NLP
Assignment N.3

January 30, 2025

Bar, Alon	Ben Tzvi, Nehoray	Baron, Lia
205476013	206389538	036635803

1 Medical RAG

a. When or why should you use RAG to access your data instead of fine-tuning?

Ans: Use Retrieval-Augmented Generation (RAG) instead of fine-tuning when:

- **Frequent Data Updates**: Fine-tuning requires retraining; RAG dynamically retrieves the latest data.
- **Private or Real-Time Data**: Fine-tuning bakes data into the model, while RAG accesses it securely at runtime.
- **Lower Cost and Compute**: Fine-tuning is expensive; RAG avoids retraining large models.
- **Source Attribution**: RAG enables citations, whereas fine-tuning does not.
- **Controlled Knowledge Access**: RAG ensures the model only retrieves from specified sources.

Use fine-tuning instead when:

- **Static Data**: If knowledge is stable and fundamental, fine-tuning internalizes it.
- **Generalization Needs**: Fine-tuning improves reasoning beyond specific retrieved documents.
- **Faster Inference**: RAG introduces retrieval latency, while fine-tuning is immediate.
- **Higher Fluency and Coherence**: Fine-tuning produces more natural responses.

b. Follow the instruction and complete the Retriver section. Load 'BAAI/bge-base-en-v1.5' embedding model from hugging face and initialize FAISS database (name it KNOWLEDGE VECTOR DATABASE).

Ans:

Done in the collab notebook attached.

c. Follow the instruction and complete the Reader section. Load 'HuggingFaceH4/zephyr-7b-beta model using BitsAndBytesConfig' as done in the turturial (name it model) and it's tokenizer.

Ans:

Done in the collab notebook attached.

d. Follow the instruction and complete the Putting it all together section. Define the rag-chain using langchain RunnableLambda, so it will get the query question and options and the context from the retriever.

Ans:

Done in the collab notebook attached.

e. Follow the instruction and complete the Visualization section. Visualize the search for the closest documents, of example key, using PaCMAP (similar to Advanced RAG turturial)

PaCMAP is high-demntionality visualization technique. We already seen example of using PCA for visualization in HW1 Q2. But there are many different visualization algorithms, such as t-SNE, UMAP and PaCMAP. Each algorithm preserve different properties of the data in the low demission. Generally speaking, t-SNE and UMAP preserve more local topology then PCA. (Fill free to read more about it.)

Group according to the textbooks titles. What can you tell from the visualization? Include the question you visualize the search for and the plot in your PDF, as well as your full answer.

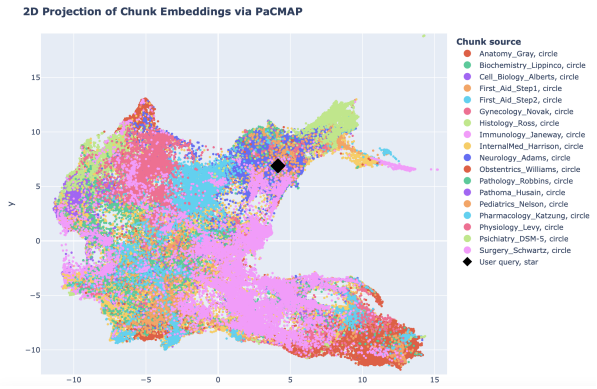

Figure 1: A lesion causing compression of the facial nerve at the stylomastoid foramen will cause ipsilateral

Ans:

Query: *A lesion causing compression of the facial nerve at the stylomastoid foramen will cause ipsilateral...*

The figure below presents a **2D projection of chunk embeddings using PaCMAP**. Each colored dot represents a chunk from different medical textbooks, grouped according to their sources. The **black star** indicates the **user query embedding**, while the closest colored points denote the most relevant retrieved snippets.

Observations:

- **Clustered by Textbook Titles**: The embeddings from the same textbook form distinct clusters, indicating strong semantic grouping.
- **Query Location and Nearest Chunks**: The user query is positioned close to **Neurology Adams, Pathology Robbins, InternalMed Harrison**, which aligns with the facial nerve-related query.
- **Importance of High-Dimensional Mapping**: Unlike PCA, which preserves global structure, **PaCMAP retains local topology**, making it ideal for retrieval-based tasks.

f. Now we have a simple pipeline for Medical QA using RAG. As you read in the tutorials, to improve our model we can tune the k hyper-parameter.

• what is the maximal reasonable K for this case? (considering the chunk-size of our document and the max seq length of the model)

Ans:

- **Model's Maximum Sequence Length**: The *Zephyr-7B-Beta* model has a maximum context length of 3,696 tokens.
- **Chunk Size**: The *MedRAG/textbooks* dataset consists of snippets averaging 182 tokens each. After investigating the textbooks we found out that there is a small fraction of the texts that are 800-1050 token long (so we will divide by 2).
- **Calculation**: To avoid exceeding the model's context window , calculate the maximum *top_k* as follows:

$$\text{Max } k = \left\lfloor \frac{\text{Model's Max Sequence Length}}{2 * \text{Average Chunk Size}} \right\rfloor = \left\lfloor \frac{3696}{364} \right\rfloor \approx 10$$

  Therefore, setting *top_k* to 20 is reasonable, ensuring the combined token count of retrieved snippets stays within the model's capacity.

- Is it always preferable to choose the largest k possible?

Ans:

In a Retrieval-Augmented Generation (RAG) system, selecting the appropriate *top_k* parameter is crucial for balancing information retrieval and model performance. The *top_k* parameter specifies the number of retrieved snippets provided to the model. Setting this parameter requires careful consideration of the model's maximum sequence length and the size of each retrieved chunk.

- **Relevance vs. Noise**: While increasing *top_k* can enhance the chance of retrieving relevant information, it also raises the risk of introducing irrelevant data, which may confuse the model.
- **Lost-in-the-Middle Phenomenon**: Providing too much information can overwhelm the model, leading to diminished performance. It's crucial to supply only the most pertinent snippets to maintain effectiveness.

Conclusion

While *top_k* aligns with the model's capacity, it's not always preferable to choose the largest possible *top_k*. Balancing the quantity of retrieved snippets with their

relevance is essential. It's advisable to experiment with different *top_k* values to determine the optimal setting for your specific application.

2 In-Context Learning

3 Pre-trained Transformers (Bonus Question)

a. Report your model's accuracy on the dev set:

Ans:

FineTune without Pretrain - Correct: 8.0 out of 500.0: 1.6

b. As a reference point, we want to also calculate the accuracy the model would have achieved if it had just predicted "London" as the birth place for everyone in the dev set. Fill in london baseline.py to calculate the accuracy of that approach and report your result in your write-up.

Ans:

London baseline - Correct: 25.0 out of 500.0: 5.0

c. Pretrain your model on wiki.txt, finetune it on NameDataset and evaluate it.

Report the accuracy on the dev set:

Ans:

Pretrain + Finetune - Correct: 122.0 out of 500.0: 24.4

d. succinctly explain why the pretrained (vanilla) model was able to achieve an accuracy of above 10%, whereas the non-pretrained model was not

Ans:

Pretraining on `wiki.txt` provided the model with a rich understanding of character-level patterns, such as common sequences of characters, frequently occurring

words, and how characters combine to form meaningful substrings. This knowledge is crucial for tasks involving character-level tokenization, as it allows the model to develop a strong foundation in recognizing and encoding meaningful subunits from raw character input.

During pretraining, the model also learned to encode long-range dependencies and patterns across characters, which are essential for understanding proper nouns, like names of people and places. For example, it could learn that certain character sequences (e.g., "New ", "York") are associated with place names. This general understanding of character sequences significantly reduced the burden during finetuning, as the model could focus on adapting this knowledge specifically to birthplaces.

In contrast, the non-pretrained model started with randomly initialized weights, meaning it had no prior understanding of character patterns or how these patterns relate to names and locations. As a result, it had to learn the relationships between character sequences and birthplaces entirely from the `NameDataset`, which is small and task-specific. This made it challenging for the non-pretrained model to capture generalizable patterns and often led to overfitting on the limited data.

Moreover, the pretrained model could leverage its learned representations to generalize better to unseen names and birthplaces during evaluation. For example, it might recognize that certain character combinations often appear in geographical locations, even if those specific places were not part of the training data. This ability to generalize gave the pretrained model a significant advantage over the non-pretrained one.

Finally, pretraining helps with weight initialization, leading to faster convergence during finetuning. The non-pretrained model, on the other hand, suffered from inefficient learning dynamics and struggled to achieve meaningful accuracy given the complexity of the task and the limited data.