News Recommendation Service

`Ge Jiang`

github: https://github.com/egotist0/news_recommendation_service

Introduction

EVA is an open-source artificial intelligence relational database that supports deep learning models. It is designed to facilitate artificial intelligence database applications capable of leveraging deep learning models to process both structured and unstructured data. The database has built-in support for popular vector databases like Pinecone. ***The project aims to extend the news recommendation tool developed in Project 1 into a practical web platform.***

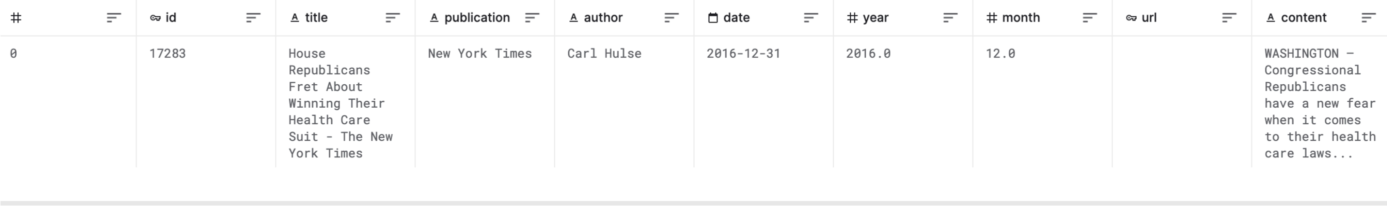
It utilizes *EvaDB, the ChatGPT model, and Pinecone for semantic similarity matching*, providing features such as document summarization, keyword extraction, and entity recognition for handling database documents. *Also uses Flask to build a web application service*.

Based on a user's previous reading history, the tool selects the top 10 articles from a library that are most likely to align with the reader's preferences and presents the recommendations.

Data Sources

The article data is derived from kaggle, encompassing 143,000 articles sourced from 15 prominent American publications, such as the New York Times, Breitbart, CNN, Business Insider, the Atlantic, Fox News, Talking Points Memo, Buzzfeed News, National Review, New York Post, the Guardian, NPR, Reuters, Vox, and the Washington Post.

This project exclusively focuses on the "articles1.csv" file within the dataset, containing 50,000 news articles (Articles 1-50,000). The file encompasses various attributes, including:



#	id	title	publication	author	date	year	month	url	content
0	17283	House Republicans Fret About Winning Their Health Care Suit - The New York Times	New York Times	Carl Hulse	2016-12-31	2016.0	12.0		WASHINGTON — Congressional Republicans have a new fear when it comes to their health care laws...

Related Work

Pinecone

Pinecone is an emerging service and tool crafted to aid organizations and developers in the effective management and utilization of extensive vector data. It stands out as a high-performance vector indexing and retrieval system tailored specifically for machine learning applications.

Functioning as a robust infrastructure, Pinecone excels in storing, indexing, and searching vector embeddings. These embeddings represent data points numerically, capturing their semantic information and relationships. Widely applicable across domains like natural language processing, computer vision, recommendation systems, and anomaly detection, vector embeddings play a crucial role.

A standout feature of Pinecone lies in its adept handling of high-dimensional vector data. Leveraging advanced indexing techniques, including approximate nearest neighbor search algorithms, Pinecone facilitates swift and accurate retrieval of similar vectors. This capability proves invaluable in scenarios demanding real-time or near-real-time responses, such as personalized recommendations or similarity-based searches.

GloVe

The Global Vectors for Word Representation (GloVe) stands as a widely embraced word embedding model crafted by researchers at Stanford University. This model represents words as dense vectors in a high-dimensional space, capturing their semantic relationships through co-occurrence patterns. GloVe merges global statistical insights with local context to generate word vectors. Through the factorization of a co-occurrence matrix, GloVe produces word vectors where the dot product signifies the likelihood of word co-occurrence.

GloVe excels in capturing both syntactic and semantic relationships among words, rendering it invaluable for various natural language processing tasks, including word similarity computation, text classification, and machine translation. Known for its simplicity, efficiency, and effectiveness, GloVe has garnered widespread adoption in both academic and industrial settings. Pre-trained GloVe word vectors are readily available for multiple languages, seamlessly integrated into machine learning models, thereby fostering advancements in language understanding and text analysis.

Below are the modifications I will make for Project 1:

1. Rename the corresponding repo on Github to: evadb_New_Recommendation.

2. Convert the code from an ipynb file format to separate Python files. The expected approach is as follows:

   Build a web application that provides news recommendation services. Utilize Pinecone SDK, EvaDB, and Python Flask to develop the backend application, abstracting the functions from the ipynb file into a service. The frontend will use HTML, static CSS, and JS resources for construction.

3. Import the necessary packages and environment for building the program into a requirement.txt file or develop it directly as a Docker image.
4. Enhance Readme.md by adding detailed instructions for application startup, environment setup, and a tutorial on how to use the web application.

Technical Details

Data Preprocessing

Use Python's Pandas library to process data in CSV format. The code is as follows:

```python
def prepare_data(data):
    # rename id column and remove unnecessary columns
    data.rename(columns={"Unnamed: 0": "article_id"}, inplace=True)
    data.drop(columns=['date'], inplace=True)

    # extract only first few sentences of each article for quicker vector calculations
    data['content'] = data['content'].fillna('')
    data['content'] = data.content.swifter.apply(
        lambda x: ' '.join(re.split(r'(?<=[.:;])\s', x)[:4]))
    data['title_and_content'] = data['title'] + ' ' + data['content']

    # create a vector embedding based on title and article columns
    encoded_articles = model.encode(
        data['title_and_content'], show_progress_bar=True)
    data['article_vector'] = pd.Series(encoded_articles.tolist())

    return data



def process_file(filename):
    data = pd.read_csv(filename)
    data = prepare_data(data)
    upload_items(data)

    return data
```

Build Vector Index

I have established a vector index utilizing the Title and Content attributes for each article. The vectors are configured with a dimension of 300, and the metric is specified as "cosine." This index is constructed within Pinecone to streamline subsequent similarity searches across the entire content and titles of the article database.

```
def initialize_pinecone():
    cursor = evadb.connect().cursor()
    warnings.filterwarnings("ignore")

    # Set api key
    api_key = ''
    os.environ["PINECONE_API_KEY"] = api_key

    openai.api_key = ""

    # Set environment
    environment = 'gcp-starter'
    os.environ["PINECONE_ENV"] = environment

    PINECONE_API_KEY = os.environ["PINECONE_API_KEY"]
    pinecone.init(api_key=api_key, environment=environment)


def delete_existing_pinecone_index():
    if PINECONE_INDEX_NAME in pinecone.list_indexes():
        pinecone.delete_index(PINECONE_INDEX_NAME)


def create_pinecone_index():
    pinecone.create_index(
    dimension=300, name=PINECONE_INDEX_NAME, metric="cosine", shards=1)
    pinecone_index = pinecone.Index(index_name=PINECONE_INDEX_NAME)

    return pinecone_index

def upload_items(data):
    upsert_batch = []
    for i, row in data.iterrows():
        upsert_batch.append((str(row.id), row.article_vector))

        if len(upsert_batch) > 500:
            pinecone_index.upsert(upsert_batch)
            upsert_batch = []

    # Process any remaining data in upsert_batch
    if upsert_batch:
        pinecone_index.upsert(upsert_batch)
```

Model

The vector creation model employed is GloVe.

```
def create_model():
    model = SentenceTransformer('average_word_embeddings_komninos')

    return model
```

Recommendation Logic

Pinecone is designed to fetch articles that closely align with a specified object vector, determined by the similarity of the index. Given that a reader might have engaged with various articles, the reference points for the similarity search consist of multiple vectors. Given that the Similarity function in EvaDB lacks the capability to search for the similarity of multiple objects, I have chosen to utilize the query API directly offered by Pinecone.

```
def query_pinecone(reading_history_ids):
    reading_history_ids_list = list(map(int, reading_history_ids.split(',')))
    reading_history_articles = uploaded_data.loc[uploaded_data['id'].isin(
        reading_history_ids_list)]

    article_vectors = reading_history_articles['article_vector']
    reading_history_vector = [*map(mean, zip(*article_vectors))]

    query_results = pinecone_index.query(
        vector=[reading_history_vector], top_k=10)
    res = query_results['matches']

    results_list = []

    for idx, item in enumerate(res):
        results_list.append({
            "title": titles_mapped[int(item.id)],
            "publication": publications_mapped[int(item.id)],
            "score": item.score,
        })

    return json.dumps(results_list)
```

Web Service

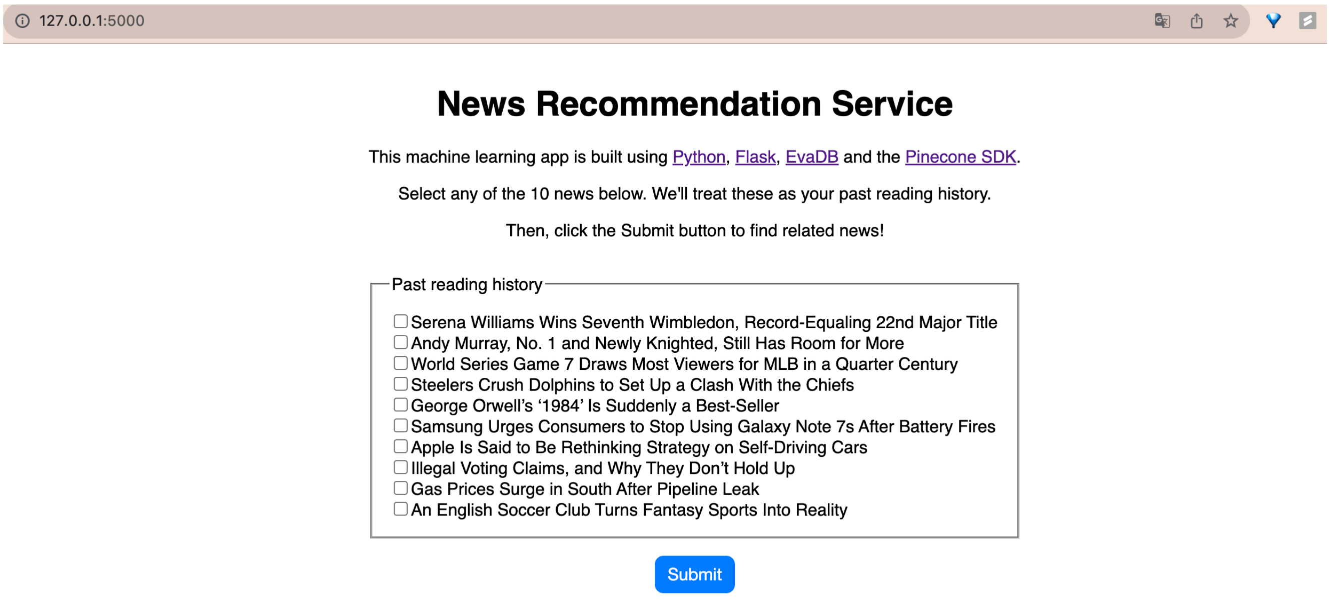
Implemented in Flask

```
app = Flask(__name__)

app.route("/")
def index():
    return render_template("index.html")
```

```
@app.route("/api/search", methods=["POST", "GET"])
def search():
    if request.method == "POST":
        return query_pinecone(request.form.history)
    if request.method == "GET":
        return query_pinecone(request.args.get("history", ""))
    return "Only GET and POST methods are allowed for this endpoint"


if __name__ == '__main__':
    app.run()
```

Sample

- Homepage



News Recommendation Service

This machine learning app is built using Python, Flask, EvaDB and the Pinecone SDK.

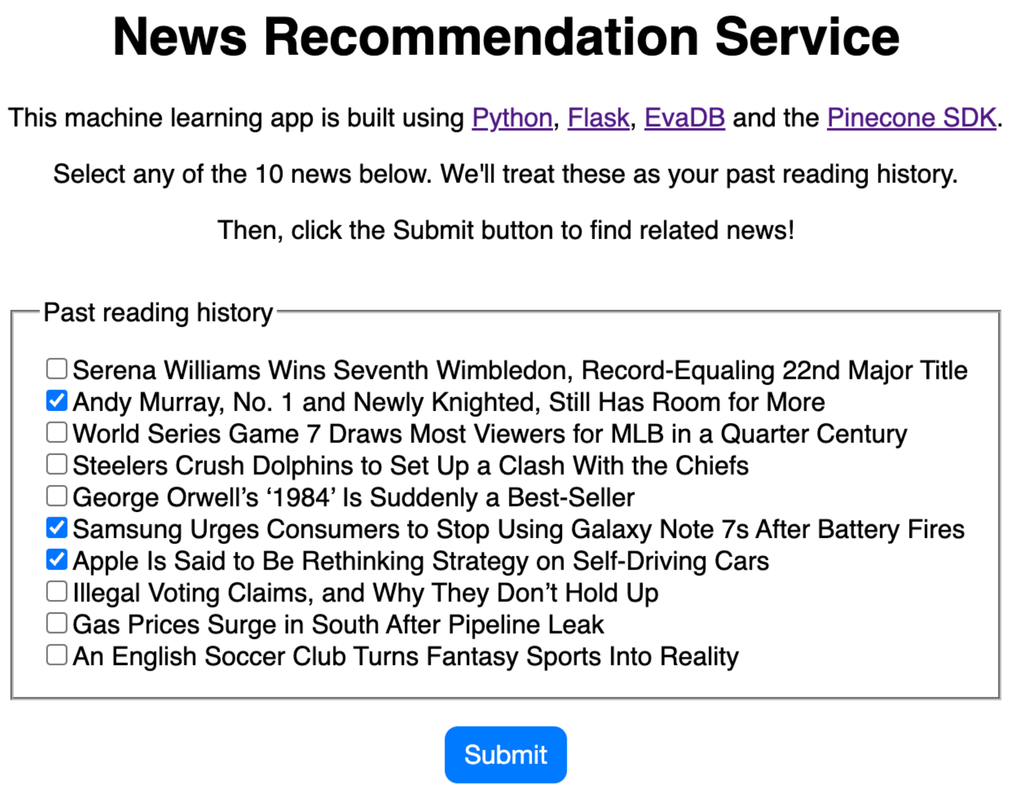
Select any of the 10 news below. We'll treat these as your past reading history.

Then, click the Submit button to find related news!

Past reading history

□ Serena Williams Wins Seventh Wimbledon, Record-Equaling 22nd Major Title
□ Andy Murray, No. 1 and Newly Knighted, Still Has Room for More
□ World Series Game 7 Draws Most Viewers for MLB in a Quarter Century
□ Steelers Crush Dolphins to Set Up a Clash With the Chiefs
□ George Orwell's '1984' Is Suddenly a Best-Seller
□ Samsung Urges Consumers to Stop Using Galaxy Note 7s After Battery Fires
□ Apple Is Said to Be Rethinking Strategy on Self-Driving Cars
□ Illegal Voting Claims, and Why They Don't Hold Up
□ Gas Prices Surge in South After Pipeline Leak
□ An English Soccer Club Turns Fantasy Sports Into Reality

- Select any articles you like in initial recommentation



News Recommendation Service

This machine learning app is built using Python, Flask, EvaDB and the Pinecone SDK.

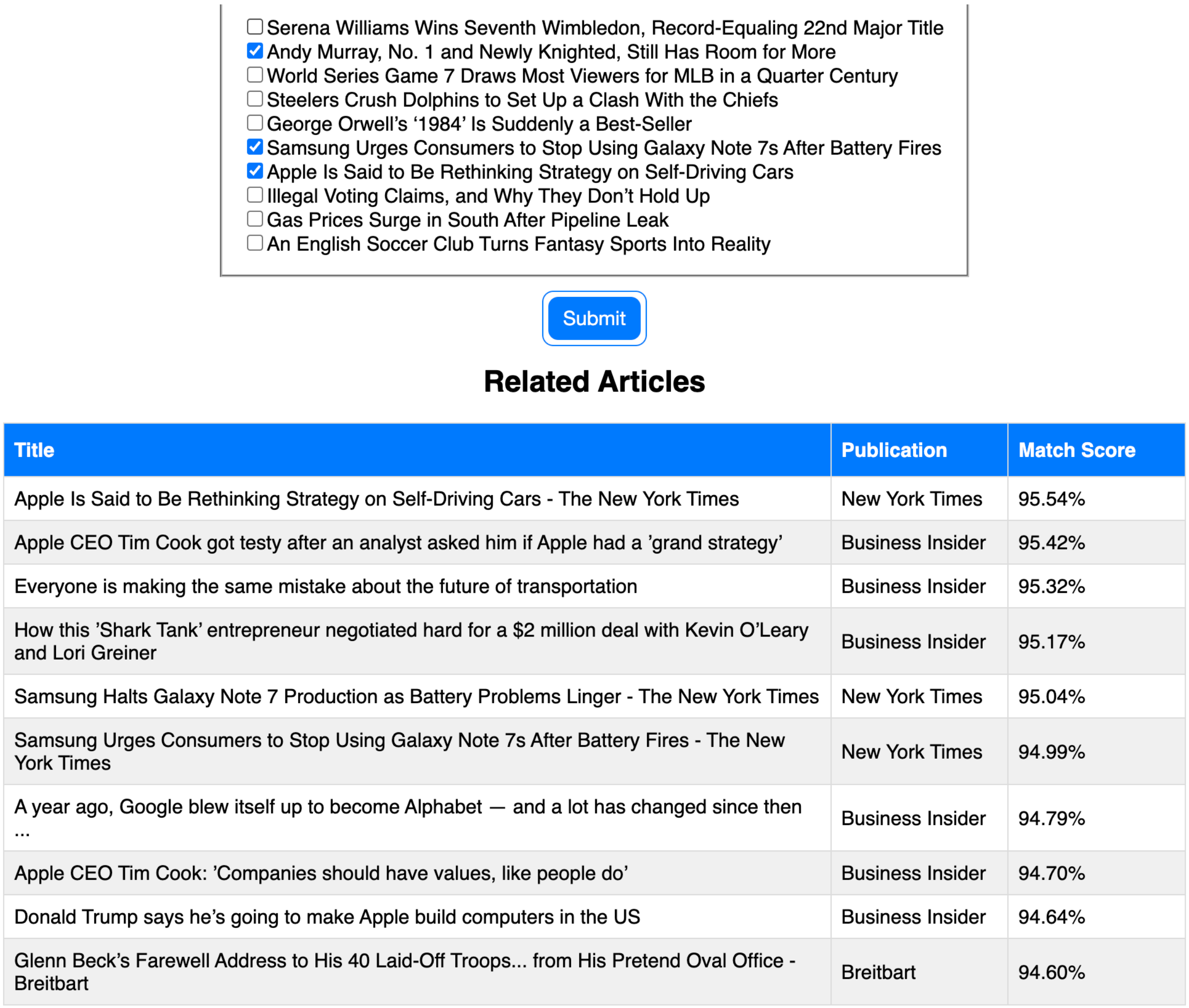
Select any of the 10 news below. We'll treat these as your past reading history.

Then, click the Submit button to find related news!

Past reading history

☐ Serena Williams Wins Seventh Wimbledon, Record-Equaling 22nd Major Title
☑ Andy Murray, No. 1 and Newly Knighted, Still Has Room for More
☐ World Series Game 7 Draws Most Viewers for MLB in a Quarter Century
☐ Steelers Crush Dolphins to Set Up a Clash With the Chiefs
☐ George Orwell's '1984' Is Suddenly a Best-Seller
☑ Samsung Urges Consumers to Stop Using Galaxy Note 7s After Battery Fires
☑ Apple Is Said to Be Rethinking Strategy on Self-Driving Cars
☐ Illegal Voting Claims, and Why They Don't Hold Up
☐ Gas Prices Surge in South After Pipeline Leak
☐ An English Soccer Club Turns Fantasy Sports Into Reality

- Tap Submit to get recommentation News and select new interested articles



□ Serena Williams Wins Seventh Wimbledon, Record-Equaling 22nd Major Title
☑ Andy Murray, No. 1 and Newly Knighted, Still Has Room for More
□ World Series Game 7 Draws Most Viewers for MLB in a Quarter Century
□ Steelers Crush Dolphins to Set Up a Clash With the Chiefs
□ George Orwell’s ‘1984’ Is Suddenly a Best-Seller
☑ Samsung Urges Consumers to Stop Using Galaxy Note 7s After Battery Fires
☑ Apple Is Said to Be Rethinking Strategy on Self-Driving Cars
□ Illegal Voting Claims, and Why They Don’t Hold Up
□ Gas Prices Surge in South After Pipeline Leak
□ An English Soccer Club Turns Fantasy Sports Into Reality

Related Articles

Title	Publication	Match Score
Apple Is Said to Be Rethinking Strategy on Self-Driving Cars - The New York Times	New York Times	95.54%
Apple CEO Tim Cook got testy after an analyst asked him if Apple had a ’grand strategy’	Business Insider	95.42%
Everyone is making the same mistake about the future of transportation	Business Insider	95.32%
How this ’Shark Tank’ entrepreneur negotiated hard for a $2 million deal with Kevin O’Leary and Lori Greiner	Business Insider	95.17%
Samsung Halts Galaxy Note 7 Production as Battery Problems Linger - The New York Times	New York Times	95.04%
Samsung Urges Consumers to Stop Using Galaxy Note 7s After Battery Fires - The New York Times	New York Times	94.99%
A year ago, Google blew itself up to become Alphabet — and a lot has changed since then ...	Business Insider	94.79%
Apple CEO Tim Cook: ’Companies should have values, like people do’	Business Insider	94.70%
Donald Trump says he’s going to make Apple build computers in the US	Business Insider	94.64%
Glenn Beck’s Farewell Address to His 40 Laid-Off Troops... from His Pretend Oval Office - Breitbart	Breitbart	94.60%

Usage

- Create a new conda env or use python venv

```
conda create --name your_env_name python=3.10
```

- Activate your env

```
conda activate your_env_name
```

- CD to your current working dir
- Install the required python package.

```
conda create --name your_env_name --file requirements.txt
```

- Download data from https://www.kaggle.com/datasets/snapcrack/all-the-news using wget.
- Run app.py

```
python app.py
```

Then you can see the website on http://127.0.0.1:5000/

Testing Approach

The detection method in this section aligns with Project 1 — utilizing the EvaDB ChatGPT API interface. Input consists of the titles and content of the articles the reader has previously read, along with the titles and content of ten recommended articles. Subsequently, ChatGPT assesses them from a semantic perspective.

Reference

- https://www.kaggle.com/datasets/snapcrack/all-the-news/data
- https://docs.pinecone.io/docs/choosing-index-type-and-size
- https://nlp.stanford.edu/projects/glove/ref=hackernoon.com
- https://github.com/thawkin3/pinecone-demo
- https://evadb.readthedocs.io/en/latest/source/reference/evaql/create_table.html