COMP4703 Natural Language Processing

Project Assignment 2, Semester 2, 2024

Marks : This assignment is worth 25 marks (25% of the overall assessment for this course).

Due Date : Friday 11 August 2024, 3:00PM (Week 11), via `blackboard`.

Objective

The key objective of this project is to experimentally study the various performance trade-offs in Retrieval Augmented Generation Systems (RAG) when solving the "Multi-Hop Retrieval" Problem, which is essentially a factoid question answering problem that requires more than one document to get the correct answer, thus the requirement to retrieve documents first, and the use a generative algorithm to synthesise the final answer using the top-k documents retrieved.. There are two major stages in a RAG system. These include a first stage retrieval that finds the top-k most relevant documents. Then, a Large Language Model (LLM) is used to generate an answer to the question (query) input using these top-k documents. The dataset is a small factoid question answering dataset for Multi-Hop QA, so the system will either answer the question succinctly based on the instructions in a carefully crafted instruction prompt, or the system will reply that there is "Insufficient Data" to answer the question. To improve the effectiveness, an additional stage can be included in the middle to rerank the first stage results. While the idea is simple to understand, there are many possible choices that can be made in any of the three stages to significantly change the quality of the the results produced. This problem also demonstrates that complex system interactions can be orthogonal. That is, the "best" first stage and "best" second stage may not produce the "best" overall results when combined. There are many subtle interactions between the two models used that can be very unpredictable. So, the only way to determine the best overall system is through a lot of experimentation.

Your goal is to explore several different alternatives for each of the stages and write a five page lab experimental report that explains how each of the methods you have used work, and an exhaustive experimental study that compares and contrast the approaches you have used in order to find the best overall system.

Provided files

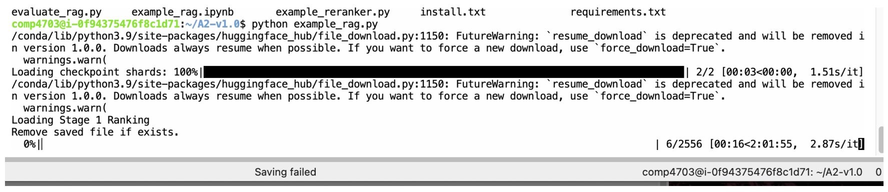
The following files have been provided to help you easily get started.

(1) **A2Spec-v1.0.pdf**: Version 1 of this specification.

(2) **README.md**: A brief description of how to recreate the python environment setup on the GPU in the event that you want to setup one on your own computer to code and stage.

(3) **data/corpus.json**: The full corpus collection file.

(4) **data/sample-corpus.json**: A small sample of the relevant plus a few negative samples for the handful of queries in `sample-rag.json` that is used for staging and testing..



```
evaluate_rag.py     example_rag.ipynb       example_reranker.py     install.txt             requirements.txt
comp4703@i-0f94375476f8c1d71:~/A2-v1.0$ python example_rag.py
/conda/lib/python3.9/site-packages/huggingface_hub/file_download.py:1150: FutureWarning: `resume_download` is deprecated and will be removed i
n version 1.0.0. Downloads always resume when possible. If you want to force a new download, use `force_download=True`.
  warnings.warn(
Loading checkpoint shards: 100%|████████████████████████████████████████| 2/2 [00:03<00:00,  1.51s/it]
/conda/lib/python3.9/site-packages/huggingface_hub/file_download.py:1150: FutureWarning: `resume_download` is deprecated and will be removed i
n version 1.0.0. Downloads always resume when possible. If you want to force a new download, use `force_download=True`.
  warnings.warn(
Loading Stage 1 Ranking
Remove saved file if exists.
  0%||                                                                                        | 6/2556 [00:16<2:01:55,  2.87s/it]
```

Figure 1: Estimated execution time of `example_RAG.py`.

(5) **data/rag.json**: The full set of queries and gold labels.

(6) **data/sample-rag.json**: A set of 5 queries and associated gold labels sampled from the original set that is used for staging and testing.

(7) **example_retrieval.py**: A complete example of a dense first stage retrieval. Easily updateable to several other algorithms.

(8) **example_retrieval.ipynb**: A Jupyter notebook of the file above with one key difference, the addition of the async loop so that it will work on Jupyter.

(9) **example_reranker.py**: An example of dense retrieval and reranking to improve the quality of the candidate set.

(10) **example_reranker.ipynb**: A Jupyter notebook of the file above with one key difference, the addition of the async loop to make it work on Jupyter.

(11) **evaluate_stage_1.py**: A very simple evaluation script for top-k retrieval.

(12) **evaluate_stage_1.ipynb**: A Jupyter notebook version of the file above.

(13) **example_rag.py**: An LLM based Retrieval Augmented Generation (RAG) example using Llama 2 7B to solve the "Multi-Hop Retrieval" problem.

(14) **evaluate_rag.py**: A simple evaluation of RAG that computes Recall, Precision, and F1.

(15) **evaluate_rag.ipynb**: A Jupyter notebook of the file above.

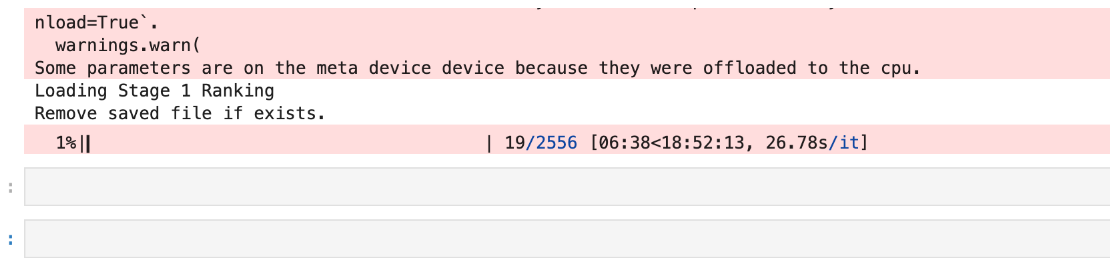
(16) **llamaindex-tutorial.md**: A markdown version of a pretty good tutorial on the Llama Index API generated from here[1].

(17) **requirements-cpu.txt**: The `requirements.txt` file for non-GPU computers.

(18) **output/**: The directory where all of the generated files will be written.

Note that there is *no* Jupyter notebook version for RAG. I could not release it since the runtime is an order of magnitude worse than the python script released. In fact the running time is so bad, that it is unusable and I have no idea why. If I can get it to work, I will release it, but for now, you must assume that it will not be released and that you will need to work with the script. You are free to use entirely different approaches than Llama Index if you wish, and if you can get them to run in 2 hours or less, they are usable on the GPU. If they have significantly longer running time, then you will exceed your GPU quota. Figure 1 and Figure 1 show the huge difference in runtime between the python script and Jupyter notebook version of RAG, showing that the use of the notebook is not viable.

[1] `https://nanonets.com/blog/llamaindex/?source=post_page-----919313a4e822--------------------------------`



```
nload=True`.
  warnings.warn(
Some parameters are on the meta device device because they were offloaded to the cpu.
Loading Stage 1 Ranking
Remove saved file if exists.
  1%||                                   | 19/2556 [06:38<18:52:13, 26.78s/it]
```

Figure 2: Estimated execution time of `example_RAG.ipynb`.

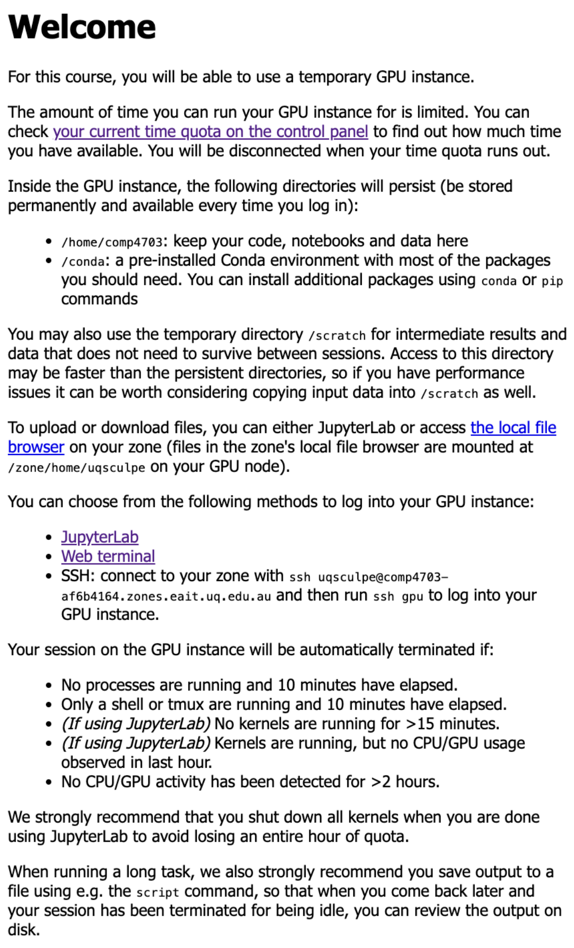
Additional Details

Despite the collection being small and the availability of GPU resources, you *need* to plan and use your time wisely. The average runtime of a dense retrieval model is an hour on the GPU, reranking is about 1.5 hours, and text generation from the results requires around 2 hours. You are expected to have 4 different algorithms for stage 1 retrieval and 4 different LLM text generations in order to provide enough data for an interesting comparison. I have given you 2 first stage (ranking and reranking) and one LLM RAG. So if we assume $\approx 2 \times 4 + \approx 2 \times 4$, your experiments, once staged properly, will require at least 16 hours to run. It should not be terribly difficult to get a good variety of models stages and running with a a half day of coding effort. You are of course free to get creative and use frameworks other than `llama-index` if you wish, but the entire assignment can be done using this API. It is very flexible and can support at least 10 stage one and 10 RAG stage models with only a few changes. I was able to stage about 12 over the weekend (but of course it will take me a week to run them all) It will take you a couple of days to collate all of your experimental results and write your final report. **Use your time wisely and plan ahead.**

Accessing the GPU

There are many caveats to using the GPU cluster provided. You should never have to wait, however the caveat is that the timeout time is aggressive to ensure that you do not use more time on the GPU than you need. The entry point is:
`https://coursemgr.uqcloud.net/comp4703`.
Figure 3 shows the entry page. You can login externally or internally using Duo authentication. Once you login, a Jupyterhub instance will start in your own zone that also has a terminal, or you can run a terminal directly from the entry page.

Another caveat when using Jupyter is that each notebook will use a significant amount of the GPU RAM, and it does not seem to go away unless you manually kill the process or stop the kernel in the notebook. This is demonstrated in Figure 4. These *may* eventually be killed by a timeout, but in my experience they don't go away and eventually make the system unusable. If you open a terminal and run the command `nvidia-smi` you can see the GPU usage. In the Figure, you can see PID 23869 and 23915 that are using 18 of 24 GB of GPU RAM. You can manually kill them using `kill -9 23869 23915` (in my example). This will remove the kernels from the GPU and everything will work normally again. The key point is that you have to pay attention to you environment when using Jupyter. If you use the terminal with `tmux`, this will never be an issue. There is no magic we can do to make this work



Welcome

For this course, you will be able to use a temporary GPU instance.

The amount of time you can run your GPU instance for is limited. You can check your current time quota on the control panel to find out how much time you have available. You will be disconnected when your time quota runs out.

Inside the GPU instance, the following directories will persist (be stored permanently and available every time you log in):

- `/home/comp4703`: keep your code, notebooks and data here
- `/conda`: a pre-installed Conda environment with most of the packages you should need. You can install additional packages using `conda` or `pip` commands

You may also use the temporary directory `/scratch` for intermediate results and data that does not need to survive between sessions. Access to this directory may be faster than the persistent directories, so if you have performance issues it can be worth considering copying input data into `/scratch` as well.

To upload or download files, you can either JupyterLab or access the local file browser on your zone (files in the zone's local file browser are mounted at `/zone/home/uqsculpe` on your GPU node).

You can choose from the following methods to log into your GPU instance:

- JupyterLab
- Web terminal
- SSH: connect to your zone with `ssh uqsculpe@comp4703-af6b4164.zones.eait.uq.edu.au` and then run `ssh gpu` to log into your GPU instance.

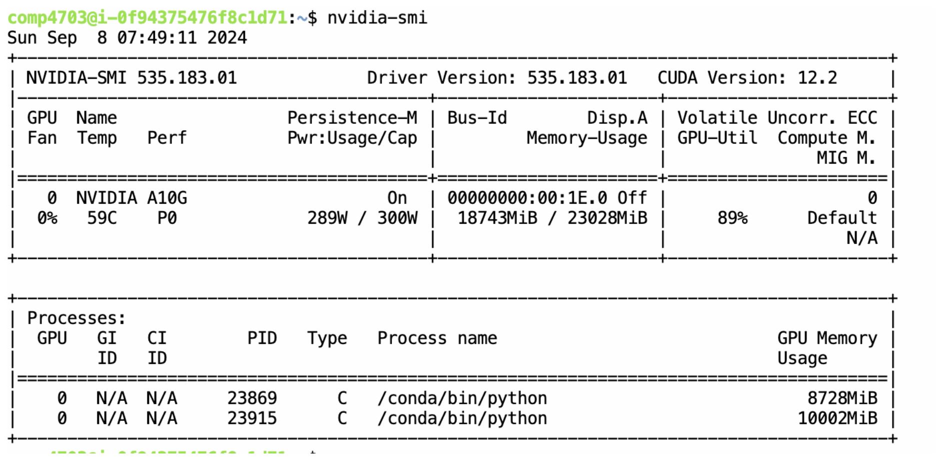
Your session on the GPU instance will be automatically terminated if:

- No processes are running and 10 minutes have elapsed.
- Only a shell or tmux are running and 10 minutes have elapsed.
- *(If using JupyterLab)* No kernels are running for >15 minutes.
- *(If using JupyterLab)* Kernels are running, but no CPU/GPU usage observed in last hour.
- No CPU/GPU activity has been detected for >2 hours.

We strongly recommend that you shut down all kernels when you are done using JupyterLab to avoid losing an entire hour of quota.

When running a long task, we also strongly recommend you save output to a file using e.g. the `script` command, so that when you come back later and your session has been terminated for being idle, you can review the output on disk.

Figure 3: Entry page to the GPU instance. Note the two links, one to a terminal and one to Jupyter.



```
comp4703@i-0f94375476f8c1d71:~$ nvidia-smi
Sun Sep  8 07:49:11 2024
+---------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.183.01             Driver Version: 535.183.01   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA A10G                    On  | 00000000:00:1E.0 Off |                    0 |
|  0%   59C    P0             289W / 300W |  18743MiB / 23028MiB |     89%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
|    0   N/A  N/A     23869      C   /conda/bin/python                          8728MiB |
|    0   N/A  N/A     23915      C   /conda/bin/python                         10002MiB |
+---------------------------------------------------------------------------------------+
```

Figure 4: Example of `nvidia-smi` output showing two dead kernels on the GPU. If you try to run a third notebook, it will throw an out of memory error.

cleanly. So the choice is yours, use Jupyter if you must, but be prepared for a number of limitations.

I realise this might all sound terrifying, but it is the reality of running AI models on remote GPU servers. It requires patience and experience to get everything to work. While we are using a pretty small dataset this time, LLM model are always a challenge. They require a lot of GPU memory, and even small tasks can take a long time to run. So, the best way to be successful is to make sure your code works as soon as possible, and learn how to write bash scripts to batch process multiple tasks, and check back periodically. If you included code to print out information at various stages of the process, you can easily know if everything is working as expected or if there is a problem. If you know a job will take 6 hours to run, you should only have to check it once, or run it just before bed and check on it in the morning. If you are really good, you would run 4x6 hour jobs in one script. If you check on it once a day over 2-3 days, all of your batch processes get ran in one go! But that requires extra time up front to stage and script the batch process.

The point is, do what works best for *you* and you abilities. There is not one way to accomplish this goal, but I believe you are going to struggle do this project in Jupyter unless you are an expert user. The command line approach was very reliable for me in my initial testing, albeit slower to get setup and working.

If you wish to install new libraries, you are free to do so. Just use the base environment as it is a one off setup for this assignment.

One final note, I have made the zip file available on an external webserver so that you can easily get a copy of the assignment to the GPU workspace. To download the file, open a terminal on the GPU and run the command:

`wget http://wight.seg.rmit.edu.au/jsc/A2-v1.0.zip`

If additional updated versions are made available on Blacboard, I will also make them available from this URL if you need it. You can also move files to and from the GPU instance in a number of other ways. This was the easiest way *for me*,

Marking Rubric

- Report (20 marks)
  - Writeup
    - Introduction (1 mark)
    - Methodology (2 marks)
    - Experiments (3 marks)
    - Conclusions (1 mark)
    - References (1 mark)
  - Other Factors
    - Clarity of Writing (2 marks)
    - Sufficient number of algorithms compared (4 stage 1 + 4 LLM gen) (3 marks)
    - Sufficient number of evaluation metrics (2+ from you) (1 marks)
    - Proper use of visuals (Tables, Diagrams, Figures, etc. – 3+ from you) (2 marks)
    - Comprehensive compare and contrast analysis (3 marks)
    - Format (length) (1 mark)
- Environment (5 marks)
  - Complete and working requirements.txt (1 mark)
  - All code artefacts (2 marks)
  - All intermediate results (2 marks)

I will include an example of a comparison paper I published a few years ago in a top conference on Blackboard. While I do not expect your report to be at this level, it does provide you with an example of what a comprehensive compare and contrast experimental report looks like. It should include Figures, Tables, Diagrams, etc. along with a discussion of each. It should include insights and even a failure analysis of why System A performs better / worse than System B. The goal is to demonstrate that you understand how the algorithms you use work, and can clearly articulate their strengths and weaknesses. This is how good research (and good engineering) should work. No one puts something in production that thousands or even millions of customers pay to use unless they are confident that it is the best system for the task, and you can't know that without a lot of candidate testing and comparison.

Submission

All submissions must be made through Blackboard. A link will be provided within Blackboard, under the Assignments tab for submissions. To submit your files, create a directory with your student number just like last time. For example, `mkdir S101010` for the assignment. Place all relevant files in this directory (no additional subdirectories), and use `zip` to archive the files. **Do not use** `rar`. Please include the directory as part of the archive. For example, if I unzip the file `S101010.zip`, I would see the following:

```
$ unzip S101010.zip
Archive:  S101010.zip
   creating: S101010/
  inflating: S101010/report.pdf
  inflating: S101010/rankerA.py
  inflating: S101010/rankerB.py
  inflating: S101010/rankerC.py
  inflating: S101010/rerankerA.py
  inflating: S101010/rerankerB.py
  inflating: S101010/RAGA.py
  inflating: S101010/RAGB.py
  inflating: S101010/RAGC.py
  inflating: S101010/RAGD.py
  inflating: S101010/MyRAGEval.py
  inflating: S101010/MyRetEval.py
  inflating: S101010/MyREADME.md
  inflating: S101010/requirements.txt
  inflating: S101010/output/rankerA.json
  inflating: S101010/output/rankerB.json
  inflating: S101010/output/rankerC.json
  inflating: S101010/output/rankerD.json
  inflating: S101010/output/rerankerA.json
  inflating: S101010/output/rerankerB.json
  inflating: S101010/output/RAGA.json
  inflating: S101010/output/RAGB.json
  inflating: S101010/output/RAGC.json
  inflating: S101010/output/RAGD.json
```

The following files should be submitted (Please do **not** submit any test files):

- `MyREADME.md` - A simple text document containing any instructions to run your code and documenting any changes to your environment that you made.
- `report.pdf` - The 5 page experimental report.
- `requirements.txt` - Generated using `pip freeze > requirements.txt` to ensure that your result can be reproduced if necessary.
- `ranker[A-Z].[py/ipynb]` - The different retrieval baselines you used.
- `reranker[A-Z].[py/ipynb]` - Any reranking baselines you used.
- `rag[A-Z].[py/ipynb]` - The different LLM text generation baselines
- `output/ranker[A-Z].json` - The different retrieval baseline results.
- `output/reranker[A-Z].json` - Any reranking baseline results.
- `output/rag[A-Z].json` - The different LLM text generation baseline results.
- `evalXX.[py/json]` - If you added additional evaluation metrics, include the code so that it can be reproduced.

NOTE 1 : Include code, scripts, and intermediate results for all of your baselines and include a complete requirements.txt file to ensure that your experiments can easily be reproduced. Code can be in the form of scripts, Jupyter notebooks, or a combination of both, along with any batch automation scripts if you wrote any.

Plagiarism Warning

University Policy on Academic Honesty and Plagiarism: It is University policy that cheating by students in any form is not permitted, and that work submitted for assessment purposes must be the independent work of the student concerned. Please see the UQ policy for more details: `https://policies.uq.edu.au/document/view-current.php?id=149`. **THIS IS NOT A GROUP PROJECT.** Students are reminded that this assignment is to be attempted individually. Plagiarism of any form will result in zero marks being given for this assessment, and can result in additional disciplinary action. We routinely use plagiarism software on projects! Please, please do your own work.

Extension Policy

Individual extensions must go through standard channels at the University of Queensland. I am not supposed to grant extensions directly. If you have suffered a personal tragedy or illness, there is a University process in place to grant extensions and alternative assessment.

Getting Help

The head tutor Xin Xia is available if you contact her. I will hold office hours every Tuesday after the lecture. I will also try to stop into some of the Wednesday Labs to see if there are any questions. In addition, the you are encouraged to discuss any issues you have with your Lab Demonstrator, email me directly, or ask on Ed. We are here to help and we want to see you succeed. This is a "real-world" state-of-the-art problem that I think is very interesting.