**Static Router Methodology**

3/9/2024

We use below table to study the deployed static router on various embedding model to find the optimal threshold value for routes, especially for **‘None’** when no match found.

**Objective:** To find the optimal similarity score threshold below which a route will be classified as 'None'.

Table 1. Configuration wise route, utterances per route and test prompts

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Study Configurations | **Routes** | **Utterances per route** | **Test prompts** | | | |
| **Exact Match**  (1 per route) | **Partial Match**  (1 per route) | Unrelated or  No match **(None)**  (same number of total routes) | **Total  test prompts** |
| Configuration (2,5) | **2** | **5** | 2 | 2 | 2 | **6** |
| Configuration (4,10) | **4** | **10** | 4 | 4 | 4 | **12** |
| Configuration (6,15) | **6** | **15** | 6 | 6 | 6 | **18** |
| Configuration (8,20) | **8** | **20** | 8 | 8 | 8 | **24** |

Table 2. Names of the routes for each configuration

|  |  |  |  |
| --- | --- | --- | --- |
| Routes of Configuration (2,5) | Routes of Configuration (4,10) | Routes of Configuration (6,15) | Routes of Configuration (8,20) |
| Physics | Physics | Physics | Physics |
| Biology | Biology | Biology | Biology |
|  | History | History | History |
| Philosophy | Philosophy | Philosophy |
|  | Chemistry | Chemistry |
| Computer\_Science | Computer\_Science |
|  | Sculpture |
| Dance |

Perform below for each embedding model

1. Run ‘static\_router\_X\_Y.py’ // X is same as routes per table 2 and Y is utterances per route
2. Run ‘curl\_caller\_X\_Y.py’ that POST prompts sequentially to ‘static\_router\_X\_Y.py’
3. Log to csv

Accumulate all csv files into a final **excel** **sheet** that will be used for analysis for the data based on ‘semantic\_similarity\_score’ to find optimal threshold for each route for a given embedding model.

The **‘semantic\_similarity\_score’** 1 means full correct route selection.

We include 2 new columns such as **“routes**” and “**utterances\_per\_route**” in the final excel sheet for 4 configurations manually.

We also append a new column **“route\_select-correctness”** in the final excel sheet and we mention 1 (correct) or 0 (wrong) manually. We aim to find the optimal threshold value based on such supervised mapping (based on our manual inclusion of 1 or 0 of selected route (i.e. route\_select\_correctness) later below which a correct ‘None’ route will be selected for a given embedding model.

We include a new column “**true\_route\_match\_type”** in the final excel sheet that has three classes such as “exact-match”, “partial\_match” and ‘None” based on the test prompts as per curl\_caller code which I included manually. Ordinarily, extact\_match and partial\_match should be matched with the ‘route\_selected’ and ‘None’ should be matched with any random route. I want to know the accuracy, precision, recall, F1 of ‘route\_selected’ with respect to my assumption of coding logic. The ‘route\_select-correctness’ value should be 1 for correct ‘route\_selected’ with respect to ‘true\_route\_match\_type’. I have to find what is optimal threshold value for ‘None’.

In my coding logic I assume that the ‘route\_select-correctness’ should be 1 for a ‘route\_selected’ for a ‘exact\_match’. If ‘exact\_match’ is 1 for respective ‘**route\_select\_correctness**’ it means the route is correctly selected. If a ‘**route\_selected**’ of ‘exact\_match’ has corresponding value 1 for ‘**route\_select\_correctness**’ then it is actually correct, else wrong.

In my coding logic I assume that the ‘route\_select-correctness’ should be 1 for a ‘route\_selected’ for a ‘partial\_match’. If ‘partial\_match’ is 1 for respective ‘**route\_select\_correctness**’ it means the route is correctly selected. If a ‘**route\_selected**’ of ‘partial\_match’ has corresponding value 1 for ‘**route\_select\_correctness**’ then it is actually correct, else wrong.

In my coding logic I assume that the ‘route\_select-correctness’ should be 0 for a ‘route\_selected’ for a ‘None. I need to find the optimal value of the ‘None’. As per the coding logic, ‘None’ route selection should always be 0.

Such 1 and 0 mapping is done to supervise the understanding which route selection is truly correct and which is not.

However, the embedding model may do wrong route selection.

**CSV important column, meaning and their unit of measurement:**

semantic\_similarity\_score No unit as per cosine similarity measurement

similarity\_metric Cosine similarity

vector Vector of the user’s prompt i.e. test prompt. Vector size is different for the embedding models.

total\_duration Time spent generating the response of (ns)

load\_duration Time spent in nanoseconds loading the embedding model(ns)

prompt\_eval\_count Number of tokens in the prompt

avg\_cpu\_usage\_during Average CPU usage during the operation (%)

memory\_usage\_mb Memeory usage during operation (MB)

network\_latency Network latency (ns)

total\_response\_time Total response (s)

========================================================================

Perform the accuracy of such route selection per configuration per embedding model must be assessed.

Perform distribution of ‘**semantic\_similarity\_score**’ per route per embedding model.  
Embedding model wise comparison of accuracy etc should be measured.

Perform **Precision, Recall, F1 and ROC** (the area under the curve (AUC) can help determine a cut-off that maximizes sensitivity and specificity.) to find which route has what optimal threshold, especially for ‘None’

This process will help identify the optimal threshold that **minimizes** **false positives** (incorrect route selections) while **maximizing true negatives** (correctly identifying **'None'**).

Perform Threshold Analysis: Focus on the 'None' class to determine the optimal threshold value based on the distribution of the semantic\_similarity\_score. The identified threshold value can be used for the given embedding model to efficiently classify user’s prompt to be ‘None’ or a match route.

Perform Response Time vs. Accuracy: Analyze how response time metrics (e.g., total\_response\_time) vary with accuracy for each model to ensure that the best-performing models are also efficient.

Perform Route Selection Accuracy per Embedding Model

Perform Distribution of Semantic Similarity Scores by Route Match Type

The whole code is based on Ollama API call being served by FastAPI application on top of Uvicorn ASGI Web Sever on Raspberry Pi 4 resource constrained edge device.

<https://github.com/ollama/ollama/blob/main/docs/api.md#generate-embeddings>

Embedding models via Ollama:

Command: [ollama pull mxbai-embed-large]

**Usage:**

**REST API**

**[Single Input]**

curl http://localhost:11434/api/**embed -d** '{

"model": "all-minilm:33m",

"input": "The sky is blue because of Rayleigh scattering"

}'

**[Multiple Input]**

curl http://localhost:11434/api/**embed -d** '{

"model": "all-minilm:33m",

"input": ["Why is the sky blue?", "Why is the grass green?"]

}'

* all-minilm:33m

This is a **sentence-transformers** model: It maps sentences & paragraphs to a 384 dimensional dense vector space and can be used for tasks like clustering or semantic search.

* mxbai-embed-large

mxbai-embed-large-v1 is our powerful English embedding model that provides state-of-the-art performance among efficiently sized models. It outperforms closed source models like OpenAI's text-embedding-ada-002.

The model was trained on a vast dataset of over 700 million pairs using contrastive training and fine-tuned on more than 30 million high-quality triplets using the **AnglE** loss function. This extensive training enables the model to adapt to a wide range of topics and domains, making it suitable for various real-world applications and Retrieval-Augmented Generation (**RAG**) use cases.

mxbai-embed-large-v1 is well-suited for **binary embeddings**.

* nomic-embed-text

Nomic Embed, the first Open source, Open data, Open training code, Fully reproducible and auditable model. It is a text embedding model with a 8192 context-length that outperforms OpenAI Ada-002 and text-embedding-3-small on both short and long context tasks. It is a high-performing open embedding model with a large token context window.

* snowflake-arctic-embed:335m

snowflake-arctic-embed is a suite of text embedding models that focuses on creating high-quality retrieval models (RAG) optimized for performance.

The models are trained by leveraging existing open-source text representation models, such as bert-base-uncased, and are trained in a multi-stage pipeline to optimize their retrieval performance.

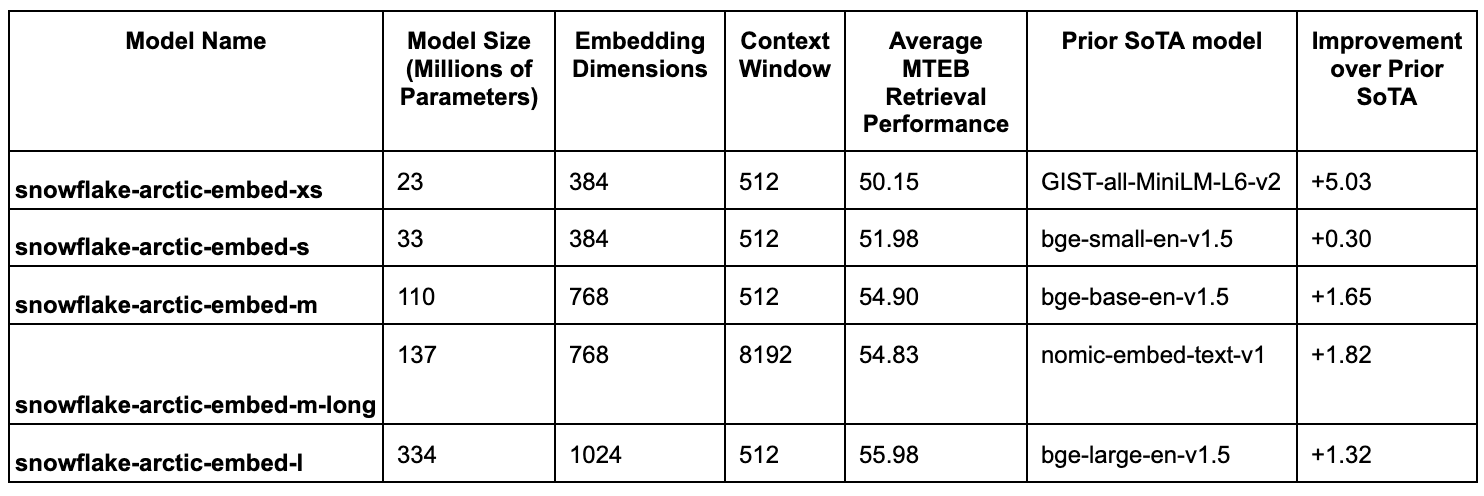
Table 1. Ollama specific model comaprisons

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ollama Embedding Models | Parameters | Size | Architecture | Quantization | Context Length (num\_ctx) | Embedding Dimensions  (max) | License |
| all-minilm:22m | 22.6M | 46MB | BERT | F16 | 256 | 384 | Apache |
| all-minilm:33m | 33.2M | 67MB | BERT | F16 | 256 | 384 | Apache |
| nomic-embed-text | 137M | 274MB | Nomic-BERT | F16 | 8192 | 768 | Apache |
| **snowflake-arctic-embed:22m** | **22.6M** | **46MB** | **BERT** | **F16** | **512** | **384** | **Apache** |
| snowflake-arctic-embed:33m | 33.2M | 67MB | BERT | F16 | 512 | 384 | Apache |
| snowflake-arctic-embed:110m | 109M | 219MB | BERT | F16 | 512 | 768 | Apache |
| mxbai-embed-large | 334M | 670MB | BERT | F16 | 512 | 1024 | Apache |
| snowflake-arctic-embed:137m | 137M | 274MB | BERT | F16 | 8192 | 768 | Apache |
| **snowflake-arctic-embed:335m** | **334M** | **669MB** | **BERT** | **F16** | **512** | **1024** | **Apache** |

References:

**nomic-embed-text v1.5** is **resizable embedding dimensions** from 64 to 768, <https://docs.nomic.ai/atlas/capabilities/embeddings>

<https://www.snowflake.com/en/blog/introducing-snowflake-arctic-embed-snowflakes-state-of-the-art-text-embedding-family-of-models/>



**MTEB Leaderboard** <https://huggingface.co/spaces/mteb/leaderboard>





