Redis Semantic Router for Document Classification

Wayne Cheng

User Story and Pain Points

- ~100K news articles per day need to be classified into 1 of 5 topics: business, entertainment, politics, sport, and tech

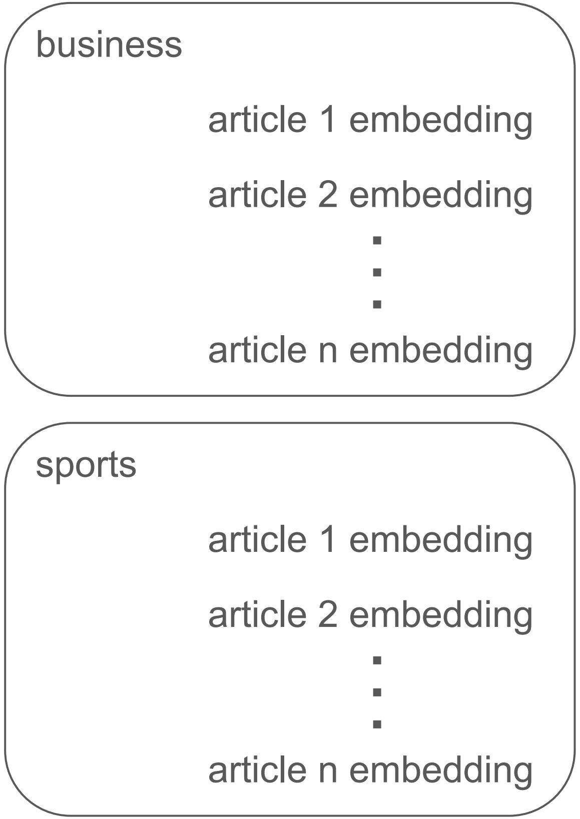
- Using a LLM (GPT) to classify the article

- Pain points:
  - Latency → bad user experience
  - Cost
  - Accuracy

Possible Solution

- Use a semantic router to classify articles
  - A semantic router takes the "semantics" (meaning) of text, and returns a "route" (class)
  - It's built with a vector database that returns the closest matching "route"

- Compared with a LLM, a semantic router is faster, less expensive, and possibly more accurate



business

article 1 embedding

article 2 embedding

⋮

article n embedding

sports

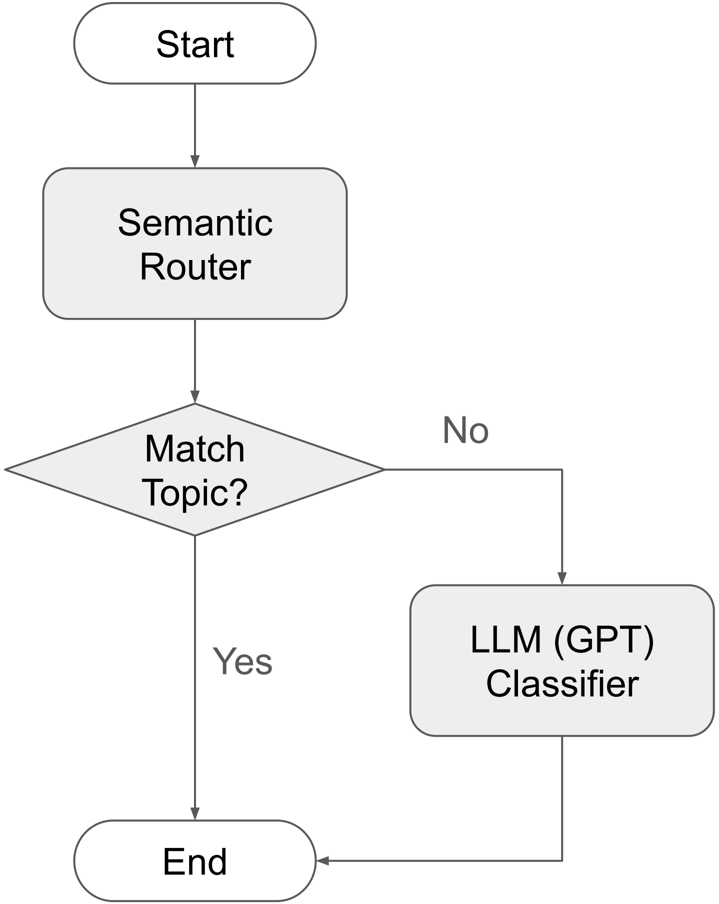
article 1 embedding

article 2 embedding

⋮

article n embedding

Architecture



Proof-Of-Concept Demo

- Github repository:
  - https://github.com/audoir/redis-project

- Prerequisites: Docker and Python must be installed

- Follow the instructions in the README.md file
- The configurations can be modified in src/config.py

Dataset

- The dataset contains articles, each labelled with a topic
- It is split into a test and train dataset
- Test dataset: 10 articles per topic (50 total)
- Train dataset: 50 articles per topic (250 total)

Baseline

- The test dataset is classified using a LLM (gpt-5-nano)
  - Simple classification task only requires a small model
  - Cost for input: $0.05 per million tokens
  - Cost for output: $0.40 per million tokens

- Average Latency: 5 seconds
- Average Cost = $0.00025
- Accuracy = 88%

With Semantic Router

- The train dataset is used for the routes
    - Hugging Face’s Sentence Transformers model (all-mpnet-base-v2) is used as the vectorizer
- The router distance threshold is tuned to 0.6 as a good balance between non-matches and false matches
    - The lower the threshold, the more non-matches
    - The higher the threshold, the more false matches

- Average Latency: 1 second
- Average Cost = $0.000034
- Accuracy = 92%
- Router match = 88%

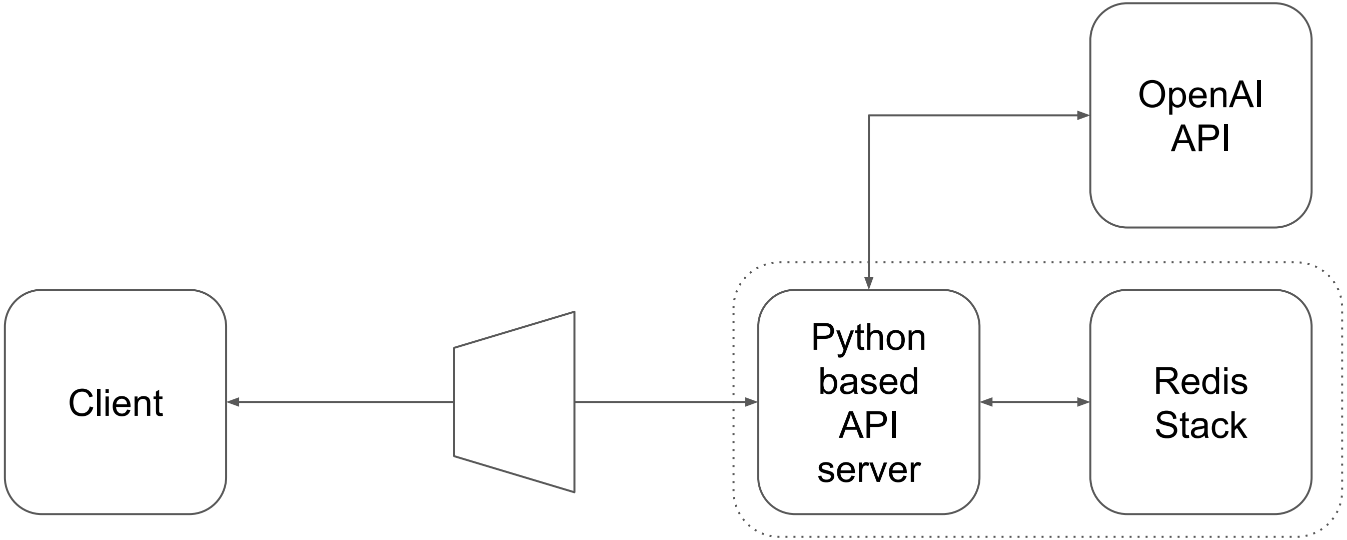
Comparison

- Percentage Improvements:
  - Latency: -79%
  - Cost: -86%
  - Accuracy: 4.6%

- For 100K articles:
  - Latency: **-112 hours**
  - Cost: **-$22** (will more than cover the cost of running Redis for a month)
  - Correct: **4000 articles**

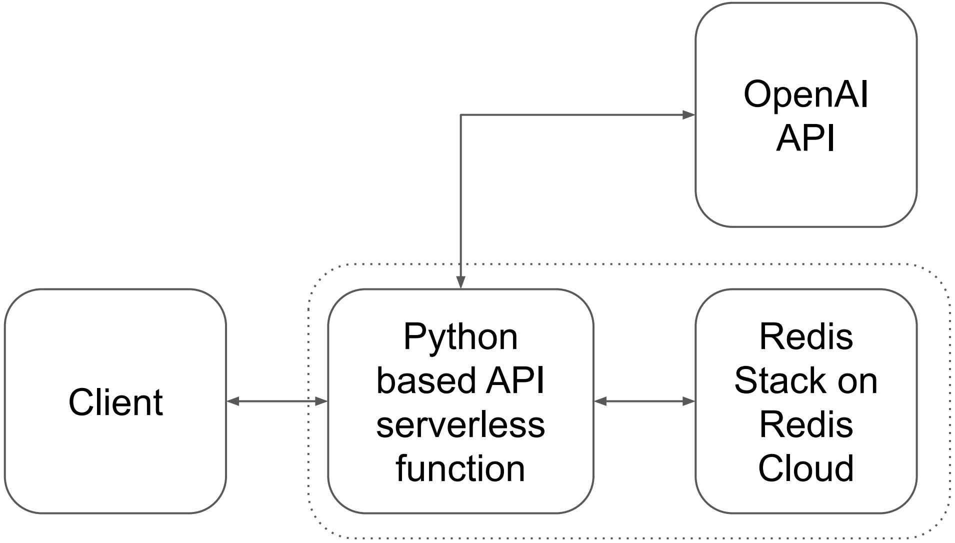
- The semantic router approach **reduces latency and cost**, while **improving accuracy**

System Design with Provisioned Compute



API server and Redis server can be replicated as needed based on demand

System Design with Serverless Services


OpenAI API
Client
Python based API serverless function
Redis Stack on Redis Cloud

Simple design → fast setup and auto-scaling

Extensibility

- Proactively create newer versions of the router with recent articles to maintain high accuracy, or add more topics

Why Redis?

- High performance using in-memory store
- Mature and well-adopted technology with lots of support

Fast and **easy** setup:

- RedisVL → build a semantic router with fewer lines of code
- Redis Cloud → fast prototyping and deployment; free to try

Implementation Strategy

- If you don't currently have Redis in your system, use Redis Cloud
- Build a prototype API server using the demo code as a template
- Duplicate the existing traffic to the prototype, and tune the configurations
- Once the latency and accuracy is satisfactory, switch over to using this architecture for production

- The total implementation time is a few hours

Recap

- Using a semantic router for classification task improves the user experience by **reducing latency** and **improving accuracy**, while **reducing cost**

- You may have to tune with train dataset size, distance threshold, and vectorizer to get the best results

- You can either deploy Redis in your own system, or use Redis cloud for fast setup

Additional Resources

- Redis Vector Library: https://docs.redisvl.com/en/latest/
  - AI-native Python client library used in this demo
- Redis Cloud: https://redis.io/docs/latest/operate/rc/
  - Fully managed Redis database on the cloud
- Redis Client APIs: https://redis.io/docs/latest/develop/clients/
  - Connect to Redis servers from languages like JS (NodeJS), Python, Go, Rust, Java, etc.

Tour of the code

Questions? Comments?