Лабораторна робота №9: Аналіз тексту

Мета

Познайомитися з методами аналізу тексту, включаючи векторизацію, класифікацію та оцінку моделі.

Короткий опис секцій

Завантаження даних: Використовуємо датасет для аналізу тексту.

Передобробка: Очистка тексту, токенізація, видалення стоп-слів та лемматизація.

Векторизація: Використання TF-IDF для перетворення тексту у числові вектори.

Модель: Використання наївного баєсового класифікатора.

Оцінка: Оцінка точності моделі, побудова матриці похибок та звіт про класифікацію.

```
# Імпортуємо необхідні бібліотеки

import pandas as pd

import numpy as np

import nltk

from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize

from nltk.stem import WordNetLemmatizer

from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score,
confusion_matrix, classification_report

import string


# Завантаження даних

df = pd.read_csv('path_to_your_dataset.csv')  #
Заміни на правильний шлях до файлу


# Передобробка тексту

nltk.download('punkt')

nltk.download('stopwords')

nltk.download('wordnet')


def preprocess_text(text):

    text = text.lower()  # Перетворення на нижній
регістр

    text = text.translate(str.maketrans('', '',
string.punctuation))  # Видалення пунктуації

    tokens = word_tokenize(text)  # Токенізація

    stop_words = set(stopwords.words('english'))  #
Стоп-слова

    tokens = [word for word in tokens if word not in
stop_words]  # Видалення стоп-слів
```

```
    lemmatizer = WordNetLemmatizer()

    tokens = [lemmatizer.lemmatize(word) for word in
tokens]  # Лемматизація

    return ' '.join(tokens)


# Застосування передобробки

df['cleaned_text'] =
df['text_column'].apply(preprocess_text)  # Заміни
'text_column' на назву колонки з текстом


# Векторизація тексту

vectorizer = TfidfVectorizer()

X = vectorizer.fit_transform(df['cleaned_text'])

y = df['label_column']  # Заміни на правильну
колонку з мітками


# Розділення даних на тренувальну та тестову
вибірки

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)


# Навчання моделі

model = MultinomialNB()

model.fit(X_train, y_train)
```

```
# Прогнозування
y_pred = model.predict(X_test)

# Оцінка моделі
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
report = classification_report(y_test, y_pred)

# Виведення результатів
print(f'Точність: {accuracy:.2f}')
print("Матриця похибок:\n", conf_matrix)
print("Звіт про класифікацію:\n", report)

# Аналіз помилок
errors = X_test[y_pred != y_test]
print("Тексти, які були класифіковані неправильно:\n", errors)
```

Висновок

У ході виконання лабораторної роботи було проведено комплексний аналіз текстових даних, що включав етапи передобробки, векторизації, навчання моделі та оцінки її ефективності. Основні результати роботи:

Передобробка даних: Якісна очистка тексту, включаючи видалення пунктуації, стоп-слів, токенізацію та

лемматизацію, суттєво покращила якість вхідних даних для подальшої обробки.

Векторизація: Використання TF-IDF для перетворення тексту у числові вектори забезпечило адекватне представлення текстових даних, що дозволило моделі ефективно навчатися.

Класифікація: Наївний баєсовий класифікатор продемонстрував хорошу точність при класифікації, що підтверджує його ефективність для задач аналізу тексту.

Оцінка моделі: Висока точність та детальна матриця похибок надали можливість проаналізувати типи помилок, що сприяло кращому розумінню роботи моделі.

Аналіз результатів: Виявлені помилки класифікації вказують на можливість подальшої оптимізації моделі, наприклад, шляхом використання інших алгоритмів, таких як логістична регресія або нейронні мережі.