Homework 1

By Luca Comba

To make it easier I created a Enum for all the classification strategies. I have then created a new Topic pydantic class. I have created a utility folder where I could store shared Exception classes and a utility library for reading and modifying any json file under the data directory.

I have then created a new `EmailSimilarityInferenceService` to inferr the similarity based on stored emails, and created a `InferenceServiceFactory` to enable a switch between the two services (similarity vs topic) based on the strategy selected.

New Endpoints

I implemented and updated the following API endpoints:

- `POST /emails`

This enpoint now stores a new email in the `data/emails.json`. The endpoint also accepts `subject`, `body`, and a optional `topic`. It then returns a message and the generated UUID email id.

- `POST /emails/classify`

This endpoint now classifies an email using a selectable strategy. The strategies can be `topic` (default), which compares the input email against topic descriptions, or `nearest` to finds the most similar labeled stored email and uses its topic.

- `POST /topic`

This endpoint is for adding a new topic. It takes a `name`, `description` as a json body request and saves it into the correct `topic_keywords.json` file

New Class Inference

As previously mentioned, I wrote a `InferenceServiceFactory` to select the correct service (`topic` or `nearest`) based on request strategy.

The `EmailTopicInferenceService` handles topic-description based classification only, while the `EmailSimilarityInferenceService` handles nearest-email classification only.

This keeps each class focused and makes it easier to extend with more strategies later.

Testing

I added/updated tests for the new behavior under the `tests` directory. Run `make test` to run all the pytests.

I have created a `test_classify.sh` to run a full manual API flow.

These output show that it was able to correctly add a new topic, add a new email.

To test the classification based on the `nearest` strategy, I have writted an email taking about the weekend and tagging it as `personal`. I then tried to classify a new email where I have mentioned the weekend. The model correctly predicted the email to be a personal email.

The `test_classify_output.txt` file contains all of the API responses and demonstrates the results of the classification.