Practical Assignment

The goal of the assignment is to implement a program that improve the performance of large language models (like ChatGPT, etc.). To this end, your solution is asked to 1) parse the output of a language model and to extract a clean answer and 2) check, with the aid of existing knowledge bases, which are available on the web, whether the answer is correct or not.

The assignment should be done in groups of four students.

The deadline for submitting the assignment is **22/12/2023 at 23:59PM.** You are supposed to submit:

- all the code and relevant documentation to replicate it
- a short report (max 2 pages) describing your solution, motivating the most important design choices, and with an empirical evaluation on test data of your choice (if you decide to create your test test, then you also need to submit a copy of your data)
- a link to a video with a short presentation of 10 minutes that describes your solution

You are free to use whatever language or technique you want to implement the assignment, provided that you report clear instructions on how to launch your program. Most students decide to use Python, which is the language I would also advice to use.

Before sending emails to the teacher, check the section "Frequently asked questions" at the end of this document.

Instructions

You are called to implement a program that receives as input a question (or more in general a text to be completed), which we henceforth call **A** and returns as output **four** things:

1. The raw text returned by a large language model (that is, what you would get if you query the language model as is). We call it **B**
2. The answer extracted from **B**. This answer can be of two types: either **yes/no** or a **Wikipedia entity**
3. The correctness of the extracted answer (correct/incorrect)
4. Entities that have been extracted from **A** and **B**

The example below describes what you are supposed to return:

Example:

Suppose that you want to know whether the capital of Nicaragua is Managua. Then the text you can provide to the language model can be either "is Managua the capital of Nicaragua?" or "The capital of Nicaragua is ...". In the first case, you are asking the language model to return an yes/no answer while in the second case it should return the name of a city.

Yes/No case:

Input (A): "is Managua the capital of Nicaragua?"

Text returned by the language model (B) (llama 2, 70B): "Yes, Managua is the capital city of Nicaragua. It is located in the southwestern part of the country and is home to many important government buildings and institutions, including the President's office and the National Assembly. The city has a population of over one million people and is known for its vibrant cultural scene, historic landmarks, and beautiful natural surroundings."

Extracted answer: "yes"

Correctness of the answer: "correct**"**

Entities extracted:

Nicaragua<TAB>https://en.wikipedia.org/wiki/Nicaragua

Managua<TAB>https://en.wikipedia.org/wiki/Managua

President's office<TAB>https://en.wikipedia.org/wiki/President_of_Nicaragua

National Assembly<TAB>https://en.wikipedia.org/wiki/National_Assembly_(Nicaragua)

The example above was quite simple, especially because we used one of the best language models (llama 2 with 70B tokens). If we use a smaller language model, like the version of llama 2 with 7B tokens, then the output returned by the model would be:

Text returned by the language model (B) (llama 2, 7B): "What is the capital of Nicaragua?
Managua is the capital of Nicaragua.
What is the capital of Nicaragua and its population?
Managua is the capital of Nicaragua. The population of Managua is 1.3 million people.
Is Managua the capital"

As you can see, it is harder (although not impossible) to detect that the extracted answer is "yes". In this case, the output should be:

Extracted answer: "yes"

Correctness of the answer: "correct**"**

Entities extracted:

Nicaragua<TAB>https://en.wikipedia.org/wiki/Nicaragua

Managua<TAB>https://en.wikipedia.org/wiki/Managua

If we were to change the input to "the capital of nicaragua is..." then we are requested the model to return an entity rather than yes/no. In this case, the returned text would be:

Input (A): "the capital of nicaragua is..."

Text returned by the language model (B) (llama 2, 7B): "Prior to 1979, Nicaragua was known as the Republic of Nicaragua. It is a republic with a presidential system of government. The capital of Nicaragua is Managua.
The capital of Nicaragua is Managua.
What is the capital of nicar**"**

and your expected output should be

Extracted answer: https://en.wikipedia.org/wiki/Managua

Correctness of the answer: "correct**"**

Entities extracted:

Nicaragua<TAB>https://en.wikipedia.org/wiki/Nicaragua

Managua<TAB>https://en.wikipedia.org/wiki/Managua

Clearly, the correctness of the answer depends on the extracted answer. If you fail to extract that the capital is Managua, then your fact checking routine should return that the answer is incorrect. Of course it could also be that the language model is incorrect. See the following example:

Input (A): "Managua is not the capital of Nicaragua. Yes or no?**"**

Text returned by the language model (B) (llama 2, 7B): "Most people think Managua is the capital of Nicaragua.
However, Managua is not the capital of Nicaragua.
The capital of Nicaragua is Managua.
The capital of Nicaragua is Managua. Managua is the capital of Nicaragua.
The capital"

Extracted answer: no

Correctness of the answer: "incorrect**"**

Entities extracted:

Nicaragua<TAB>https://en.wikipedia.org/wiki/Nicaragua

Managua<TAB>https://en.wikipedia.org/wiki/Managua

Docker image

To help the development, I've created a Docker image with a small example that shows how you can interact with a language model. As language model, I've loaded in the image the llama2 model with 7B parameters, compressed with a 4bit quantization. This is the smallest model in the llama2 family. You can easily install a larger model if you want, provided you have enough computing resources. To download the docker image, please type the following command:

```
docker pull karmaresearch/wdps2
```

To run the docker image, type the following

```
docker run -ti karmaresearch/wdps2
```

This will open a shell prompt that will allow you to run some python code.

Submission

You need to submit the code and detailed instructions on how to run your code. Make sure that your program works with the Docker image that I provide (karmaresearch/wdps). If your program requires additional libraries, please write them down. I'm also fine if you create a new docker image with everything already installed.

To make the grading process more automatic, I ask you to provide your output in a specific format. Basically, you should expect that the input of your program is a text file of the form <ID question> <TAB>text of the question/completion<newline>

For instance, it could be: question-001<TAB>Is Managua the capital of Nicaragua?<newline>

Your output should be a file where the answer are in the following format

<ID question><TAB>[R,A,C,E]<answer> where "R" indicates the raw text produced by the language model, "A" is the extracted answer, "C" is the tag correct/incorrect and "E" are the entities extracted. For instance, for the example above the output should be:

question-001<TAB>R"Most people think Managua is the capital of Nicaragua.
However, Managua is not the capital of Nicaragua.
The capital of Nicaragua is Managua.
The capital of Nicaragua is Managua. Managua is the capital of Nicaragua.
The capital"<newline>

question-001<TAB>A"no"<newline>

question-001<TAB>C**"**incorrect**"**

question-001<TAB>E"Nicaragua"<TAB>"https://en.wikipedia.org/wiki/Nicaragua"

question-001<TAB>E"Managua"<TAB>"https://en.wikipedia.org/wiki/Managua"

During the grading, I will launch your program using the Docker image that I've provided on some datasets that I've created for the grading. If your program does not execute or return the data in a format that is different to the one that I've reported above, then points will be detracted. So make sure that your program is complaint with the specifications.

Code and report have to be submitted on Canvas before the deadline. If you are late, you will receive 1 point less if you submit within 24 hours or 2 points if you submit within 48 hours. After 48 hours, your assignment will get 0 points.

The report should not be longer than two pages and the presentation should not be longer than 10 minutes.

Grading policy

Your submission will be graded with the following criteria:

- **Originality (5%):** Did you implement an interesting and sophisticated solution?
- **Performance on entity disambiguation (10%):** Was your program able to recognize and link entities/relations to Wikipedia?
- **Performance on answer extraction (10%):** Was your program able to extract an answer (yes/no or entity) from the completions returned by the language model?
- **Performance on the fact checking (10%):** Was your program able to determine accurately whether the answer returned was true or false?
- **Code quality (15%):** Is your code readable and clear?
- **Documentation (10%):** This point should be self-explanatory
- **Compliance (15%):** Does your program return the output properly?
- **Scalability (15%):** Is your program capable to perform knowledge extraction on large inputs (e.g., using parallelism)?
- **Presentation (10%):** Is the presentation clear? Are all members presenting? Does it show the program running?

Depending on the quality of the submission, I reserve the right to change the percentages and/or add more criteria. Also, group members may receive different grades if I discover that the workload was uneven. In principle, any group member should be able to describe every aspect of the assignment.

Frequently Asked Questions

1) Do I have to consider text written in multiple languages

No, you can assume that we will only work with English text.

2) Which language should I use for the linking to Wikipedia?

Please use the English version for the links.

3) Can I use APIs to language models (like ChatGPT) to obtain the completions?

Yes, but keep in mind the following. 1) If you use an external language model, I may not be able to test it since I don't have access (nor financial resources) to access the service that you used. Moreover, services like ChatGPT already do some of the processing that you are asked to implement. Therefore, you will receive a lower score on **Performance** (because some of the work is already done by the service), **Scalability** (because accessing an external service is not scalable), and **Originality** (because your solution will be simpler.

4) Can I use external libraries to do the entity disambiguation?

No, because those libraries do what you are supposed to do. You can use libraries that do the entity **recognition** but not the **disambiguation.**