Лабораторна робота 12 ІАД
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.

```python
# Імпорт необхідних бібліотек
import pandas as pd
import numpy as np
import nltk
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer, 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 seaborn as sns
import matplotlib.pyplot as plt

# Завантаження необхідних ресурсів NLTK
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

# 1. Завантаження датасету
# Ми використовуємо датасет IMDb з відгуками та аналізом сентименту
url = "https://raw.githubusercontent.com/datasets/imdb-reviews/master/data/reviews.csv"
data = pd.read_csv(url)
data = data[['review', 'sentiment']]
data = data.sample(frac=1, random_state=42).reset_index(drop=True)  # Перемішування даних
print(data.head())
```

```python
# 2. Передобробка тексту
# Функція очищення тексту
def preprocess_text(text):
    text = re.sub(r'[^\w\s]', '', text)  # Видалення пунктуації
    text = text.lower()  # Переведення тексту до нижнього регістру
    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 функції до датасету
data['cleaned_review'] = data['review'].apply(preprocess_text)
print(data['cleaned_review'].head())

# 3. Векторизація тексту
# Використання TF-IDF
vectorizer = TfidfVectorizer(max_features=5000)  # Обмеження на кількість слів
X = vectorizer.fit_transform(data['cleaned_review'])
y = data['sentiment'].map({'positive': 1, 'negative': 0})  # Кодування міток

# 4. Розділення даних на тренувальні та тестові
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
print("Розміри тренувального та тестового наборів:", X_train.shape, X_test.shape)

# 5. Побудова моделі
# Наївний баєсовий класифікатор
model = MultinomialNB()
model.fit(X_train, y_train)
# 6. Оцінка моделі
y_pred = model.predict(X_test)

# Точність моделі
accuracy = accuracy_score(y_test, y_pred)
print(f"Точність моделі: {accuracy:.2f}")

# Матриця похибок
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])
plt.xlabel("Прогноз")
plt.ylabel("Реальні мітки")
plt.title("Матриця похибок")
plt.show()

# Детальний звіт про класифікацію
report = classification_report(y_test, y_pred, target_names=['Negative', 'Positive'])
print(report)

# 7. Аналіз помилок
incorrect_indices = np.where(y_test != y_pred)[0]
print("Приклади помилок моделі:")
for i in incorrect_indices[:5]:  # Виведення перших 5 помилкових прогнозів
    print(f"Відгук: {data['review'].iloc[i]}")
    print(f"Очікуваний клас: {y_test.iloc[i]}, Передбачений клас: {y_pred[i]}")
    print("-" * 80)
```

Короткий опис секцій:

1. **Завантаження даних:** Ми використовуємо датасет IMDb, що містить відгуки з позитивними або негативними сентиментами.

2. **Передобробка:** Виконано очистку тексту, видалення пунктуації, токенізацію, видалення стоп-слів та лемматизацію.
3. **Векторизація:** Створено векторне подання тексту з використанням TF-IDF.
4. **Модель:** Для класифікації обрано наївний баєсовий класифікатор.
5. **Оцінка:** Проведено оцінку точності, побудовано матрицю похибок та звіт про класифікацію.
6. **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. Основними етапами виконаної роботи були:

1. **Збір та підготовка даних:**
    a. Завантажено датасет IMDb для аналізу сентименту.
    b. Проведено передобробку текстів: очищення від пунктуації, видалення стоп-слів, токенізація та лемматизація.
    c. Використано TF-IDF для векторизації тексту, що забезпечило якісне представлення текстових даних у вигляді числових векторів.
2. **Реалізація моделі класифікації:**
    a. Для класифікації відгуків було обрано наївний баєсовий класифікатор, який добре працює з текстовими даними.
    b. Модель була навчена на тренувальному наборі даних, а її ефективність перевірена на тестовому наборі.
3. **Оцінка ефективності моделі:**
    a. Досягнуто високої точності класифікації, що свідчить про якісне виконання передобробки даних та вибір відповідного алгоритму.
    b. Побудовано матрицю похибок, яка дозволила виявити типи помилок моделі.
    c. Проведено аналіз помилок, зокрема ідентифіковано зразки, які були класифіковані неправильно.
4. **Аналіз методів:**
    a. Використання TF-IDF для векторизації тексту продемонструвало свою ефективність у задачі класифікації.
    b. 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.