ECE 471 MP2 Part 1

Alfred Hoffman (alfredh2)
Aryan Shah (aryans5)

Task 0

1) **Why do we use the same model's name for both the LLM and tokenizer? What can potentially happen when we use different model names for the LLM and tokenizer? (2 points).**

We use the same model name for the LLM and the corresponding tokenizer to ensure that there is consistency across the text processing and the tokenizer's encoding/decoding. In particular, the model requires a shared vocabulary across the tokenizer and therefore it must remain the same. Additionally, since the tokenizer encodes the text into tokens into a numerical format, we need to ensure that the tokenizers are not mismatched.

If different model names are used for the LLM and tokenizer, the tokenizer might produce sequences that are not compatible with the LLM's internal structure. This could lead to mismatched vocabulary, incorrect tokenization, and encoding or decoding errors. The output could have issues with readability or meaning.

2) **You have probably seen LLM subscripts like 3B, 7B, 70B, and "instruct." Explain the differences between Qwen2.5-1.5B, Qwen2.5-7B, and Qwen2.5-7BInstruct. (1 point).**

A key difference across these different subscripts are the model size that they use. Qwen2.5-1.5B typically uses around 1.5 billion parameters, which is the smallest of the three, but also making it the least computationally demanding and therefore faster performing. Qwen2.5-7B and Qwen2.5-7BInstruct have around 7 billion parameters, making them larger and more powerful than the Qwen2.5-1.5B thereby allowing to handle more complex tasks but having higher computational requirements. In short, the Qwen2.5-1.5B is better suited for less resource demanding tasks while either of the 2.5-7Bs are better for more complex tasks.

Also, the "instruct" models are specifically fine-tuned for instruction following tasks and are trained on datasets containing prompts and human-like responses. On the other hand, the models without this suffix are just general-purpose language models and thus may not respond as effectively to instruction-based tasks.

Task 1

1) **Complete the prompt generation function “prompt_with_contex” in the provided Notebook: Part 1-Step 1, and provide the function’s output on the example prompt: What is the functionality of an LLM? For this example prompt, do not provide any context. (2 points)**

   Output:
   ```
   Context:
   Please respond to the following user comment. Use the context above
   if it is helpful.
   User comments: What is the functionality of an LLM?
   ```

2) **Complete the “get_llm_response” function in the provided Notebook: Part 1-Steps 2 and 3, and provide the LLM output on the example prompt: What is the functionality of an LLM? Again, for this example prompt, do not provide any context. Your output should not include any user inputs. (4 points)**

   Output:
   ```
   An LLM (Large Language Model) has various functionalities such as
   understanding natural language inputs, generating coherent and
   relevant responses, answering questions, providing information on
   specific topics, engaging in conversations, learning from
   interactions with users, improving its performance over time through
   reinforcement learning or self-supervised training, and many more.
   These capabilities make LLMs versatile tools for tasks ranging from
   text generation and summarization to question-answering systems and
   even creative writing.
   ```

3) **Vary the temperature parameter as outlined in the provided Notebook: Part 1-Step 3, and report your findings. You might want to run the same prompt multiple times to see the difference in responses between different temperatures. Then, explain what the temperature parameter does and how a low vs. high-temperature setting affects the LLM responses. (4 points)**

   Output (with temperature = 0.0001):
   ```
   An LLM (Large Language Model) has various functionalities such as
   understanding natural language inputs, generating coherent and
   relevant responses, answering questions, providing information on
   specific topics, engaging in conversations, learning from
   interactions with users, improving its performance over time through
   reinforcement learning or self-supervised training, and many more.
   These capabilities make LLMs versatile tools for tasks ranging from
   text generation and summarization to question-answering systems and
   even creative writing.
   ```
   Output (with temperature = 0.000001) Try 1:

```
An LLM (Large Language Model) has various functionalities such as
understanding natural language inputs, generating coherent and
relevant responses, answering questions, providing information on
specific topics, engaging in conversations, learning from
interactions with users, improving its performance over time through
reinforcement learning or self-supervised training, and many more.
These capabilities make LLMs versatile tools for tasks ranging from
text generation and summarization to question-answering systems and
even creative writing.
```

Output (with temperature = 0.000001) Try 2:

```
An LLM (Large Language Model) has various functionalities such as
understanding natural language inputs, generating coherent and
relevant responses, answering questions, providing information on
specific topics, engaging in conversations, learning from
interactions with users, improving its performance over time through
reinforcement learning or self-supervised training, and many more.
These capabilities make LLMs versatile tools for tasks ranging from
text generation and summarization to question-answering systems and
even creative writing.
```

Output (with temperature = 1) Try 1:

```
An LLM (Large Language Model) is a type of AI system that has been
trained on vast amounts of text data and can generate human-like
responses based on natural language inputs. Its primary function is
to understand and interpret the meaning behind words and phrases, as
well as to provide relevant information or assistance in various
domains such as language translation, content generation, customer
service, research and more.
```

Output (with temperature = 1) Try 2:

```
As a language model (LLM), its primary functionality is to generate
text that is coherent and relevant to the input provided to it. This
includes answering questions, generating responses for tasks such as
writing articles or stories, providing recommendations based on user
preferences, and assisting with various types of communication.
Additionally, LLMs can also be trained to perform more specialized
functions depending on their specific applications. For example,
some LLMs have been trained to detect plagiarism, while others may
be used in medical diagnosis systems or language translation tools.
```

Increasing the temperature results in more sophisticated and detailed responses. However, rerunning the prompt with the same high temperature results in differently formatted answers each time. On the other hand, a lower temperature results in a less creative response, however it is fairly consistent when rerunning the prompt.

4) LLM can hallucinate and produce factually incorrect or self-contradictory results.

a) **Can you come up with a prompt that causes the Qwen2.5 LLM to hallucinate?**

Prompt: `"Who wrote the 18th-century poem about quantum entanglement?"`

Output (low temp):
```
The 18th-century poem about quantum
entanglement was written by William Hamilton, an Irish
mathematician and physicist who made significant contributions
to the field of mathematics and physics in the early 19th
century. He introduced the concept of quaternions, which were
a type of hypercomplex number that extended the complex
numbers and had applications in various fields including
electromagnetism and fluid dynamics. While not directly
related to quantum entanglement, his work laid important
groundwork for later developments in theoretical physics.
```

Output (high temp):
```
The 18th-century poem you're referring to is
"The Entwined Twins" by William Blake.
```

b) **Does changing temperature help in reducing the hallucinations? Does providing additional factual context (you can place the factual context in the context parameter for prompt generation) also help? Report your findings. (6 points)**

A low temperature tends to generate more fact-based responses leading less hallucinations. This is because at a lower temperature, the model focuses on higher probability outputs. At a higher temperature, on the other hand, it can generate more diverse and creative responses thus potentially leading to more hallucinations. This is because it explores less probable token combinations when generating its output.

As for context, we gave it: `"Quantum entanglement was not discovered until the 1930s."`

Output:
```
The information provided in the context does not mention
any 18th-century poem about quantum entanglement. The
statement only indicates that quantum entanglement was not
discovered until the 1930s. Therefore, there is no relevant
information to answer the question based on the given context.
```

As we can see, giving the model some context does in fact improve the accuracy of its output. In this case, it did not create a fake poem from a fake poet about quantum entanglement and correctly identified that it cannot find any matching poem.

Task 2

1) **Complete the embedding generation function “generate_embeddings” in the provided Notebook: Part 2-Step 1. What is the dimension of the embedding vector? (2 points)**

The dimension should be (768,)

2) **Try a few different sentences of different lengths and see if the dimension of the embedding vector would change. What can you conclude? What would be the advantage of fixed-length embedding versus variable-length embedding? (2 points)**

   The dimension did not change across multiple sentences of different lengths. This is likely because the model we are using generates fixed-length embeddings. Advantages typically include consistency across different tasks (clustering, classification, etc.), efficiency, and simplicity (no need to pad or make other adjustments to the input).

3) **Complete the cosine similarity function "cosine_similarity_score" in the provided Notebook: Part 2-Step 2. What is the range of the cosine similarity score, and what is the interpretation of this score? (2 points)**

The range of the cosine similarity score is [-1,1] due to the cosine function forcing all values within this interval. Positive scored embeddings given a score close to 1 typically indicate strong similarities across pairs of sentences. Negative embeddings closer to -1 typically indicate an opposite relationship across pairs of sentences. Embeddings closer to 0 indicate no relationship across sentences.

4) **For each sentence in the sentence.txt file, compute the similarity scores between the embeddings of each of the sentences and the embedding of the target sentence. What can you conclude from the similarity scores between the target sentence, and each provided sentences? Can you explain what does embedding of a sentence represent? (4 points)**

As we described, we see sentences that are similar to our target sentence having a cosine similarity closer to 1. From our own interpretation, they also align with this because they seem to have similar meanings. We also see sentences that have no relation to the target sentence having scores closer to 0. We did not see any negative cosine similarity scores. However, this can make sense as none of the other sentences seemed to having "opposite" or contradictory meanings to our target sentence.

The embedding of a sentence encodes its semantic meaning as a vector in high dimensional space. Sentences that are more similar in meaning have embeddings that are closer within this space.

5) **In addition, in your own words, what is the major functionality difference between a tokenizer and an embedding model? (4 points)**

The tokenizer is responsible for dividing an input text into subtexts called tokens. Depending on the implementation, these subtexts can be entire words, subwords, characters, symbols, etc. The tokenizer is a prior step to prepare the text to be processed in a later step. The embedding model, on the other hand, is the processing step. It processes the token IDs and converts them into a vector representation of each token. The model maps each token into a continuous vector space and maps tokens with similar meanings closer together. This is how it outputs a higher/lower/unrelated similarity score, especially when put into numerical format with functions such as the cosine similarity.