گزارش پروژه داده‌کاوی

مقدمه

این پروژه شامل پردازش متون تحقیقاتی، استخراج ویژگی‌ها، کاهش ابعاد، خوشه‌بندی و مدل‌سازی موضوعی است. در ادامه به بررسی مراحل مختلف این پروژه می‌پردازیم.

کتابخانه‌های مورد نیاز

در این پروژه از کتابخانه‌های متعددی برای پردازش داده، تجزیه و تحلیل و مصورسازی استفاده شده است. این کتابخانه‌ها مجموعه‌ای از ابزارهای مفید برای تحلیل داده و پردازش زبان طبیعی هستند:

1. NumPy و Pandas: این دو کتابخانه برای کار با داده‌های عددی و ساختار داده‌های جدولی (مانند جداول Excel) استفاده می‌شوند. NumPy برای محاسبات عددی و Pandas برای کار با داده‌های جدولی.

2. Matplotlib و Seaborn: این دو کتابخانه برای رسم نمودارها و تصویرسازی داده‌ها استفاده می‌شوند. Matplotlib یک کتابخانه محبوب برای ایجاد نمودارهای متنوع است و Seaborn برای تولید نمودارهای زیبا و پیشرفته مورد استفاده قرار می‌گیرد.

3. TQDM: این کتابخانه برای نمایش نوار پیشرفت استفاده می‌شود که در زمان اجرای کدها برای نشان دادن پیشرفت مفید است.

4. Langdetect: این کتابخانه برای تشخیص زبان متن‌ها استفاده می‌شود. با استفاده از آن می‌توانید زبان متن‌ها را تشخیص دهید.

5. Spacy: این یک کتابخانه پردازش زبان طبیعی است که امکانات مختلفی برای پردازش متن‌ها فراهم می‌کند، از جمله تشخیص موجودیت‌های اسمی، تحلیل گرامری، و استفاده از مدل‌های زبانی پیش‌آموزش داده شده.

6. Scikit-learn: این کتابخانه برای یادگیری ماشین و تحلیل داده استفاده می‌شود و شامل ابزارهای مختلفی برای بخش‌های مختلفی از فرآیند تحلیل داده مانند استخراج ویژگی‌ها، کاهش بعد، خوشه‌بندی، و مدل‌سازی موضوعی است.

7. Gensim: این کتابخانه معمولاً برای مدل‌سازی موضوع و پردازش متن استفاده می‌شود، از جمله روش‌های مانند LDA (Latent Dirichlet Allocation) که به شما اجازه می‌دهد موضوعات مختلف در متون را تشخیص دهید.

لود کردن مجموعه داده

ابتدا باید مجموعه داده‌ای که شامل متون تحقیقاتی است را لود کنیم. سپس به دلیل زمان‌بر بودن پردازش کل داده‌ها، یک نمونه تصادفی از داده‌ها انتخاب می‌کنیم.

فایل CSV به نام k_df.csv10 با استفاده از تابع read_csv از کتابخانه‌ی Pandas خوانده شده و داده‌ها در یک داده‌فریم به نام df_10k ذخیره می‌شوند.

مقادیر گمشده (NaN) در داده‌فریم df_10k با مقدار یک رشته خالی (یعنی " ") جایگزین می‌شوند. استفاده از inplace=True باعث می‌شود این تغییرات به‌صورت مستقیم روی داده‌فریم df_10k اعمال شود و نیاز به اختصاص دادن به یک متغیر جدید نباشد.

یک نمونه تصادفی از 1500 رکورد از داده‌فریم df_10k گرفته می‌شود و در داده‌فریم df ذخیره می‌گردد. استفاده از random_state=42 باعث می‌شود که نمونه‌گیری به‌صورت تکرارپذیر (قابل بازتولید) انجام شود، به این معنی که هر بار اجرای این کد، همان نمونه 1500 تایی انتخاب خواهد شد. داده‌فریم df_10k از حافظه حذف می‌شود تا فضای حافظه آزاد شود.

پیش پردازش متن

برای پیش‌پردازش متون از کتابخانه‌های مختلفی مانند NLTK، Gensim یا SpaCy استفاده می‌شود. در اینجا به توضیح و پیاده‌سازی هر یک از مراحل پیش‌پردازش متن برای "body_text" مقاله‌ها با استفاده از SpaCy می‌پردازیم.

تشخیص زبان: تشخیص زبان مقاله‌ها با استفاده از کتابخانه langdetect انجام می‌شود. این کار برای بررسی زبان مورد استفاده در ویژگی "body_text" هر مقاله است. پس از تشخیص زبان، مقاله‌های غیرانگلیسی حذف می‌شوند. این کار به کاهش نویز و افزایش دقت مدل کمک می‌کند.

توکنایز کردن: توکنایز کردن فرآیندی است که در آن متن به واحدهای کوچکتری به نام توکن‌ها (کلمات یا جملات) تقسیم می‌شود. برای این مجموعه داده، از پایپ‌لاین en_core_sci_lg از کتابخانه SpaCy استفاده می‌شود که به طور خاص برای داده‌های زیستی و پزشکی بهینه شده است. با استفاده از این مدل، می‌توانیم از بردارهای 600 هزار کلمه‌ای و واژگان بزرگتر بهره ببریم که به افزایش دقت مدل کمک می‌کند. برای مقابله با محدودیت حداکثر یک میلیون کاراکترSpaCy ، می‌توان از max_length برای کاهش این محدودیت استفاده کرد.

Lemmatization یا Stemming

Lemmatization و Stemming دو روش متداول برای تبدیل کلمات به ریشه‌های آن‌ها هستند. Lemmatization به عنوان روشی دقیق‌تر کلمات را به ریشه‌ی اصلی آن‌ها تبدیل می‌کند، در حالی که Stemming با حذف پیشوندها و پسوندها تلاش می‌کند کلمات را به شکل پایه‌ای تر تبدیل کند.

Stopwords

Stopwords کلمات رایجی هستند که اطلاعات معنایی کمی دارند و معمولاً در تحلیل متن مفید نیستند مانند "the"، "and"و "is". حذف این کلمات باعث می‌شود که مدل‌ها بهتر بتوانند بر کلمات مهم و معنادار تمرکز کنند. SpaCy شامل لیستی از stopwords است که می‌توان آنها را در پیش پردازش متن حذف کرد.

حذف علائم نشانه‌گذاری: علائم نشانه‌گذاری (مانند نقطه، ویرگول، علامت سوال) معمولاً برای تحلیل معنایی متن لازم نیستند و می‌توانند نویز ایجاد کنند. حذف این علائم کمک می‌کند که مدل‌ها بهتر بتوانند کلمات و عبارات مهم را تشخیص دهند.

در کدهای بخش، ابتدا DetectorFactory.seed = 0 برای تعیین مقدار اولیه دترمینیستی در تشخیص زبان استفاده می‌شود. سپس مدل پردازش زبان طبیعی en_core_sci_lg از کتابخانه spaCy بارگذاری می‌شود. همچنین حداکثر طول متن به 3,000,000 کاراکتر افزایش می‌یابد.

سپس از تابعی برای تشخیص زبان متون استفاده می‌شود. اگر تشخیص زبان موفق نباشد، 'unknown' برگردانده می‌شود.

آنگاه زبان متون موجود در ستون body_text را تشخیص می‌دهیم و نتیجه را در ستون جدیدی به نام language ذخیره می‌کند. فقط مقالاتی را که زبان آن‌ها انگلیسی است نگه می‌داریم و بقیه مقالات را فیلتر می‌کند.

```
def preprocess_text(text, parser, stopwords, punctuations):

    doc = parser(text)

    tokens = [token.lemma_ for token in doc if token.lemma_ not in stopwords
              and token.lemma_ not in punctuations]

    return ' '.join(tokens)
```

این تابع برای پیش‌پردازش متن استفاده می‌شود. ابتدا متن ورودی به توکن‌های (کلمه‌های) جداگانه تجزیه و سپس با استفاده از lemma به صورت پایه‌ای‌ترین فرم کلمه (ریشه) تبدیل می‌شود. توکن‌هایی که در لیست توقف کلمات (stopwords) و علائم نگارشی (punctuations) هستند حذف می‌شوند.

سپس با این تابع پیش‌پردازش متون موجود در ستون body_text را با انجام داده و نتیجه را در ستون جدیدی به نام processed_text ذخیره می‌کنیم.

استخراج ویژگی

از الگوریتم TF-IDF برای تبدیل متن به بردارهای ویژگی استفاده شده است.

TF-IDF (Term Frequency-Inverse Document Frequency)

یکی از تکنیک‌ها برای تبدیل متن به قالب عددی است که اهمیت هر کلمه در یک متن خاص را نشان می‌دهد. این روش ترکیبی از دو مفهوم اصلی است:

TF (Term Frequency) فرکانس هر کلمه در یک سند خاص. به عبارت دیگر، نشان می‌دهد که هر کلمه چند بار در یک سند ظاهر شده است.

IDF (Inverse Document Frequency) یک معیار برای کاهش وزن کلماتی که در اکثر اسناد ظاهر می‌شوند. این مفهوم نشان می‌دهد که هر چقدر کلمه‌ای در اسناد بیشتری ظاهر شود، اهمیت کمتری دارد.

این روش باعث می‌شود که کلماتی که در تعداد کمی از اسناد ظاهر می‌شوند، وزن بیشتری نسبت به کلماتی که در اکثر اسناد ظاهر می‌شوند، داشته باشند.

برای اجرای این روش روی داده‌های پیش‌پردازش شده، می‌توانیم از کتابخانه‌ی `scikit-learn` در پایتون استفاده کنیم.

ابتدا یک نمونه از کلاس TfidfVectorizer از کتابخانه scikit-learn ایجاد می‌شود. پارامتر max_features=4096 به این معنی است که فقط 4096 ویژگی (کلمه) پرکاربرد (براساس مقدار TF-IDF) در نظر گرفته می‌شود. fit_transform بر روی ستون processed_text داده‌فریم df اعمال می‌شود. این عملیات، ابتدا tfidf_vectorizer را با متون آموزشی تنظیم (fit) می‌کند و سپس متون را به ماتریس ویژگی‌های TF-IDF تبدیل می‌کند. نتیجه این عملیات یک ماتریس پراکنده (sparse matrix) است که حاوی مقادیر TF-IDF برای هر متن می‌باشد.

برای کاهش ابعاد ویژگی‌های TF-IDF با استفاده از تحلیل مولفه‌های اصلی (PCA) عمل میکنیم. هدف حفظ 95٪ از واریانس داده‌هاست.

یک نمونه از کلاس `PCA` از کتابخانه scikit-learn ایجاد می‌شود. پارامتر `n_components=0.95` به این معنی است که PCA تعداد مولفه‌های اصلی را طوری انتخاب می‌کند که 95٪ از واریانس داده‌ها حفظ شود.

`fit_transform` بر روی ویژگی‌های TF-IDF اعمال می‌شود. این عملیات، ابتدا PCA را با ویژگی‌های TF-IDF تنظیم (fit) می‌کند و سپس ویژگی‌ها را به فضای مولفه‌های اصلی (principal components) تبدیل می‌کند. نتیجه این عملیات یک آرایه است که ویژگی‌های تبدیل‌شده را در خود نگه می‌دارد.

```
pca_df = pd.DataFrame(tfidf_pca)
```

در این خط، آرایه حاصل از PCA به یک داده‌فریم Pandas تبدیل می‌شود تا بررسی و تحلیل آن آسان‌تر باشد.

###خوشه‌بندی

این کد برای انجام خوشه‌بندی K-Means بر روی داده‌های کاهش‌یافته با PCA و تعیین تعداد بهینه خوشه‌ها با استفاده از روش Elbow Method است. مراحل اصلی این کد عبارتند از:

ایجاد لیستی برای ذخیره مقادیر اینرسی:

```
inertia = []
```

تعریف بازه‌ای از مقادیر K برای تست:

```
K_range = range(1, 31)
```

اجرای K-Means برای مقادیر مختلف K و ذخیره مقادیر اینرسی:

```
for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(pca_df)
    inertia.append(kmeans.inertia_)
```

در این قسمت، K-Means برای مقادیر مختلف K (از 1 تا 30) اجرا می‌شود و مقدار اینرسی (Inertia) برای هر K محاسبه و در لیست `inertia` ذخیره می‌گردد. اینرسی به‌عنوان مجموع مربعات فواصل نقاط داده از مرکز خوشه‌هایشان تعریف می‌شود و معیار خوبی برای ارزیابی کیفیت خوشه‌بندی است.

Elbow Method

```
plt.figure(figsize=(10, 6))
plt.plot(K_range, inertia, marker='o')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal K')
plt.show()
```

در این قسمت، نمودار Elbow Method ترسیم می‌شود که نشان می‌دهد چگونه مقدار اینرسی با افزایش تعداد خوشه‌ها تغییر می‌کند. نقطه‌ی بهینه (Elbow) جایی است که کاهش اینرسی به‌صورت محسوس کند می‌شود.

بر اساس نمودارElbow Method ، optimal_k ده بدست می آید.

اجرای K-Means با تعداد بهینه خوشه‌ها:

```
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
clusters = kmeans.fit_predict(pca_df)
```

در این قسمت، K-Meansبا تعداد بهینه خوشه‌ها اجرا شده و برچسب خوشه‌ها برای هر داده محاسبه می‌شود.

افزودن برچسب خوشه‌ها به داده‌فریم اصلی:

```
df['cluster'] = clusters
```

بررسی توزیع خوشه‌ها:

```
print(df['cluster'].value_counts())
```

کاهش بعد با t-SNE

t-SNE (t-Distributed Stochastic Neighbor Embedding)

t-SNE یک تکنیک کاهش ابعاد و بصری‌سازی است که به‌طور خاص برای نمایش داده‌های با ابعاد بالا به صورت دو یا سه‌بعدی استفاده می‌شود. این تکنیک توسط Laurens van der Maaten و Geoffrey Hinton در سال 2008 معرفی شده است. t-SNE برای حفظ ساختار محلی داده‌ها به کار می‌رود، به این معنی که داده‌هایی که در فضای با ابعاد بالا به هم نزدیک هستند، در فضای با ابعاد پایین نیز به هم نزدیک خواهند بود.

مراحل اصلی t-SNE:

محاسبه شباهت‌ها در فضای با ابعاد بالا:

- برای هر جفت داده در فضای با ابعاد بالا، احتمال شباهت محاسبه می‌شود. این احتمال بیانگر این است که دو نقطه داده با هم به عنوان همسایه در نظر گرفته می‌شوند.

محاسبه شباهت‌ها در فضای با ابعاد پایین:

- در فضای با ابعاد پایین (مثلاً دو بعدی)، احتمال مشابهی محاسبه می‌شود. t-SNE تلاش می‌کند این احتمالات را طوری تنظیم کند که با احتمالات فضای با ابعاد بالا تطابق داشته باشند.

کاهش اختلاف احتمالات (Kullback-Leibler divergence):

- t-SNE بهینه‌سازی انجام می‌دهد تا اختلاف بین توزیع احتمالات در فضای با ابعاد بالا و پایین را به حداقل برساند. این فرآیند منجر به نمایش مناسب‌تر و واقعی‌تر داده‌ها در فضای با ابعاد پایین می‌شود.

کد زیر t-SNE را برای کاهش ابعاد داده‌ها به دو بعد اعمال کرده و نتایج را به همراه برچسب‌های خوشه‌ها (Clusters) به صورت یک نمودار پراکندگی نمایش می‌دهد:

اعمال t-SNE برای کاهش ابعاد به 2

```
tsne = TSNE(n_components=2, random_state=42)
tsne_results = tsne.fit_transform(pca_df)
```

اضافه کردن نتایج t-SNE به داده‌فریم

```
df['tsne_dim1'] = tsne_results[0 ,:]
df['tsne_dim2'] = tsne_results[1 ,:]
```

نمایش نتایج t-SNE با برچسب‌های خوشه‌ها

```
plt.figure(figsize=(10, 8))
sns.scatterplot(
   x='tsne_dim1', y='tsne_dim2,'
   hue='cluster,'
   palette=sns.color_palette('hsv', optimal_k),
   data=df,
   legend='full,'
   alpha=0.6
)
plt.title('t-SNE results with K-Means clusters')
plt.xlabel('t-SNE Dimension 1')
```

```
plt.ylabel('t-SNE Dimension 2')
plt.legend(title='Cluster')
plt.show()
```

مدلسازی موضوعی

برای استخراج موضوعات کلیدی از هر خوشه از روش LDA استفاده شد و سپس کلمات کلیدی هر موضوع استخراج و نمایش داده شدند.

LDA(تخصیص دیریکله نهان) یک مدل مولد آماری است که برای شناسایی موضوعات پنهان (مخفی) در مجموعه‌ای از اسناد متنی استفاده می‌شود. این مدل فرض می‌کند که هر سند ترکیبی از چندین موضوع و هر موضوع ترکیبی از کلمات است LDA .به‌طور خودکار موضوعات و توزیع کلمات مرتبط با هر موضوع را از مجموعه اسناد استخراج می‌کند.

مراحل اصلی LDA

مدل مولد:

- فرض بر این است که هر سند مجموعه‌ای از کلمات است که از چندین موضوع تشکیل شده‌اند. هر موضوع نیز مجموعه‌ای از کلمات با توزیع احتمال خاص خود است.

LDA مدل مولد این فرایند را شبیه‌سازی می‌کند که چگونه کلمات در اسناد با توجه به موضوعاتشان تولید می‌شوند.

پارامترهای مدل:

- `α` (آلفا): پارامتر دیریکله برای توزیع موضوعات در سند.

- `β` - (بتا): پارامتر دیریکله برای توزیع کلمات در موضوع.

تخصیص موضوعات:

- هر کلمه در هر سند به یک موضوع اختصاص داده می‌شود به طوری که تخصیص نهایی به بهترین شکل ممکن توزیع کلمات و موضوعات را تبیین کند.

الگوریتم‌های یادگیری:

- الگوریتم‌های مختلفی برای آموزش مدل LDA استفاده می‌شوند، مانند Gibbs Sampling و Variational Bayes.

کد برای استخراج موضوعات با استفاده از LDA

کد زیر یک تابع برای استخراج موضوعات با استفاده از LDA تعریف می‌کند و سپس این تابع برای هر خوشه از اسناد اعمال می‌شود:

تعریف تابع برای استخراج موضوعات با استفاده ازLDA

```
def extract_topics_lda(docs, n_topics=5, n_top_words=10):
```

برداری‌سازی اسناد

```
vectorizer = CountVectorizer(max_features=1000, stop_words='english')
doc_term_matrix = vectorizer.fit_transform(docs)
```

آموزش مدل LDA

```
lda = LatentDirichletAllocation(n_components=n_topics, random_state=42)
```

```
    lda.fit(doc_term_matrix)
```

استخراج نام‌های ویژگی (کلمات)

```
    feature_names = vectorizer.get_feature_names_out()
```

استخراج موضوعات و چاپ کلمات برتر

```
    topics = []
    for topic_idx, topic in enumerate(lda.components_):
        top_words = [feature_names[i] for i in topic.argsort()[:-n_top_words - 1:-1]]
        topics.append(top_words)
    return topics
```

استخراج موضوعات برای هر خوشه

```
n_topics_per_cluster = 3  # تعداد موضوعات برای هر خوشه را می‌توان تنظیم کرد
n_top_words = 10  # تعداد کلمات برتر برای نمایش در هر موضوع

for cluster in range(optimal_k):
    cluster_docs = df[df['cluster'] == cluster]['processed_text']
    print(f"Cluster {cluster}:")

    if not cluster_docs.empty:
        topics = extract_topics_lda(cluster_docs, n_topics=n_topics_per_cluster, n_top_words=n_top_words)
```

```
    for i, topic in enumerate(topics):
        print(f"  Topic {i+1}: {', '.join(topic)}")
  else:
    print("  No documents in this cluster.")
  print()
```

این کد کمک می‌کند تا موضوعات اصلی موجود در هر خوشه از اسناد را شناسایی و بررسی کنسم. این فرایند می‌تواند به درک بهتر از محتوای اسناد و ساختار موضوعی آن‌ها کمک کند.