_train)
y_pred = best_pipeline.predict(X_test)

print(f'Akurasi: {accuracy_score(y_test, y_pred)}')
print(classification_report(y_test, y_pred))

joblib.dump(best_pipeline, 'emotion_detection_model.pkl')
```

```
def predict_emotion(text):
    model = joblib.load('emotion_detection_model.pkl')
    clean_text = preprocess_text(text)
    prediction = model.predict([clean_text])
    return prediction[0]
```

Contoh penggunaan

```
new_text = "I am so happy today!"
predicted_emotion = predict_emotion(new_text)
print(f'Teks: "{new_text}"')
print(f'Prediksi Emosi: {predicted_emotion}')
```

Output

```
Teks: "aku senang sekali denganmu"
Prediksi Emosi: happy
```

Penjelasan Program

1. **Import Library dan Download Stopwords**

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.pipeline import Pipeline
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import GridSearchCV
import joblib


nltk.download('stopwords')
```

- **Library Import**: Mengimpor pustaka yang dibutuhkan untuk pengolahan data, pembelajaran mesin, dan evaluasi model.
- **Download Stopwords:** Mengunduh daftar kata-kata umum (stopwords) yang akan dihapus dari teks.

2. **Fungsi untuk Membersihkan Teks:**

```
def preprocess_text(text):
    stop_words = set(stopwords.words('english'))
    words = text.split()
    words = [word for word in words if word.lower() not in stop_words]
    return ' '.join(words)
```

- **Preprocess Text**: Menghapus stopwords dari teks untuk fokus pada kata-kata yang lebih bermakna.

3. **Contoh Dataset Sederhan**:

```
data = {
    'text': [
        'I am so happy today!',
        'I feel sad and depressed.',
        'What a wonderful experience!',
        'I am so angry right now.',
        'Feeling excited and joyful!',
        'I am very happy with the results.',
        'I am feeling very sad and lonely.',
        'This is the best day ever!',
        'I am frustrated and upset.',
        'I am overjoyed with happiness.'
    ],
    'emotion': ['happy', 'sad', 'happy', 'angry', 'happy', 'happy', 'sad', 'happy', 'angry', 'happy']
}
df = pd.DataFrame(data)
```

- **Dataset**: Contoh data yang terdiri dari kalimat-kalimat dan label emosinya.

4. **Pra-pemrosesan Teks**:

```
df['clean_text'] = df['text'].apply(preprocess_text)
```

- **Clean Text**: Menerapkan fungsi `preprocess_text` untuk membersihkan teks.

5**. Membagi Data Menjadi Data Pelatihan dan Pengujian:**

```
X_train, X_test, y_train, y_test = train_test_split(df['clean_text'], df['emotion'], test_size=0.2, random_state=42)
```

- **Train-Test Split**: Membagi dataset menjadi data pelatihan (80%) dan data pengujian (20%).

6. **Membangun Pipeline:**

```
pipeline = Pipeline([
    ('tfidf', TfidfVectorizer()),
    ('clf', MultinomialNB())
])
```

- **Pipeline**: Menggabungkan dua langkah utama:
- **TfidfVectorizer`**: Mengonversi teks menjadi representasi vektor dengan menggunakan teknik TF-IDF.
- MultinomialNB`: Model Naive Bayes untuk klasifikasi teks.

7. **Parameter Tuning dengan GridSearchCV**:

```
parameters = {
    'tfidf__max_df': [0.8, 0.9, 1.0],
    'tfidf__ngram_range': [(1,1), (1,2)],
    'clf__alpha': [0.1, 0.5, 1.0]
}

grid_search = GridSearchCV(pipeline, parameters, cv=5, n_jobs=-1, verbose=1)
grid_search.fit(X_train, y_train)
```

- **GridSearchCV**: Teknik untuk mencari kombinasi parameter terbaik untuk pipeline dengan menggunakan cross-validation (cv=5).

8. **Melatih Model Terbaik**:

```
best_pipeline = grid_search.best_estimator_
best_pipeline.fit(X_train, y_train)
```

- **Best Estimator**: Menggunakan model terbaik yang ditemukan oleh GridSearchCV untuk melatih ulang pada data pelatihan.

9. **Memprediksi dan Evaluasi**:

```
y_pred = best_pipeline.predict(X_test)
print(f'Akurasi: {accuracy_score(y_test, y_pred)}')
print(classification_report(y_test, y_pred))
```

- **Predict**: Memprediksi label emosi untuk data pengujian.
- **Evaluation**: Mengevaluasi model menggunakan metrik akurasi dan laporan klasifikasi.

10. **Menyimpan Model**:

```
joblib.dump(best_pipeline, 'emotion_detection_model.pkl')
```

- Model Persistence: Menyimpan model yang dilatih ke dalam file untuk penggunaan di masa mendatang.

11. **Penggunaan Model:**

```
def predict_emotion(text):
    model = joblib.load('emotion_detection_model.pkl')
    clean_text = preprocess_text(text)
    prediction = model.predict([clean_text])
    return prediction[0]
```

- **Predict Emotion**: Fungsi untuk memprediksi emosi dari teks baru menggunakan model yang telah disimpan.

Link GitHub: https://github.com/MeissyIraniaPutri/Kecerdasan-Buatan