Semantic-Driven Topic Modeling Using Transformer-Based Embeddings and DBSCAN Clustering

در اینجا لینک مناسب برای دیتاست AG News در Hugging Face موجود است که می‌توانید از آن استفاده کنید:

در این حالت، با استفاده از کتابخانه‌ی datasets از Hugging Face ، داده‌ها به‌طور خودکار از این لینک بارگذاری می‌شوند و نیازی به آپلود آن‌ها در مخزن GitHub نیست.

dataset = load_dataset("ag_news", split="train[:500]")

https://huggingface.co/datasets

بارگذاری کتابخانه‌ها و توابع موردنیاز:

در ابتدا باید تمامی کتابخانه‌ها و توابع موردنیاز را بارگذاری کنید. این کتابخانه‌ها شامل sentence-transformers، scikit-learn، umap-learn، hdbscan و datasets است.

نصب پکیج‌ها:

!pip install sentence-transformers scikit-learn pandas numpy hdbscan umap-learn datasets

بارگذاری کتابخانه‌ها:

```
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
import hdbscan
import umap
from sklearn.metrics.pairwise import cosine_similarity
from sentence_transformers import SentenceTransformer
from datasets import load_dataset
import warnings
```

نادیده گرفتن اخطارها:

```
warnings.filterwarnings("ignore")
```

بارگذاری داده‌ها از اینترنت:

در این بخش، داده‌ها را از Hugging Face با استفاده از کتابخانه datasetsبارگذاری می‌کنیم. در اینجا از AG News به عنوان دیتاست استفاده می‌کنیم که داده‌ها را به‌صورت مستقیم بارگذاری می‌کند

بارگذاری داده‌ها ازAG News :

```
print("Loading data...")
dataset = load_dataset("ag_news", split="train[:500]")
documents = dataset["text"]
true_labels = dataset["label"]
print(f"Data loaded: {len(documents)} documents")
```

تعریف مدل Semantic-Driven Topic Modeling با DBSCAN وUMAP

در این بخش، مدل موردنظر که شامل DBSCAN برای خوشه‌بندی و UMAP برای کاهش ابعاد است، تعریف می‌شود. همچنین مرحله‌ی استخراج موضوعات با استفاده از این مدل انجام می‌شود.

کلاس برای مدل Semantic Topic Modeling با DBSCAN و UMAP:

```
class SemanticTopicModelDBSCAN:
   def __init__(self, n_topics=3, embedding_model='all-MiniLM-L6-v2', min_cluster_size=10,
umap_dim=5):
      self.n_topics = n_topics
      self.model = SentenceTransformer(embedding_model)
      self.vectorizer = CountVectorizer(stop_words='english', max_features=1000)
```

مدل UMAP برای کاهش ابعاد:

```
      self.umap_model = umap.UMAP)
         n_neighbors=15,
         n_components=umap_dim,
         metric="cosine,"
```

```
        random_state=42
(
```

خوشه‌بندی با HDBSCAN:

```
    self.clusterer = hdbscan.HDBSCAN)
        min_cluster_size=min_cluster_size,
        metric="euclidean,"
        cluster_selection_method="eom"
(
```

Generating Embeddings: تبدیل سند متنی به بردار عددی:

```
  def fit_transform(self, docs):
    print("1. Generating Embeddings...")
    self.doc_embeddings = self.model.encode(docs, show_progress_bar=False)
    print("2. Reducing Dimensions with UMAP...")
```

Reducing Dimensions with UMAP:

```
    self.embeddings_umap = self.umap_model.fit_transform(self.doc_embeddings)
    print("3. Clustering with DBSCAN...")
```

Clustering with DBSCAN:

```
    self.labels = self.clusterer.fit_predict(self.embeddings_umap)
    n_clusters = len(set(self.labels)) - (1 if -1 in self.labels else 0)
    print(f"Clusters found (excluding noise): {n_clusters}")
    print("4. Semantic topic extraction...")
```

Semantic topic extraction:

```
    return self._extract_topics(docs)
  def _extract_topics(self, docs, top_n=10):
```

ساخت واژگان مهم باCountVectorizer :

```
    vocab = self.vectorizer.fit(docs).get_feature_names_out()
    word_embeddings = self.model.encode(vocab, show_progress_bar=False)
```

```
    topics{} =
    for label in set(self.labels):
        if label == -1:
            continue
```

محاسبه مرکز خوشه:

```
        idx = np.where(self.labels == label)[۰]
        center = np.mean(self.doc_embeddings[idx], axis=0)
```

محاسبه شباهت کسینوسی بین مرکز خوشه و تمام واژگان:

```
        sims = cosine_similarity([center], word_embeddings)[۰]
        top_words = [vocab[i] for i in sims.argsort()[-top_n:][::-1]]
```

ذخیره نتایج در دیکشنری:

```
        topics[f"Topic {label}"] = top_words
    return topics
```

اجرای مدل HDBSCAN و مقایسه باLDA

در این بخش، مدل HDBSCAN اجرا می‌شود و نتایج استخراج شده با مدل LDA مقایسه می‌شود.

مدل LDA برای مقایسه:

```
def run_lda_baseline(docs, n_topics=3):
    print("Running LDA Baseline...")
```

آماده‌سازی داده‌ها باCountVectorizer :

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

تعریف و آموزش مدلLDA :

```
    lda = LatentDirichletAllocation(n_components=n_topics, random_state=42)
    lda.fit(X)
```

استخراج کلمات کلیدی هر موضوع:

```
    topics = {}
    feature_names = vectorizer.get_feature_names_out()
    for topic_idx, topic in enumerate(lda.components_):
        top_words = [feature_names[i] for i in topic.argsort()[:-11:-1]]
        topics[f"Topic {topic_idx+1}"] = top_words
    return topics
```

ارزیابی Semantic Coherence :

```
def calculate_semantic_coherence(topics_dict, model):
    scores = []
    for topic, words in topics_dict.items():
        if not words: continue
        embeddings = model.encode(words)
```

محاسبه شباهت کسینوسی بین کلمات موضوع:

```
        sim_matrix = cosine_similarity(embeddings)
```

محاسبه میانگین شباهت معنایی:

```
        avg_sim = (np.sum(sim_matrix) - len(words)) / (len(words) * (len(words)-1))
        scores.append(avg_sim)
    return np.mean(scores)
```

اجرای مدل‌ها:

```
stm_dbscan = SemanticTopicModelDBSCAN(n_topics=3, min_cluster_size=25)
dbscan_topics = stm_dbscan.fit_transform(documents)
lda_topics = run_lda_baseline(documents, n_topics=3)
```

نمایش نتایج:

```
print("\n" + "="*50)
print("نتایج روش مقاله (Semantic-Driven):")
print("="*50)
for topic, words in dbscan_topics.items():
    print(f"{topic}: {', '.join(words)}")

print("\n" + "="*50)
print("نتایج روش سنتی (LDA):")
print("="*50)
for topic, words in lda_topics.items():
    print(f"{topic}: {', '.join(words)}")
```

مقایسه Semantic Coherence Score :

```
print("\n" + "="*50)
print("مقایسه نهایی (Semantic Coherence Score):")
print("(امتیاز بالاتر نشان‌دهنده ارتباط معنایی قوی‌تر کلمات در یک موضوع است)")
print("="*50)
stm_score = calculate_semantic_coherence(dbscan_topics, stm_dbscan.model)
lda_score = calculate_semantic_coherence(lda_topics, stm_dbscan.model)
print(f"Paper Method (DBSCAN + UMAP) Score: {stm_score:.4f}")
print(f"LDA Baseline Score: {lda_score:.4f}")
if stm_score > lda_score:
    print("\n✅ نتیجه: روش مقاله کلمات مرتبط‌تری را استخراج کرده است.")
else:
    print("\n❌ نتیجه: تفاوت معناداری مشاهده نشد (ممکن است به دلیل حجم کم داده باشد)")
```

رسم نمودار UMAP برای خوشه‌ها

در این بخش، خوشه‌ها را با استفاده از **UMAP** رسم می‌کنیم.

رسم نمودار UMAP برای خوشه‌ها

```
import matplotlib.pyplot as plt
import seaborn as sns
def plot_semantic_clusters(model_instance):
    umap_2d = umap.UMAP(
        n_neighbors=15,
        n_components=2,
        metric="cosine",
        random_state=42
    ).fit_transform(model_instance.doc_embeddings)

    df_viz = pd.DataFrame({
        'x': umap_2d[:, 0],
        'y': umap_2d[:, 1],
        'cluster': model_instance.labels
    })
    plt.figure(figsize=(12, 8))
    sns.scatterplot(
        data=df_viz,
        x='x',
        y='y',
        hue='cluster',
        palette='viridis',
        s=100,
```

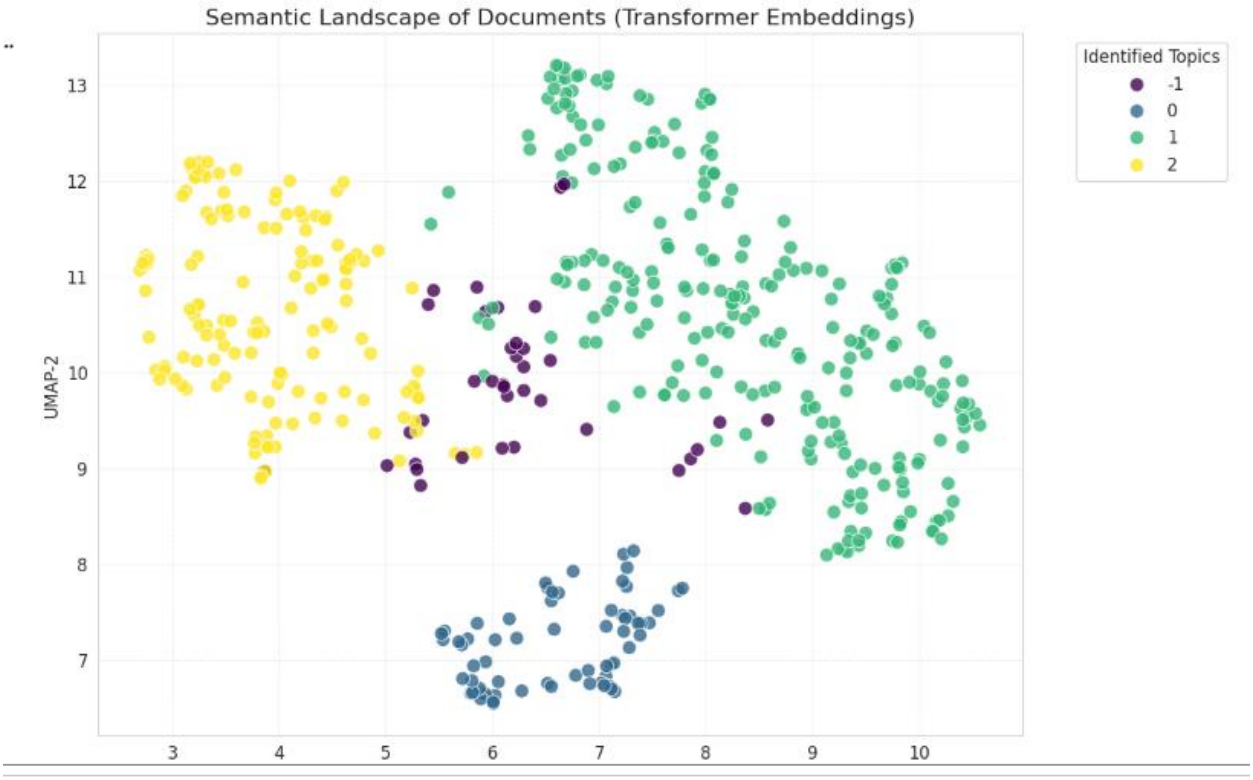
```
        alpha=0.8
    )
    plt.title('Semantic Landscape of Documents (Transformer Embeddings)', fontsize=16)
    plt.xlabel('UMAP-1', fontsize=12)
    plt.ylabel('UMAP-2', fontsize=12)
    plt.legend(title='Identified Topics', bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.grid(True, linestyle='--', alpha=0.3)
    plt.tight_layout()
    plt.show()
```

رسم نمودار

```
print("Drawing the scatter plot...")
plot_semantic_clusters(stm_dbscan)
```

نتیجه‌گیری:

در انتها، نتایج ارزیابی مدل‌ها را مشاهده کرده و تحلیل می‌کنید که کدام مدل DBSCAN+UMAP یا LDA عملکرد بهتری داشته است.



nlp final.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▷ Run all

```
/usr/local/lib/python3.12/dist-packages/hdbscan/robust_single_linkage_.py:175: SyntaxWarning: invalid escape sequence '\{'
  $max \{ core_k(a), core_k(b), 1/\alpha d(a,b) \}$.
WARNING:torchao.kernel.intmm:Warning: Detected no triton, on systems without Triton certain kernels will not work
Loading data...
README.md: 8.07k/? [00:00<00:00, 624kB/s]
data/train-00000-of-00001.parquet: 100% 18.6M/18.6M [00:02<00:00, 13.4MB/s]
data/test-00000-of-00001.parquet: 100% 1.23M/1.23M [00:00<00:00, 2.60MB/s]
Generating train split: 100% 120000/120000 [00:00<00:00, 422355.96 examples/s]
Generating test split: 100% 7600/7600 [00:00<00:00, 198902.49 examples/s]
Data loaded: 500 documents
modules.json: 100% 349/349 [00:00<00:00, 28.6kB/s]
config_sentence_transformers.json: 100% 116/116 [00:00<00:00, 8.79kB/s]
README.md: 10.5k/? [00:00<00:00, 770kB/s]
sentence_bert_config.json: 100% 53.0/53.0 [00:00<00:00, 5.27kB/s]
config.json: 100% 612/612 [00:00<00:00, 53.4kB/s]
model.safetensors: 100% 90.9M/90.9M [00:00<00:00, 162MB/s]
tokenizer_config.json: 100% 350/350 [00:00<00:00, 31.7kB/s]
vocab.txt: 232k/? [00:00<00:00, 12.1MB/s]
tokenizer.json: 466k/? [00:00<00:00, 23.0MB/s]
special_tokens_map.json: 100% 112/112 [00:00<00:00, 9.66kB/s]
config.json: 100% 190/190 [00:00<00:00, 15.8kB/s]
1. Generating Embeddings...
```

خروجی:

1. Generating Embeddings...

2. Reducing Dimensions with UMAP...

3. Clustering with DBSCAN...

Clusters found (excluding noise): 3

4. Semantic topic extraction...

Running LDA Baseline...

==

نتایج روش مقاله (Semantic-Driven):

==

Topic 0: olympic, olympics, athletes, sports, inning, competition, qualifying, rivals, athens, race

Topic 1: microsoft, ibm, companies, market, hackers, markets, industry, corp, mozilla, verizon

Topic 2: nasa, meteor, comets, spacecraft, scientists, planets, astronauts, solar, planet, observatory

==

نتایج روش سنتی (LDA):

==

Topic 1: space, ap, com, new, microsoft, reuters, said, nasa, windows, work

Topic 2: lt, gt, ap, reuters, space, new, team, scientists, oil, phone

Topic 3: google, ap, new, ipo, company, search, time, olympic, engine, says

==

مقایسه نهایی (Semantic Coherence Score):

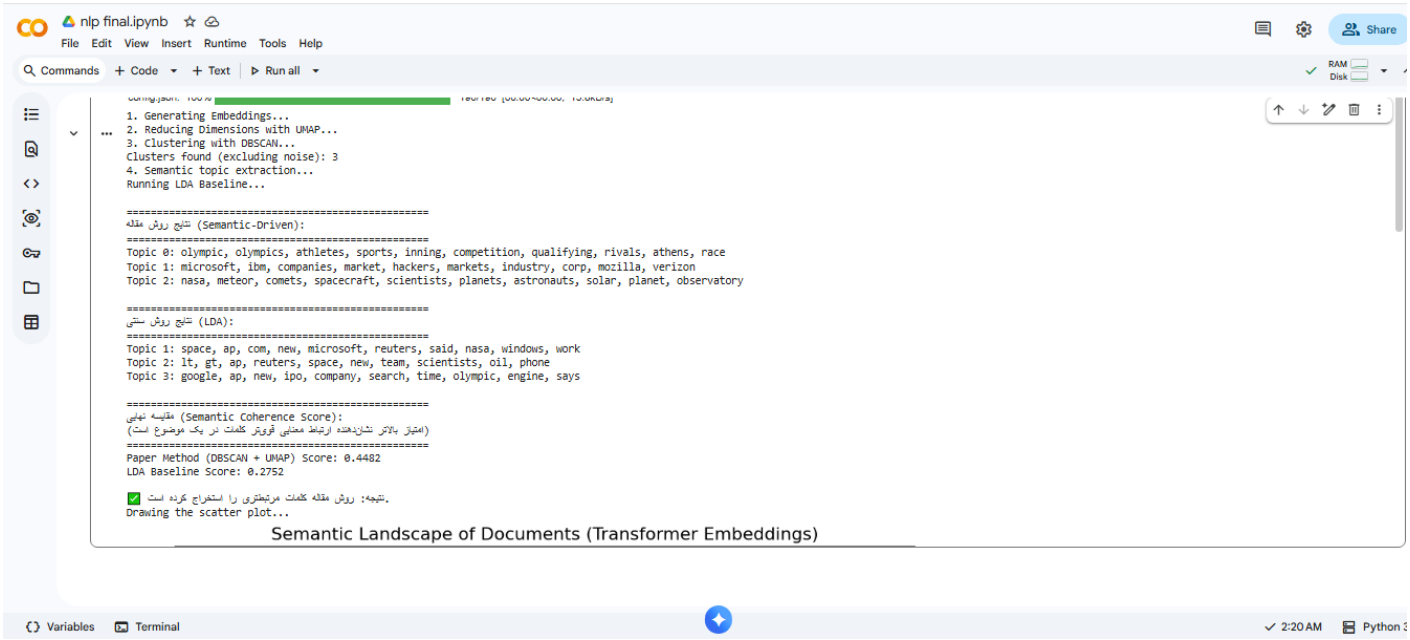
(امتیاز بالاتر نشان‌دهنده ارتباط معنایی قوی‌تر کلمات در یک موضوع است)

==

Paper Method (DBSCAN + UMAP) Score: 0.4482

LDA Baseline Score: 0.2752

✅ **نتیجه: روش مقاله کلمات مرتبط‌تری را استخراج کرده است.**


nlp final.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
1. Generating Embeddings...
2. Reducing Dimensions with UMAP...
3. Clustering with DBSCAN...
Clusters found (excluding noise): 3
4. Semantic topic extraction...
Running LDA Baseline...
==
نتایج روش مقاله (Semantic-Driven):
==
Topic 0: olympic, olympics, athletes, sports, inning, competition, qualifying, rivals, athens, race
Topic 1: microsoft, ibm, companies, market, hackers, markets, industry, corp, mozilla, verizon
Topic 2: nasa, meteor, comets, spacecraft, scientists, planets, astronauts, solar, planet, observatory
==
نتایج روش سنتی (LDA):
==
Topic 1: space, ap, com, new, microsoft, reuters, said, nasa, windows, work
Topic 2: lt, gt, ap, reuters, space, new, team, scientists, oil, phone
Topic 3: google, ap, new, ipo, company, search, time, olympic, engine, says
==
مقایسه نهایی (Semantic Coherence Score):
(امتیاز بالاتر نشان‌دهنده ارتباط معنایی قوی‌تر کلمات در یک موضوع است)
==
Paper Method (DBSCAN + UMAP) Score: 0.4482
LDA Baseline Score: 0.2752
نتیجه: روش مقاله کلمات مرتبط‌تری را استخراج کرده است.
Drawing the scatter plot...
Semantic Landscape of Documents (Transformer Embeddings)
Variables
Terminal
2:20 AM
Python 3