Take-Home Exercise. Political-Tweets ETL

By Reputación Digital

Introduction

At **Reputación Digital** 🩷 we use social data extracted from several social networks, and transform it using a wide set of tools, from simple **Hugging Face** models to **LLMs** with up to 7 B parameters. All this is done in a pipeline that prepares the data ready to be consumed by our analysts and clients.

🚨 **Your mission here is to show you can do similar to that. But small.**

We're not looking for the perfect candidate. The team is looking for **ownership and pragmatic trade-offs**. This implies responsibility with your work and knowing *why* you are doing what you're doing

Some heads-up before starting:

Estimated time: ~ 8 hours, but you will have the deadline established in the email you received this document. If you need more, note it in the README and tell us what you'd improve with extra time.

Tools: In the document we will be explaining what we want you to use. But feel free to use anything you love. **AI included!** We rely on AI every day.

Evaluation: We'll look at implementation, architecture, error handling, and code quality.

What you are expected to create.

1. **Extract** tweets content from the provided JSON file.
2. **FastAPI** routes (Python 3.11). **Mandatory.**
3. **Classify emotion** with a Hugging Face transformer through an endpoint. **Mandatory.**
4. **Detect stance tag** with a tiny local LLM through an endpoint. **Optional. This is a "WOW!" for us.**
5. **Ingest** everything into a single **Elasticsearch** index. **Mandatory.**
6. **Create** a one-page report.md with metrics, failure cases, and lessons learned. **Optional**
7. **Provide a Docker** setup (Makefile or docker compose up) that downloads models, starts Elasticsearch, and launches FastAPI. **Optional, but highly valuable.**

Dataset

Next to this document you'll receive `tweets_dataset.json`. This file is an array with several objects, each representing a tweet's metadata. Extract only the fields needed for inference and indexing.

FastAPI Routes

In this exercise we will be testing the creation of an API. This simple API will contain several endpoints. Each with a clear task. All must return a JSON response.

/emotion: **Tag emotions & store in Elasticsearch.** Reads tweets (optionally filtered by date), classifies *emotion*, then upserts docs into the tweets index in Elasticsearch.

/stance: **Tag stances & store.** Reads tweets, runs the TinyLlama prompt to assign *stance*, then upserts each doc's stance field in Elasticsearch.

/tweets: **Retrieve tweets.** Returns docs from Elasticsearch with emotion and stance. It should support pagination and date filters.

Each of the 3 endpoints requires a certain use of software architecture to be maintainable and usable. We intend that you keep in mind that **clean code is good code.**

/emotion and /tweets are mandatory endpoints 🚨

Each endpoint should receive different parameters, manage how the endpoint receives them as you consider best. The ones that we would like to see are:

limit: Max tweets to process in this call.

start_date: filter base on the field created_at in the dataset

end_date: filter base on the field created_at in the dataset

Models to Use

Our personal recommendation is for you to use the following models in each of the endpoint for the transformations. In the case of emotion, **it is mandatory** to use that one. In the case of the stance LLM model **we recommend** that small one that can be run locally without any GPU, even though it could bring some hallucinations.

1. Emotion Classification (Mandatory)

Model: j-hartmann/emotion-english-distilroberta-base

Labels: anger, disgust, fear, joy, sadness, surprise, neutral

Ref: https://huggingface.co/j-hartmann/emotion-english-distilroberta-base

2. Stance Detection (Optional, but extremely valuable)

Model: TinyLlama/TinyLlama-1.1B-Chat-v1.0 (~ 450 MB, CPU-only)

Prompt:

```
Tweet: {tweet}.

Does the author SUPPORT, OPPOSE, or remain NEUTRAL in its opinion?

Only return any of the three options.
```

Feel free to tweak the prompt for clarity/performance. Explain why you did it.

Ref: https://huggingface.co/TinyLlama/TinyLlama-1.1B-Chat-v1.0

Elasticsearch (tweets index)

To set up Elasticsearch, use the commands below to run a local instance with security disabled.

1. Start Elasticsearch locally in Docker

```
docker run --name es-demo -d \
  -p 9200:9200 -p 9300:9300 \
  -e "discovery.type=single-node" \
  -e "xpack.security.enabled=false" \
  docker.elastic.co/elasticsearch/elasticsearch:8.8.1
```

Note: This is a tip. You can do it as you like.

2. Install the Python client in the FastAPI project's .venv.

```
pip install elasticsearch==8.8.1
```

Note: Remember: this **must use** Python 3.11.

3. Tips

Use the tweet's numeric `id` as the Elasticsearch document `_id`.
Define a concise mapping in the index—store only what you need.

Deliverables

You should provide us with:

A **GitLab** public repository containing the project and a clear step-by-step README.md on how to set up and run the program.

We would need any requirements.txt file in the project to install any dependencies you have used.

Provide a file with the .env variables you used, if any.

EVERYTHING SHOULD RUN LOCALLY!

Good luck! We're excited to see your work.