گزارش پروژه سوم داده‌کاوی

پرهام طالبیان و بهزاد اسمی خانلو

مقدمه

این گزارش به بررسی و اجرای یک پروژه پردازش زبان طبیعی (NLP) با استفاده از SciSpaCy می‌پردازد. در این پروژه، داده‌های متنی پردازش و تجزیه و تحلیل شده‌اند تا بتوان از آن‌ها برای خوشه‌بندی و مدل‌سازی موضوعی استفاده کرد.

پیش‌نیازها

ابتدا باید کتابخانه‌ها و ابزارهای مورد نیاز را نصب کنیم. این ابزارها شامل numpy، pandas، matplotlib، seaborn، tqdm، langdetect، spacy و مدل en_core_sci_lg، scikit-learn و scipy می‌باشند. همچنین، فایل داده 10k_df.csv نیز باید در دسترس باشد.

بارگذاری و پیش‌پردازش داده‌ها

ابتدا داده‌ها را بارگذاری کرده و برخی پیش‌پردازش‌ها را انجام می‌دهیم.

```
# Block 2
df_10k = pd.read_csv('10k_df.csv')

# Block 3
df_10k.fillna(value=" ", inplace=True)

# Block 4
df = df_10k.sample(1500, random_state=42)
del df_10k
```

تشخیص زبان و پیش‌پردازش متون

در این بخش، زبان متون تشخیص داده می‌شود، توکن‌ها استخراج و ریشه‌یابی شده و کلمات توقف حذف می‌شوند.

```
# Block 5
from langdetect import detect
import en_core_sci_lg
from nltk.stem import PorterStemmer
```

```
from nltk.corpus import stopwords
import re

def detect_language(text):
return detect(text)

def tokenize_text(text):
parser = en_core_sci_lg.load(disable=["tagger", "ner"])
parser.max_length = 3000000
doc = parser(text)
tokens = [token.text for token in doc]
return tokens

def stem_text(tokens):
stemmer = PorterStemmer()
stemmed_tokens = [stemmer.stem(token) for token in tokens]
return stemmed_tokens

def remove_stopwords(tokens):
stop_words = set(stopwords.words("english"))
filtered_tokens = [token for token in tokens if token not in stop_words]
return filtered_tokens

def remove_punctuation(text):
clean_text = re.sub(r"[^\w\s]", "", text)
return clean_text

# Sample usage
text =   " "
language = detect_language(text)
tokens = tokenize_text(text)
stemmed_tokens = stem_text(tokens)
filtered_tokens = remove_stopwords(tokens)
clean_text = remove_punctuation(text)
```

استخراج ویژگی‌ها

ویژگی‌های متون پیش‌پردازش‌شده با استفاده از TF-IDF استخراج می‌شوند.

```
# Block 6
def extract_tfidf_features(texts):
vectorizer = TfidfVectorizer(max_features=4096)
features = vectorizer.fit_transform(texts)
return features

preprocessed_texts = df['preprocessed_text'].tolist()
features = extract_tfidf_features(preprocessed_texts)
print("        ":    , features.shape)
```

کاهش بعد

بعد ویژگی‌ها با استفاده از PCA کاهش می‌یابد تا ۹۵% واریانس حفظ شود.

```
# Block 7
from sklearn.decomposition import PCA

n_components = 0.95
pca = PCA(n_components=n_components)
reduced_features = pca.fit_transform(features.toarray())
print("                ":   , reduced_features.shape)
```

خوشه‌بندی

با استفاده از روش KMeans، تعداد بهینه خوشه‌ها تعیین و داده‌ها خوشه‌بندی می‌شوند.

```
# Block 8
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
```

```
max_clusters = 30
inertia = []
for n_clusters in range(1, max_clusters+1):
kmeans = KMeans(n_clusters=n_clusters, random_state=42)
kmeans.fit(reduced_features)
inertia.append(kmeans.inertia_)

plt.plot(range(1, max_clusters+1), inertia, marker='o')
plt.xlabel(   '    ('
plt.ylabel(    '       ('
plt.title('Elbow Method')
plt.show()

distances = []
for i in range(1, len(inertia)):
distances.append(inertia[i-1] - inertia[i])

elbow_index = np.argmax(distances) + 1
best_n_clusters = elbow_index
print("        ":    , best_n_clusters)

kmeans = KMeans(n_clusters=best_n_clusters, random_state=42)
kmeans.fit(reduced_features)
labels = kmeans.labels_

unique, counts = np.unique(labels, return_counts=True)
cluster_counts = dict(zip(unique, counts))
print("            ":   , cluster_counts)
```

نمایش داده‌ها

برای نمایش داده‌های خوشه‌بندی‌شده از t-SNE استفاده می‌شود.

```
# Block 9
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans
```

```
import matplotlib.pyplot as plt

tsne = TSNE(n_components=2, random_state=42)
tsne_features = tsne.fit_transform(reduced_features)

kmeans = KMeans(n_clusters=4, random_state=42)
kmeans.fit(reduced_features)
cluster_labels = kmeans.labels_

plt.scatter(tsne_features[:, 0], tsne_features[:, 1], c=cluster_labels)
plt.title('t-SNE Scatterplot')
plt.xlabel(  '    t-SNE')
plt.ylabel(  '    t-SNE')
plt.show()
```

مدل‌سازی موضوعی

با استفاده از LDA، مدل‌سازی موضوعی برای هر خوشه انجام می‌شود.

```
# Block 10
from sklearn.decomposition import LatentDirichletAllocation

num_topics_per_cluster = 5

for cluster_id in range(num_clusters):
cluster_articles = articles[cluster_labels == cluster_id]

vectorizer = TfidfVectorizer(max_df=0.8, min_df=2, stop_words='english')
tfidf = vectorizer.fit_transform(cluster_articles)

lda = LatentDirichletAllocation(n_components=num_topics_per_cluster, random_sta
lda.fit(tfidf)

feature_names = vectorizer.get_feature_names()
for topic_id, topic in enumerate(lda.components_):
topic_keywords = [feature_names[i] for i in topic.argsort()[:-6:-1]]
```

```
print(f"   {cluster_id} -    {topic_id}: {', '.join(topic_keywords)}")
```

نتیجه‌گیری

در این پروژه، با استفاده از ابزارهای مختلف NLP و الگوریتم‌های یادگیری ماشین، داده‌های متنی پردازش و تحلیل شده و به خوشه‌های مختلف تقسیم شدند. همچنین، با استفاده از LDA مدل‌سازی موضوعی برای هر خوشه انجام گرفت.

توضیحات الگوریتم‌ها

در این بخش، توضیحاتی در مورد الگوریتم‌های استفاده‌شده ارائه می‌شود:

تشخیص زبان

الگوریتم LangDetect برای تشخیص زبان متون استفاده می‌شود. این الگوریتم با تحلیل کاراکترهای متون، زبان هر متن را تشخیص می‌دهد.

استخراج توکن‌ها

الگوریتم SciSpaCy برای استخراج توکن‌ها استفاده می‌شود. این الگوریتم با استفاده از مدل‌های پیش‌آموزش‌داده‌شده، توکن‌های متون علمی را استخراج می‌کند.

ریشه‌یابی و حذف کلمات توقف

الگوریتم PorterStemmer برای ریشه‌یابی کلمات استفاده می‌شود. همچنین، از لیست کلمات توقف NLTK برای حذف کلمات توقف از متون استفاده می‌شود.

استخراج ویژگی‌ها

الگوریتم TF-IDF برای استخراج ویژگی‌ها استفاده می‌شود. این الگوریتم با محاسبه اهمیت کلمات در متون مختلف، ویژگی‌های متون را استخراج می‌کند.

کاهش بعد

الگوریتم PCA برای کاهش بعد ویژگی‌ها استفاده می‌شود. این الگوریتم با حفظ واریانس بیشینه، بعد ویژگی‌ها را کاهش می‌دهد.

خوشه‌بندی

الگوریتم KMeans برای خوشه‌بندی داده‌ها استفاده می‌شود. این الگوریتم با محاسبه فاصله نمونه‌ها از مراکز خوشه‌ها، داده‌ها را به خوشه‌های مختلف تقسیم می‌کند.

نمایش داده‌ها

الگوریتم t-SNE برای نمایش داده‌های خوشه‌بندی‌شده در فضای دوبعدی استفاده می‌شود. این الگوریتم با حفظ ساختار فاصله‌ها در فضای ویژگی، داده‌ها را به فضای دوبعدی تبدیل می‌کند.

مدل‌سازی موضوعی

الگوریتم LDA برای مدل‌سازی موضوعی استفاده می‌شود. این الگوریتم با تحلیل هم‌زمان توزیع کلمات در متون، موضوعات مختلف را استخراج می‌کند.