October 2023

AI-based Adaptive Load Balancer for Secure Access to Large Language Models

Assaf Namer

Hector Diaz

Jim Miller

Brandon Maltzman

Hauke Vagts

AI-based Adaptive Load Balancer for Secure Access to Large Language Models

ABSTRACT

Different large language models (LLMs) specialized to domains such as writing code, engaging in conversations, generating content, etc. are available. A specialized LLM can only reliably answer questions in domains over which it has been trained. Large numbers of types of specialized LLMs can make it difficult for a user, such as an application that generates LLM queries, to choose the right type of LLM. This disclosure describes techniques to automatically route query payloads between large language models specialized for different domains. The techniques utilize a vector database to semantically match an LLM to a user query. The techniques also provide a real-time feedback and adaptation mechanism. Security checks and access controls are applied in a centralized manner while adhering to security compliance regimes. The techniques provide improved end-to-end security posture of AI-based applications and user experience. The techniques can also reduce the costs of querying large LLMs.

KEYWORDS

- Large language model (LLM)
- Generative AI
- AI security
- AI reliability
- Data governance
- Multi-LLM architecture
- Load balancing
- Similarity search

BACKGROUND

Large language models (LLMs) are a type of trained artificial intelligence (AI) that can generate text and other content, translate languages, answer questions in an informative way, etc. Some applications of LLMs include:

- chatbots that can have nearly natural conversations with humans;
- text content generation for articles, blog posts, books, etc.;
- translation between natural languages;
- question-answering systems; etc.

Training a single LLM that suits multiple tasks or domains can be difficult. Rather, models optimized to specific tasks or domains are created, providing better performance and accuracy. The criteria by which models are created for specific tasks include:

- **Data type:** An LLM for writing code is trained on a dataset of code, while an LLM for chats is trained on a dataset of conversations.
- **Accuracy:** An LLM for mission-critical applications, e.g., medical diagnosis, etc., is highly accurate, whereas an LLM that generates free-form text is less constrained by accuracy. Accuracy can be controlled by 'temperature,' a parameter that controls the degree of randomness in generative AI.
- **Compliance:** It is important that LLMs used in particular industries are compliant with the regulations for that industry. Task-specific LLMs (rather than generic LLMs) can be better designed and trained to ensure regulatory compliance.

Currently, there is effectively a one-to-one mapping between the domain and the type of LLM. There are LLMs specialized to writing code, engaging in conversations, generating content, etc. There are even LLMs specialized to specific tasks, such as cybersecurity malware

and detection. These can find insight from suspicious files; discover access-pattern anomalies, write detection rules; debug code; etc. An LLM specialized to a certain domain generally has low accuracy when generating questions unrelated to the domain.

The large numbers of types of specialized LLMs can make it difficult for a user to choose the right type of LLM. Users may not know the existence and suitability of particular LLMs and may therefore find it difficult to identify a model appropriate to the task at hand.

Embeddings are a machine-learning technique of representing data as points in n-dimensional space such that similar data points cluster together. Embeddings can enable discovery of similar data using techniques such as nearest neighbor search, range search, hierarchical search, etc. Embeddings search, also known as similarity search, is the search for data similar to given data. Embeddings search can be done by cloud-based matching engines that can respond to queries that request data similar to given data, e.g., return similar rows from a database.

A network load balancer uses network attributes such as source/destination IP, protocol, source/destination port, etc. to route network traffic to different backends or targets. An HTTPS load balancer, also known as an application load balancer, works by terminating the HTTPS connection at the load balancer and by proxying the connection to the backend server. The load balancer decrypts HTTPS traffic and re-encrypts it to send it to the backend server. Attributes used to route traffic can include host header, path, query string, HTTPS headers, etc. An HTTPS (or similar) load balancer can be implemented by a cloud-based cluster of containers, in which case it may be referred to as a global load balancer.

DESCRIPTION

This disclosure describes techniques to automatically select and balance payloads between large language models (LLMs) specialized to different domains, with a real-time feedback and adaptation mechanism and a vector database to semantically match LLMs to user queries. Security checks and controls are applied in a centralized manner while adhering to security compliance regimes. The techniques provide an improved end-to-end security posture of AI-based applications and an improved user experience.

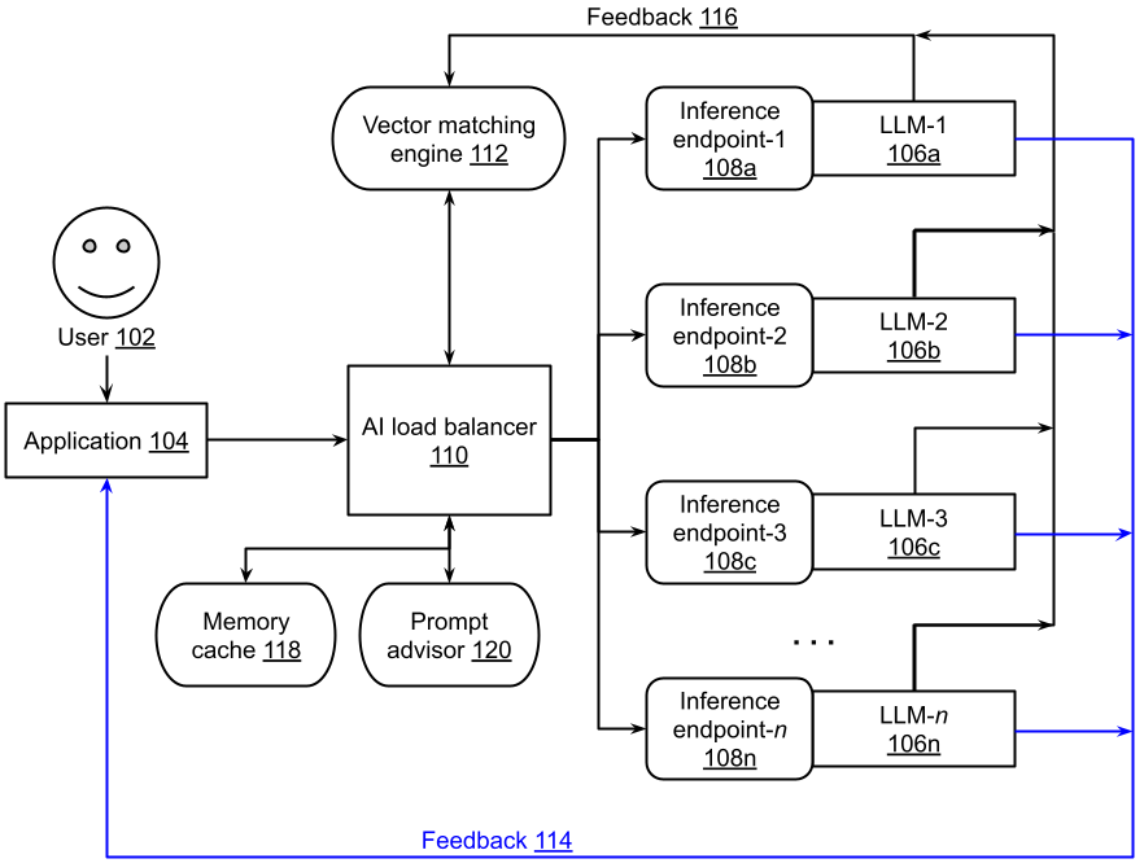

Fig. 1: Adaptive AI-based load balancer for secure access to large language models

Fig. 1 illustrates an example adaptive AI-based load balancer for secure access to large language models. A user (102) provides a query or prompt via an application (104), thereby

generating a payload. The payload generated by the user/application is to be divided among one or more of a set of LLMs (106a-n). The LLMs in the set may each be specialized to particular domains, e.g., code writing, blog writing, translation, summarization, reasoning, etc. Multiple different payloads may be assigned to respective LLMs.

Each LLM in the set has an application programming interface (API) endpoint, also known as prediction or inference endpoint (108a-n), that is used by the application to access the LLM. An AI-based load balancer (110) queries a vector matching engine (112) or matching database and uses the results from the matching engine/database to route requests from the application to the appropriate LLM(s). The API endpoints of the LLMs are used as the backends of the AI load balancer. The load balancer caches the results of the query from the vector matching engine in a memory cache (118).

For example, a request from the application related to the analysis of a financial report is routed, based on the result from the matching engine, to an LLM that is trained over financial data. As another example, a request related to a cybersecurity attack vector is routed by the AI load balancer to an LLM trained over security data.

The described AI-based LLM load balancer terminates client (application) requests to the LLMs by running a query against a matching engine database. The query results identify an appropriate LLM to which payload is assigned. Thus, the operations are similar to a global load balancer of the type used for HTTPS load-balancing.

The quality of the text (or other response) generated by the LLM is fed back via a first feedback mechanism (114, in blue) to enable adaptive improvement of the accuracy of the AI-based load balancer. The quality metric that is fed back can be generated by humans, or can be generated by the LLM model itself. Examples of techniques to score the quality of LLMs include

perplexity (which measures how well the LLM predicts a given sequence of words, with lower perplexity indicating better performance); BLEU (bilingual evaluation understudy) [6]; ROUGE (recall-oriented understudy for gisting evaluation) [7]; METEOR (metric for evaluation of translation with explicit ordering) [8]; etc. In addition to quality, the LLM output can also be assessed for other criteria, e.g., toxicity (output that is offensive or includes unwanted biases). Toxicity can be measured by a variety of techniques, such as by the use of a toxicity lexicon, etc.

A second feedback mechanism (116) feeds the quality, toxicity, and other metadata relating to the LLM output to the vector matching engine. Requests to the LLM are maintained in the vector matching engine/database and applied towards future requests. Specifically, when the AI load balancer receives a request, the request is matched against the metadata of past requests in the vector database. A predefined policy is looked up to determine the course of action for the request. For example, a request that matches a past request associated with high toxicity can be blocked or sent to a prompt advisor (120, explained in greater detail below) that suggests an alternative prompt. The response is integrated into the vector database in the same row as the encoded text.

When a prompt similar to a previously processed prompt is received, the AI load balancer can adjust the routing of the prompt by checking the LLM that was used in the previous call and the quality of the result generated during the previous call. For example, if the result from the previous similar call had a low relevance score, the AI load balancer can route the request to a different LLM. Alternatively, the prompt can be forwarded to a prompt advisor (120), which returns an updated version of the original prompt that is more relevant to the task being requested. The prompt advisor can also be used to adjust the accuracy and/or temperature of the request.

In this manner, the described AI-based load balancer deploys an application frontend, multiple backend LLMs, a vector matching engine, a prompt advisor, a memory cache, etc. to match a received query against the vector matching engine to automatically select an LLM that is appropriate for the query. LLM output metrics are fed back to the vector matching engine and saved for future use. The cache can be searched for similar past requests. If a similar (in a nearest-neighbor sense) past request exists, the quality of the result corresponding to the past request can inform the selection of the LLM for the present request. A prompt advisor can modify the prompt based on the quality of past results. The adaptive load balancer serves as a centralized point for security and auditing. The adaptive feedback mechanism can improve the quality of responses and of routing to LLMs. The complexity of specialized LLMs is hidden from the user. Application frontends can thus provide improved LLM performance and reliability.

The described AI-based load balancer provides security and compliance advantages including:

- Centralized access to the LLMs can improve security. Since requests from applications that are directed to LLMs are routed via the AI-based load balancer, strict access control can be applied with differing security levels (e.g., identity-aware proxy, multi-factor authentication, device attributes, etc.). For example, a request to summarize the future business strategy of a company can be associated with stricter security controls than a request to summarize the news of the day. Security-related checks can also be performed prior to sending the requests from the application to the AI-based load balancer.
- Auditing, logging, and monitoring of requests coming into the LLMs is centralized through the AI-based load balancer in a manner that is transparent to the user.

- Prompt checks and prompt recommendations can alleviate offensive prompts.
- For improved resilience, the AI-based load balancer can have multiple endpoints in different cloud regions. If a cloud-region goes offline, other regions can seamlessly take over requests. Furthermore, a customer can configure an endpoint as being on-premise or off-premise (in the cloud) based on compliance, security, and/or redundancy requirements.
- By functioning as an enforcement point for customer security policies, the AI-based load balancer effects a zero-trust principle over and above the existing security measures of the organization.
- The adaptive feedback mechanism of the AI-based load balancer can improve LLM routing decisions by updating the vector database with up-to-date information.

CONCLUSION

This disclosure describes techniques to automatically route query payloads between large language models specialized for different domains. The techniques utilize a vector database to semantically match an LLM to a user query. The techniques also provide a real-time feedback and adaptation mechanism. Security checks and access controls are applied in a centralized manner while adhering to security compliance regimes. The techniques provide improved end-to-end security posture of AI-based applications and user experience. The techniques can also reduce the costs of querying large LLMs.

REFERENCES

1. “External application load balancer overview,” available online at https://cloud.google.com/load-balancing/docs/https accessed Oct. 7, 2023.
2. “Open-source vector similarity search for Postgres,” available online at https://github.com/pgvector/pgvector accessed Oct. 7, 2023.
3. “Artificial intelligence risk management framework (AI RMF 1.0),” available online at https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-1.pdf accessed Oct. 7, 2023.
4. “Test code generation prompts,” available online at temperature https://cloud.google.com/vertex-ai/docs/generative-ai/code/test-code-generation-prompts accessed Oct. 7, 2023.
5. “Supercharge security with AI,” available online at https://cloud.google.com/security/ai accessed Oct. 7, 2023.
6. Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "Bleu: a method for automatic evaluation of machine translation." In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311-318. 2002.
7. Lin, Chin-Yew. "Rouge: A package for automatic evaluation of summaries." In *Text summarization branches out*, pp. 74-81. 2004.
8. Banerjee, Satanjeev, and Alon Lavie. "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments." In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65-72. 2005.