11791 Homework 5 Report

Splitter Module Design

I used the starter code `Splitter/service.py` as the Splitter class implementation and followed the pattern of other service files for the Splitter service. The main idea of Splitter class is to tokenize the input question and store sentences and tokens as member variables of the `Question` and `Question.snippets`.

Ranker Integration

The main file I changed in the Ranker module is `CoreMMR.py` and `SimilarityJaccard.py`. In the file `SimilarityJaccard.py`, I added another function `calculateSimilarityTokens`, which instead of taking two questions, takes two token sets. Then in the file `CoreMMR.py`, I replace parts calling `calculateSimilarity` to call `calculateSimilarityTokens`, and made changes to input data type accordingly.

Other Changes

In order to support different hosts other than localhost for RabbitMQ server, I changed the code of `deiis/deiis/rabbit.py` to set `MessageBus(host=os.environ.get("RABBIT_HOST"))` so that it will get the host address from the environment variable. If running locally, RABBIT_HOST can be set as 127.0.0.1. If running in docker, it should be set as `172.17.0.2`. If the whole service is deployed on the cloud platform (AWS, GCP, etc.), the `RABBIT_HOST` in both cases can be set as the external IP/DNS.

I also added the Splitter module into the start and stop script to ease service starting and stopping process. It should also be added to the pipeline so that when a question arrives, it is first sent to the splitter module to get tokens generated.

Docker Image Deploy

I deploy the whole BioASQ software as one docker image. It is in the repository `hw9603/whole:FINAL`. Since the username cannot contain non-alphanumeric characters, I cannot set the username to be `bioasq-wenhe` as required.

How to run the pipeline

```
git clone https://github.com/hw9603/BioASQ-Rabbit.git
cd BioASQ-Rabbit
export RABBIT_HOST=$(RABBIT_HOST)
docker run -t -d -e RABBIT_HOST=$(RABBIT_HOST) -v
/path/to/data:/root/data hw9603/whole:FINAL /bin/bash
# In the local shell
python pipeline.py data/training.json
# WARNING: the directory to save is the one in the docker!
```

```
python save.py /root/data/submission.json
# After done, stop the service
python stop.py
# We can find the submission.json file in data/
ls data/submission.json
```

Since I deploy all my service on AWS EC2 server, the RABBIT_HOST in both local and docker is the external IP address of the EC2 server (ec2-*.compute.amazonaws.com).