Самостійна робота

Пошук схожих за темами постів

Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити процес пошуку подібних за темою постів у датасеті.

Хід роботи:

Завдання. Написати програму, яка буже шукати схожі за темою пости, використовуючи запит користувача.

Бібліотеки, які потрібно підключити:

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score
```

Для тренування та тестування моделі буде використовуватися датасет Medium Articles, який має наступні поля: author, claps, reading_time, link, title, text.

Завантажимо дані та видалимо рядки, які повторюються (рис. 1).

```
# 1. Завантажуємо та змінюємо дані
articles = pd.read_csv('articles.csv')
articles = articles.drop_duplicates(subset='title', keep='first')
articles['id'] = range(1, len(articles) + 1)
articles = articles[['id', 'title', 'text']]
print("Кількість записів:", len(articles))
print("Поля:", articles.columns.tolist())
print(articles.head(5))
print("-"*50)
```

```
Кількість записів: 230
Поля: ['id', 'title', 'text']
   id                                          title                                               text
0   1  Chatbots were the next big thing: what happene...  Oh, how the headlines blared:\nChatbots were T...
1   2  Python for Data Science: 8 Concepts You May Ha...  If you've ever found yourself looking up the s...
2   3  Automated Feature Engineering in Python – Towa...  Machine learning is increasingly moving from h...
3   4  Machine Learning: how to go from Zero to Hero ...  If your understanding of A.I. and Machine Lear...
4   5  Reinforcement Learning from scratch – Insight ...  Want to learn about applied Artificial Intelli...
--------------------------------------------------
```

Рис. 1. Виведення інформації про датасет

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.25.121.13.000 – Ср		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Козлик С.О.			Звіт з самостійної роботи	Літ.	Арк. 1 / Аркушів 8
Перевір.		Маєвський О.В.					
Керівник						ФІКТ, гр. ІПЗ-22-2	
Н. контр.							
Зав. каф.							

Далі необхідно векторизувати текст. Для цього використаємо TF-IDF, який перетворює текст на числові ознаки і дає зможу оцінити важливість певного слова відносно інших у всіх постах. TF визначає, як часто з'являється слово в певному пості, а IDF – навпаки, наскільки рідко слово зустрічається у всіх постах (чим рідкісніше слово, тим більше значення IDF).

```
# 2. Створюємо TF-IDF вектори
vectorizer = TfidfVectorizer(stop_words='english', max_features=10000)
tfidf_matrix = vectorizer.fit_transform(articles['title'])
```

Наступним кроком кластеризуємо пости у кількість кластерів (рис. 2), визначених змінною n_clusters. Це робиться для того, щоб згрупувати схожі між собою за темою пости. Далі пошук подібних до запиту постів буде відбуватися лише в релевантному кластері, щоб прискорити і покращити якість пошуку.

```
# 3. Кластеризація TF-IDF векторів
n_clusters = 20
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
articles['cluster'] = kmeans.fit_predict(tfidf_matrix)
print(articles['cluster'].value_counts())
print("-"*50)
```

```
cluster
7     34
17    27
1     22
8     21
9     18
3     18
18    12
4     11
11    10
12    10
19     8
16     8
0      8
6      5
13     4
2      3
14     3
5      3
15     3
10     2
Name: count, dtype: int64
--------------------------------------------------
```

Рис. 2. Виведення кількості записів в утворених кластерах

Створимо та використаємо функцію search_with_clusters, яка шукає схожі за темою пости в певному кластері. Алгоритм її роботи:

1. Перетворення запиту у TF-IDF вектор.
2. Обчислення його схожості з центрами кластерів.
3. Визначення найбільш подібного кластера.
4. Пошук найбільш схожих постів лише у вибраному кластері.

```
# 4. Функція пошуку схожих постів
def search_with_clusters(df, tfidf_matrix, vectorizer, query, top_n=5):
    # визначення вектора TF-IDF для запиту
    query_vector = vectorizer.transform([query])

    # визначення найближчого кластера
    cluster_distances = cosine_similarity(query_vector, kmeans.cluster_centers_)
    best_cluster = np.argmax(cluster_distances)

    # статті з цього кластера
    cluster_indices = df[df['cluster'] == best_cluster].index
    cluster_pos = df.index.get_indexer(cluster_indices)  # позиції у tfidf_matrix
    cluster_tfidf = tfidf_matrix[cluster_pos, :]

    # визначення косинусної схожості
    similarities = cosine_similarity(query_vector, cluster_tfidf).flatten()

    # визначення Top-N позицій у кластері за подібністю
    top_idx = similarities.argsort()[::-1][:top_n]

    # отримання правильних індексів постів у DataFrame
    cluster_indices_list = cluster_indices.to_list()
    top_article_indices = [cluster_indices_list[i] for i in top_idx]

    # отримання топ-статей
    top_articles = df.loc[top_article_indices].copy()
    top_articles['scores'] = similarities[top_idx]
    return top_articles
```

Передаємо запит у функцію та виводимо отримані результати (рис. 3), тобто id та назви постів, їхню оцінку схожості (для цього використовується метрика Cosine similarity, яка вимірює подібність між двома ненульовими векторами і базується на косинусі кута між ними; для векторів TF-IDF значення зазвичай від 0 до 0.5 – 0.7).

```
query = "Recurrent Neural Network in TensorFlow"
search_results = search_with_clusters(
    articles, tfidf_matrix, vectorizer, query, top_n=5
)
```

```
print("Схожі пости:")
for _, row in search_results.iterrows():
    print(f"ID: {row['id']}")
    print(f"TITLE: {row['title']}")
    print(f"SCORE: {row['scores']:.4f}")
    print(f"CLUSTER: {row['cluster']}")
    print("-"*50)
    print()
```

```
Схожі пости:
ID: 136
TITLE: How to build a Recurrent Neural Network in TensorFlow (1/7)
SCORE: 0.8826
CLUSTER: 11
--------------------------------------------------

ID: 118
TITLE: How to build your own Neural Network from scratch in Python
SCORE: 0.3409
CLUSTER: 11
--------------------------------------------------

ID: 190
TITLE: You can build a neural network in JavaScript even if you don’t really understand neural networks
SCORE: 0.3088
CLUSTER: 11
--------------------------------------------------

ID: 105
TITLE: How to build a multi-layered neural network in Python
SCORE: 0.2902
CLUSTER: 11
--------------------------------------------------

ID: 82
TITLE: How to build a simple neural network in 9 lines of Python code
SCORE: 0.2696
CLUSTER: 11
--------------------------------------------------

Оптимальна кількість кластерів за Silhouette Score: 28
```

Рис. 3. Результати пошуку схожих постів за запитом

Наступним кроком є оцінка якості кластеризації. Її проводимо, враховуючи дві метрики: Silhouette score і Inertia. Обчислимо їх для різної кількості кластерів (від 2 до 30) і подивимося, яка кількість дає найкращі результати.

```
# 6. Визначення метрик Silhouette score і Inertia
sil_scores = []
inertia = []
K = range(2, 30)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    labels = kmeans.fit_predict(tfidf_matrix)
    score = silhouette_score(tfidf_matrix, labels)
    sil_scores.append(score)
    inertia.append(kmeans.inertia_)
```

```python
plt.plot(K, sil_scores, marker='o')
plt.xlabel("Кількість кластерів")
plt.ylabel("Silhouette score")
plt.title("Silhouette method для KMeans")
plt.show()

best_k_silhouette = K[sil_scores.index(max(sil_scores))]
print(f"Оптимальна кількість кластерів за Silhouette Score: {best_k_silhouette}")

plt.plot(K, inertia, marker='o')
plt.xlabel("Кількість кластерів")
plt.ylabel("Inertia")
plt.title("Elbow method для KMeans")
plt.show()
```

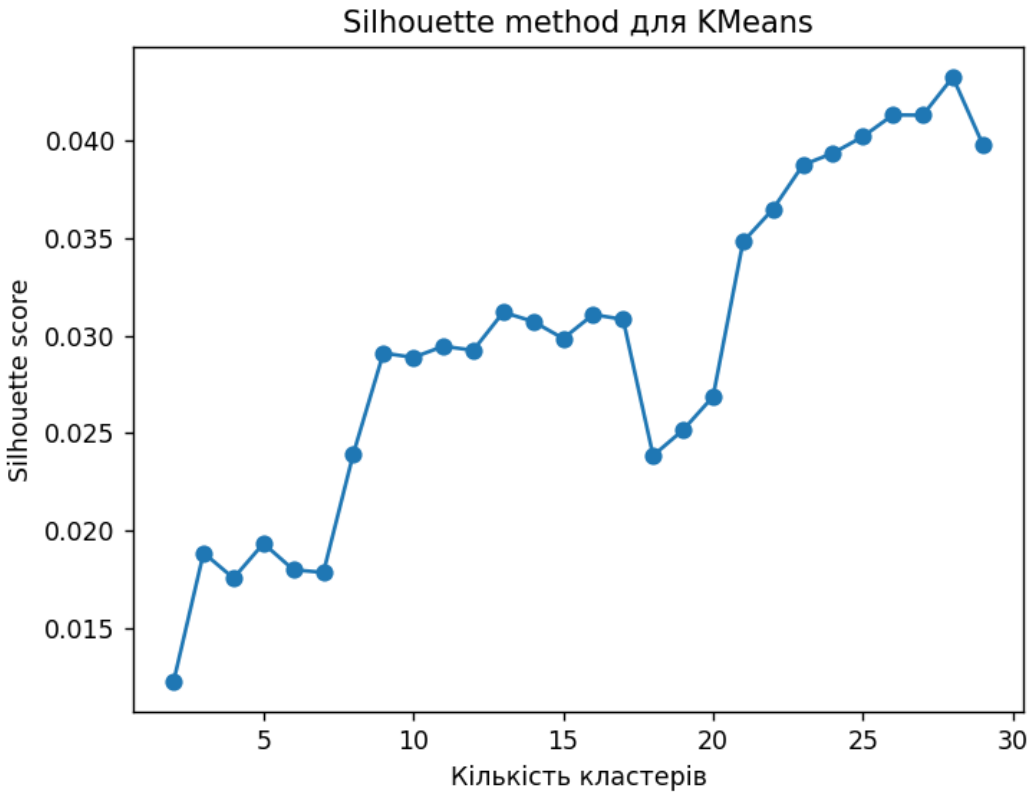

Рис. 4. Графік зміни значення метрики Silhouette score

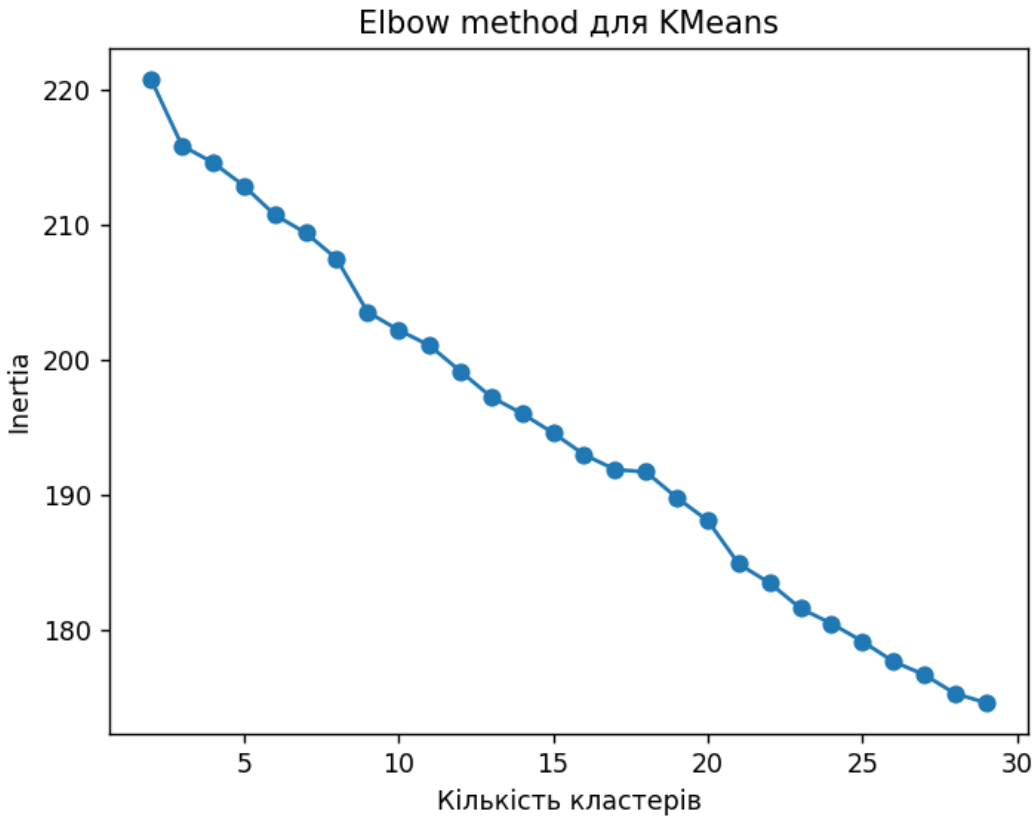

Рис. 5. Графік зміни значення метрики Inertia

На графіках (рис. 4 – 5) видно, що 28 кластерів дають найкращі результати вимірів з-поміж інших, тому змінимо значення n_clusters на 28.

```
Схожі пости:
ID: 136
TITLE: How to build a Recurrent Neural Network in TensorFlow (1/7)
SCORE: 0.8826
CLUSTER: 3
--------------------------------------------------

ID: 223
TITLE: Neural Network Architectures – Towards Data Science
SCORE: 0.3686
CLUSTER: 3
--------------------------------------------------

ID: 118
TITLE: How to build your own Neural Network from scratch in Python
SCORE: 0.3409
CLUSTER: 3
--------------------------------------------------

ID: 190
TITLE: You can build a neural network in JavaScript even if you don’t really understand neural networks
SCORE: 0.3088
CLUSTER: 3
--------------------------------------------------

ID: 105
TITLE: How to build a multi-layered neural network in Python
SCORE: 0.2902
CLUSTER: 3
--------------------------------------------------
```

Рис. 6. Результати повторного пошуку схожих постів за запитом

На рис. 6 можна побачити, що якість кластеризації поліпшилася, і програма краще шукає схожі пости з вищими оцінками подібності.

У коді для створення TF-IDF векторів було використано назви постів. Проте можна здійснювати векторизацію тексту, використовуючи назву та повний текст статті. Тоді ми будемо шукати схожі між собою за вмістом пости. Для цього створимо нове поле в датасеті.

```
articles['all_text'] = articles['title'].astype(str) + ' ' +
articles['text'].astype(str)
```

Векторизуємо текст використовуючи поле «all_text».

```
tfidf_matrix = vectorizer.fit_transform(articles['all_text'])
```

Виберемо один з постів датасету для прикладу, щоб знайти схожі на нього статті (рис. 7).

```
query = articles.loc[articles['id'] == 136, 'all_text'].iloc[0]
search_results = search_with_clusters(
    articles, tfidf_matrix, vectorizer, query, top_n=5
)
```

```
Схожі пости:
ID: 136
TITLE: How to build a Recurrent Neural Network in TensorFlow (1/7)
SCORE: 1.0000
CLUSTER: 22
--------------------------------------------------

ID: 67
TITLE: A simple deep learning model for stock price prediction using TensorFlow
SCORE: 0.3951
CLUSTER: 22
--------------------------------------------------

ID: 222
TITLE: A Guide For Time Series Prediction Using Recurrent Neural Networks (LSTMs)
SCORE: 0.2672
CLUSTER: 22
--------------------------------------------------

ID: 139
TITLE: Neural networks for algorithmic trading. Simple time series forecasting
SCORE: 0.2116
CLUSTER: 22
--------------------------------------------------

ID: 70
TITLE: A Tour of The Top 10 Algorithms for Machine Learning Newbies
SCORE: 0.1814
CLUSTER: 22
--------------------------------------------------
```

Рис. 7. Результати пошуку схожих постів за вибраним з датасету

Для першого в списку поста метрика Cosine similarity дорівнює 1.0, тобто вектори ідентичні.

Висновок: в ході виконання самостійної роботи ми дослідили процес пошуку подібних за темою постів у датасеті, використовуючи спеціалізовані бібліотеки та мову програмування Python. Було успішно розроблено алгоритм пошуку, визначено наступні його переваги: пошук у межах кластера прискорює роботу на великих датасетах завдяки зменшенню кількості постів для порівняння, векторизація тексту у TF-IDF є простою та зрозумілою. Проте TF-IDF ігнорує семантику, тобто якщо статті схожі за змістом, але не за ключовими словами, то вони потраплять у різні кластери. Також пости Medium Articles мають велику тематичну різноманітність, тому KMeans не може добре їх згрупувати.

Репозиторій: https://github.com/SofiiaKozlyk/AI-systems