Sentence Transformer Model Testing

Model Information:

Model name: Alibaba-NLP/gte-Qwen2-1.5B-instruct

Max sequence length: 512 (Max: 8192)

Model Size: 1.5B

Embedding Dimension: 1536

Max Input Tokens: 32k

Size on disk: 6.62GB

Testing Platform:

Platform name: Google Colab

Device: T4 GPU

Dataset Description:

Dataset 1:

Number of queries: 10

Average query length: 65 characters

Number of documents: 10

Average document length: 410 characters

Language: Nepali

Dataset 2:

Document type: PDF

Language: Nepali

Number of pages: 100

Evaluation Metrics:

Latency: The time taken to embed the given text.

Memory usage: The memory utilization during embedding.

Time Measurement Technique:

The following decorator function was used to measure execution time:

```python
import time
def time_it(func):
    def wrapper(*args, **kwargs):
        start_time = time.time()  # Record the start time
        result = func(*args, **kwargs)  # Call the wrapped function
        end_time = time.time()  # Record the end time
        elapsed_time = end_time - start_time  # Calculate elapsed time
        print(f"Function '{func.__name__}' took {elapsed_time:.6f} seconds to execute.")
        return result  # Return the result of the wrapped function
    return wrapper
```

Results:

GPU memory usage for model:

```
|===========================================================================|
|                  PyTorch CUDA memory summary, device ID 0                 |
|---------------------------------------------------------------------------|
|            CUDA OOMs: 0            |        cudaMalloc retries: 0         |
|===========================================================================|
|        Metric         | Cur Usage  | Peak Usage | Tot Alloc  | Tot Freed  |
|---------------------------------------------------------------------------|
| Allocated memory      |   9471 MiB |   9471 MiB |   9471 MiB |      0 B   |
|---------------------------------------------------------------------------|
| Active memory         |   9471 MiB |   9471 MiB |   9471 MiB |      0 B   |
|---------------------------------------------------------------------------|
| Requested memory      |   9471 MiB |   9471 MiB |   9471 MiB |      0 B   |
|---------------------------------------------------------------------------|
| GPU reserved memory   |   9592 MiB |   9592 MiB |   9592 MiB |      0 B   |
|---------------------------------------------------------------------------|
| Non-releasable memory | 123784 KiB | 123802 KiB | 459206 KiB | 335422 KiB |
|---------------------------------------------------------------------------|
| Allocations           |     422    |     422    |     422    |       0    |
|---------------------------------------------------------------------------|
| Active allocs         |     422    |     422    |     422    |       0    |
|---------------------------------------------------------------------------|
| GPU reserved segments |     171    |     171    |     171    |       0    |
|---------------------------------------------------------------------------|
| Non-releasable allocs |      61    |      61    |     115    |      54    |
|---------------------------------------------------------------------------|
| Oversize allocations  |       0    |       0    |       0    |       0    |
|---------------------------------------------------------------------------|
| Oversize GPU segments |       0    |       0    |       0    |       0    |
|===========================================================================|
```

Memory Usage: 9471 MiB

Time and Memory Evaluations:

For Dataset 1

For	Time	Memory usage
1 query embedding	0.208867s	10 MiB per page
Batch of 10 queries	0.650527s	35 MiB
Multithreaded 10 queries	1.425632s	10 MiB per thread

For Dataset 2
20 pages

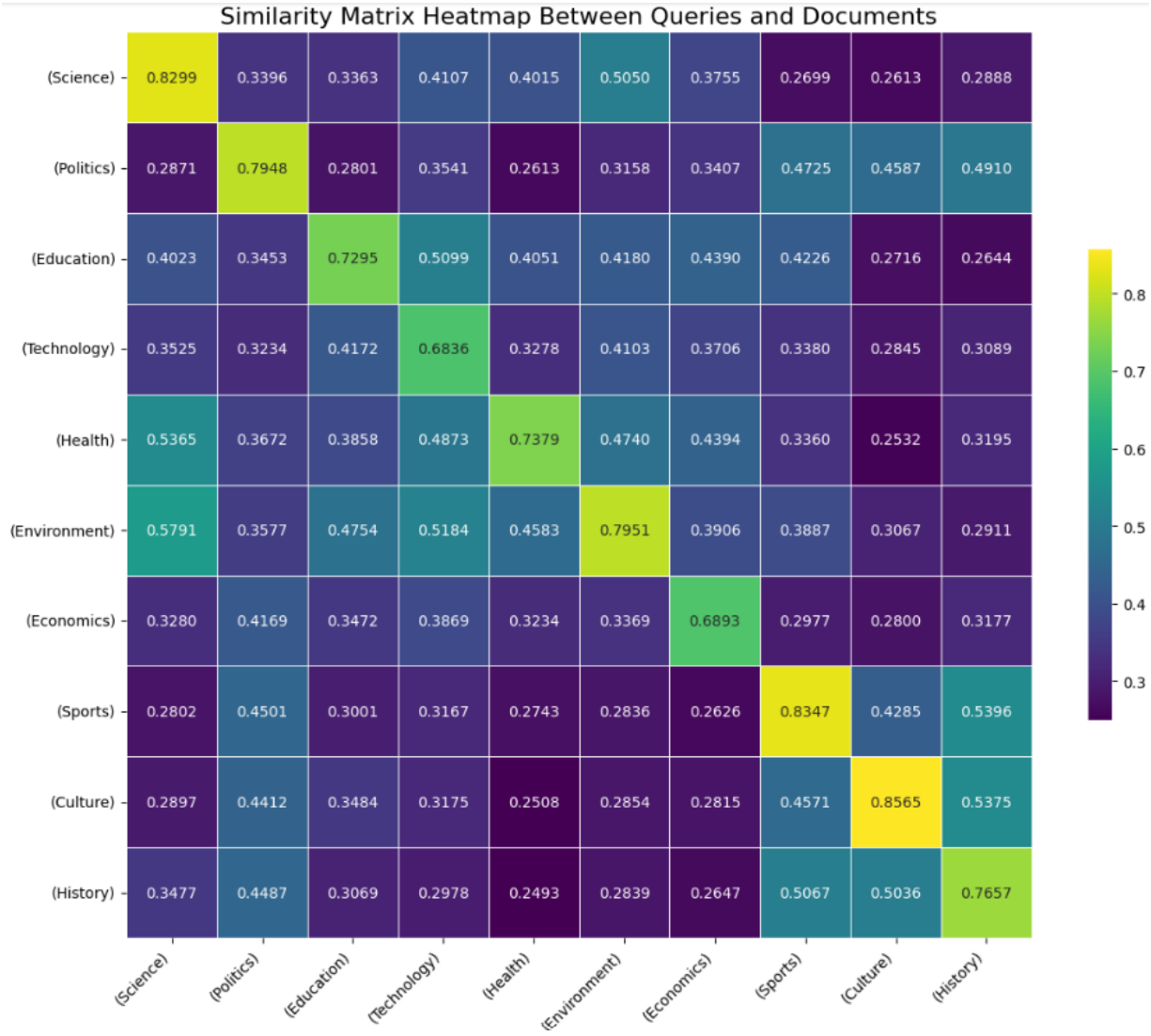
For	Time	Memory usage
1 page embedding at a time	9.21s	32 MiB per page
Batch embedding of 20 pages	7.881586s	621 MiB
Multithreaded with 6 threads	8.479194s	186 Mib per thread

100 pages

For	Time	Memory usage
1 page embedding at a time	45.602843s	32 MiB per page
Batch embedding 100 pages (25 per batch)	36.929543s	2422 MiB per batch
Multithreaded with 6 threads	37.447301s	176 MiB per thread
Batch Multithreaded with 6 threads (4 per batch)	9.793176s	724 MiB per thread

Embedding Evaluation:

Similarity Matrix



The similarity matrix shown in the figure above provides a heatmap representation of the similarity scores between 10 queries and 10 corresponding documents. Each cell in the matrix represents the similarity score between a specific query and a document, where higher values indicate greater similarity.

Key Observation:

- The diagonal entries of the matrix consistently exhibit the highest similarity scores. These entries represent cases where queries are matched with their corresponding documents (e.g., Science query to Science document, Politics query to Politics document).
- For example, the similarity score for "Science" is 0.8299, and for "Sports," it is 0.8347, indicating strong alignment between queries and documents in these categories.

Findings:

For Query dataset

- Batching results in significant increase in processing speed with moderate increase in memory usage.
- Multithreading resulted in higher processing time compared to batching, likely due to thread synchronization overhead, despite lower per-thread memory usage.

For PDF dataset

- Embedding single page at a time is memory efficient but highly time in-efficient.
- Batching improves processing time but requires significantly more memory.
- Multithreading reduces memory usage compared to batch embedding but shows marginally higher time due to thread overhead.
- Combines the benefits of batching and multithreading, significantly reducing processing time and optimizing memory usage.

Recommendations:

- For small datasets (e.g., single queries or pages), single embedding is sufficient.
- For medium-sized datasets, batching without multithreading is effective but memory-intensive.
- For large datasets, batch multithreading is the optimal choice to balance time and memory usage.

Conclusion:

The **Alibaba-NLP/gte-Qwen2-1.5B-instruct** model exhibits excellent semantic representation capabilities, as evident from its ability to produce high similarity scores between related queries and documents. Its efficient handling of diverse input queries and adaptability for batching and multithreading make it a robust choice for large-scale embedding tasks and retrieval systems. However, its high memory usage indicates that it is best suited for environments with adequate computational resources.