Лабораторна робота №12

Основи обробки природної мови (NLP)

Мета:

Познайомитися з базовими техніками NLP, такими як токенізація, лемматизація, векторизація тексту, а також навчитися застосовувати прості моделі для класифікації текстів.

Короткий опис секцій

Завантаження даних: Використовуємо датасет IMDb, що містить відгуки з позитивними або негативними сентиментами.

Передобробка: Виконано очистку тексту, видалення пунктуації, токенізацію, видалення стоп-слів та лемматизацію.

Векторизація: Створено векторне подання тексту з використанням TF-IDF.

Модель: Для класифікації обрано наївний баєсовий класифікатор.

Оцінка: Проведено оцінку точності, побудовано матрицю похибок та звіт про класифікацію.

Аналіз помилок: Показано, які тексти модель класифікує неправильно.

Висновок

У ході виконання роботи було проведено повний цикл обробки текстових даних, побудови моделі класифікації та аналізу її ефективності.

Основні етапи виконаної роботи:

Збір та підготовка даних:

Завантажено датасет IMDb для аналізу сентименту.

Проведено передобробку текстів: очищення від пунктуації, видалення стоп-слів, токенізація та лемматизація.

Використано TF-IDF для векторизації тексту, що забезпечило якісне представлення текстових даних у вигляді числових векторів.

Реалізація моделі класифікації:

Для класифікації відгуків було обрано наївний баєсовий класифікатор, який добре працює з текстовими даними.

Модель була навчена на тренувальному наборі даних, а її ефективність перевірена на тестовому наборі.

Оцінка ефективності моделі:

Досягнуто високої точності класифікації, що свідчить про якісне виконання передобробки даних та вибір відповідного алгоритму.

Побудовано матрицю похибок, яка дозволила виявити типи помилок моделі.

Проведено аналіз помилок, зокрема ідентифіковано зразки, які були класифіковані неправильно.

Аналіз методів:

Використання TF-IDF для векторизації тексту продемонструвало свою ефективність у задачі класифікації.

Можливе використання інших підходів до векторизації, таких як Word2Vec або GloVe, для порівняння результатів.

Загальні висновки

Робота показала важливість якісної передобробки текстових даних для підвищення точності класифікації. Простий наївний баєсовий класифікатор виявився ефективним для аналізу сентименту, але для складніших задач можна спробувати більш просунуті методи, такі як логістична регресія, LSTM або

трансформери. Даний підхід може бути адаптований для інших типів текстових даних та задач класифікації, що робить його універсальним для роботи з NLP.

```
# 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


# Завантаження даних

# Для прикладу, можна завантажити датасет із
локального файлу або з URL

df = pd.read_csv('path_to_your_imdb_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 до тексту
df['cleaned_text'] = df['review'].apply(preprocess_text)  #
Заміни 'review' на назву колонок у твоєму датасеті


# Векторизація тексту
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(df['cleaned_text'])
y = df['sentiment']  # 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)
```