

Few-shot Learning with LLMs

Adapted from Carloyn Anderson's CS 333: Natural Language Processing at Wellesley College.

Install gradio

activate nlp and then run:

```
pip install gradio
```

Using LLaMa

You will work with a large language model called LLaMa, which was developed by Meta and released publicly. I am running this model on my research server. I have given you a Python file called `query_llama.py`. This provides a function, `generate`, that sends a query request to LLaMa.

Pig Latin

In pig latin, if the word starts with a consonant, you move it to the back of the word and append "ay". If it starts with a vowel, you simply append "ay."

Can LLaMA break this code?

Dataset creation

Hand create a dataset of pig latin words. Try to include a variety of words (low frequency, high frequency, words starting with vowels, etc.).

Few-shot learning

Next, write a program that tests how well a large language model performs on decoding pig latin. Your program should import and use the `query_llama.py` program that I have provided. This file contains a function for querying LLaMa. Your task is to craft prompts to feed to the model so that it solves the pig latin decoding task. `pig_latin_probe.py` is provided as a scaffold.

Prompt Engineering

Start by writing a function called `make_prompt`. This function should take in a pig latin word and prepare a prompt. You should try out different formats for prompts. Try at least the following:

- Describing the task in different ways
- Adding different numbers of examples to the prompt
- Formatting the examples in different ways

Query model

Next, write a wrapper for the model query function called `query_model`. Your function should take in a prompt and call `query_llama.py` to get the model's prediction. Unless you are interested in tinkering with the model hyperparameters, you can call the `generate` function from `query_llama` on a single prompt as follows:

```
query_llama.generate(prompt)
```

Evaluation

When you run a prompt through the LLM, you will need to do some additional string processing to get back an answer. Sometimes, the model will go on to produce many examples rather than just completing the single task that you set. Write a function called `get_answer`. Your function should take in the response from the LLM. It should post-process the response and return a single answer as a string.

Running the experiment

Write a function called `run_one_prompt` that combines the functions that you have written above. It should take a pig latin word, its decoded answer, and return 1 or 0 based on whether the LLM successfully decoded the word.

Now you can finish the main function for your program. Your main function should read in the test items and labels from file. It should loop through the pig latin words, calling `run_one_prompt` on each word/label pair. It should sum up the scores returned by and print out the model's accuracy.