CDW Documentation

Product Sentiment Analysis App

Product Sentiment Analysis App

This is a web application that searches X/Twitter for tweets about a product, analyzes their sentiment using Azure Cognitive Services and Azure OpenAI, and displays a summary.

Infrastructure

Required Services

- **Azure OpenAI Service** – Used to summarize tweet sentiments via the Chat Completions API.
- **Azure Text Analytics** – For classifying sentiment (positive, negative, neutral) of individual tweets via the Sentiment Analysis API.
- **Twitter API v2** – To fetch tweets related to the searched keyword using the recent search endpoint.
- **Azure App Service** – For hosting the backend using the FastAPI framework.

Environment Variables

- `AZURE_OPENAI_KEY`
- `AZURE_OPENAI_ENDPOINT`
- `AZURE_OPENAI_DEPLOYMENT_NAME`
- `AZURE_TEXT_ANALYTICS_KEY`
- `AZURE_TEXT_ANALYTICS_ENDPOINT`
- `X_BEARER_TOKEN`

Store them in a `.env` file or as App Settings in Azure App Service.

Backend Code (app.py)

Libraries Used

- `fastapi` – Defines HTTP endpoints.
- `requests` – Used to interact with the Twitter API.
- `openai` – Azure OpenAI SDK for calling GPT models.
- `azure.ai.textanalytics` – Azure SDK for Text Analytics API.
- `jinja2` – Template rendering.
- `uvicorn` – Development server for FastAPI.

Endpoints

- `GET /` – Renders the form (HTML template).
- `POST /analyze` – Handles form submission, performs:
    1. Tweet search.

2. Sentiment classification.
3. Summary generation.

Key Functions

search_tweets(product_name)

1. Calls Twitter API v2:
    1. Endpoint: `https://api.twitter.com/2/tweets/search/recent`
    2. Auth: Bearer Token (`X_BEARER_TOKEN`)
    3. Returns up to 10 recent English tweets about the keyword.

analyze_sentiment(texts)

1. Calls Azure Text Analytics API:
    1. Endpoint: `<AZURE_TEXT_ANALYTICS_ENDPOINT>/text/analytics/v3.1/sentiment`
    2. Auth: API Key (`AZURE_TEXT_ANALYTICS_KEY`)
    3. Returns document sentiment (positive, negative, neutral) and confidence scores.

summarize_with_gpt(tweets)

1. Calls Azure OpenAI Chat Completions API:
    1. Endpoint: `<AZURE_OPENAI_ENDPOINT>/openai/deployments/<DEPLOYMENT_NAME>/chat/completions`
    2. Model: gpt-35-turbo or compatible
    3. Prompts GPT to summarize public sentiment based on labeled tweet data.

Frontend Code (index.html)

Uses Bootstrap for styling. Features:

- Form for entering a product.
- Display area showing:
    1. Sentiment-labeled tweets.
    2. GPT-generated summary.

User Steps

1. Configure Environment

Create `.env` with: ``` AZURE_OPENAI_KEY=your_key
AZURE_OPENAI_ENDPOINT=https://your-endpoint.openai.azure.com/
AZURE_OPENAI_DEPLOYMENT_NAME=your_deployment

AZURE_TEXT_ANALYTICS_KEY=your_text_analytics_key
AZURE_TEXT_ANALYTICS_ENDPOINT=https://your-text-analytics.cognitiveservices.azure.com/
TWITTER_BEARER_TOKEN=your_twitter_bearer_token ```

2. Install Dependencies

Run:

```
pip install -r requirements.txt
```

3. Launch the App

Run:

```
uvicorn app:app --reload
```

Then open your browser to:

```
http://localhost:8000
```

4. Use the App

- Enter a product keyword (e.g., “AirPods”).
- Submit.
- View tweet list with sentiment tags.
- View summary generated by GPT.

File Structure

File	Purpose
`app.py`	Main backend logic
`templates/index.html`	Frontend
`requirements.txt`	Dependencies

How It Works

1. User inputs a keyword.
2. Tweets are fetched using Twitter API v2.
3. Sentiment analysis is performed using Azure Text Analytics.
4. Tweets are summarized using Azure OpenAI GPT.
5. Result is rendered via HTML template.