Memória Episódica em Modelos Generativos de Conversação

1st Nilo Garcia Monteiro
Departamento de Informática
UFES – Vitória, ES
nilo.monteiro@edu.ufes.br

2nd Pedro Igor Gomes Morais
Departamento de Informática
UFES – Vitória, ES
pedro.i.morais@edu.ufes.br

***Abstract*—A implementação de memória episódica em modelos generativos tem se mostrado uma área promissora. Ela permite que sistemas de conversação façam uso de informações de interações pretéritas, o que proporciona respostas contextualizadas e personalizadas. Este trabalho investiga uma das possíveis formas de implementação, explorando a combinação de técnicas de extração de tema por meio de YAKE e RAKE com a recuperação de janelas temporais. Para avaliação de desempenho, foram utilizados um *dataset* de perguntas e respostas em conjunto com os modelos generativos Llama, GLM e DeepSeek, bem como de encoder, GTE, JINA e MULTILINGUAL-E5-LARGE. Resultados são mostraram-se passíveis de exploração futura, alcançando até 52% de sucesso.**

***Index Terms*—Retrieval-Augmented Generation, Memória Episódica, Modelos Generativos de Conversação, DeepSeek, Llama**

I. Introdução

A implementação de memória episódica em modelos generativos de conversação tem se mostrado uma área promissora para aprimorar a coerência e a contextualização em diálogos de longo prazo. A memória episódica permite que sistemas de conversação acessem e utilizem informações de interações passadas, proporcionando respostas mais contextualizadas e personalizadas. No entanto, a integração eficiente dessa memória em modelos generativos ainda apresenta desafios, especialmente no que diz respeito à recuperação e ao uso de informações relevantes de grandes volumes de dados de conversação.

Desta forma, outros trabalhos propuseram-se a investigar possíveis formas de implementar memória episódica em modelos generativos de conversação, utilizando técnicas avançadas de processamento de linguagem natural (PLN) e recuperação de informações. Em [1], a recuperação de trechos relevantes para o modelo generativo tinha como intuito contextualizar a conversa com base em seu tema. Com tal escopo, a *framework* de Retrieval-Augmented Generation (RAG) obteve resultados satisfatórios. Entretanto, ao introduzir a dimensão temporal no problema, o método pode ser insuficiente para interpretar mensagens referentes a períodos.

O objetivo principal deste estudo, portanto, é investigar possíveis formas de implementar memória episódica em modelos generativos de conversação, explorando a combinação de técnicas de extração de tema com a recuperação de memória em janelas de tempo. Logo, a hipótese construída é que tal abordagem pode oferecer ganhos significativos na precisão da busca e na relevância das memórias encontradas, gerando respostas mais congruentes entre as conversações passadas e a atual. Para avaliar o desempenho da abordagem, um *dataset* de perguntas e respostas foi testado utilizando vários modelos generativos (META-LLAMA/LLAMA-3.1-8B-INSTRUCT [2], THUDM/GLM-4-9B-CHAT [3] e DEEPSEEK-AI/DEEPSEEK-R1-DISTILL-LLAMA-8B [4]) e de *encoder* (THENLPER/GTE-LARGE [5], JINAAI/JINA-EMBEDDINGS-V3 [6] e INTFLOAT/MULTILINGUAL-E5-LARGE [7]). Além disso, foram testadas técnicas de extração de temas por meio de *YAKE* [8] e *RAKE* [9]. Resultados mostram uma eficiência razoável da abordagem, com uma experimentação final alcançando até 52% de sucesso em recuperar as memórias corretas para o contexto. Logo, após certa adaptação, a abordagem é passível de ser adicionada à *ChatBots*, os oferecendo a habilidade de lembrar de informações relevantes sobre interações passadas com o usuário.

Desta forma, as contribuições deste trabalho são listadas abaixo:

- Exploração de métodos de extração de temas de memórias para aumentar a performance da busca por similaridade;
- Implementação de uma pipeline para análise crítica das memórias encontradas, que seleciona memórias baseando-se em sua relevância em relação ao contexto;
- Estudo da eficiência de vários modelos generativos e de *encoding* na tarefa de extração de datas implícitas em uma mensagem.

II. Implementação

A abordagem proposta por este trabalho leva em consideração o *framework* de RAG, com a adição de uma busca temporal de memórias. O sistema, portanto, pode ser dividido em dois módulos: Geração do Banco de Memórias e Geração de Respostas. A Geração do Banco de Memórias transforma as conversações passadas em um banco de vetores, em que cada item é um *emebedding*. Enquanto isso, a Geração de Respostas analisa a mensagem do usuário, extrai informações importantes para a busca de memórias, analisa e filtra memórias relevantes e então formula uma resposta para a mensagem do usuário. A Figura 1 apresenta ambos os módulos do sistema,

mostrando suas relações no processo executado para responder à mensagem do usuário.

A. Geração do Banco de Memórias

No *framework* de RAG, um dos principais componentes é o banco de vetores, essencial para a busca de contextos semanticamente próximos da mensagem do usuário. Com isso, o módulo Geração do Banco de Memórias constrói uma estrutura com vetores que representam cada memória no espaço. Como neste trabalho memórias são caracterizadas como várias rodadas de conversação entre o modelo generativo e o usuário, cada vetor então representa um diálogo. Logo, ao realizar a busca semântica, são procuradas memórias que se assemelham com o tema abordado pela mensagem do usuário.

Ademais, para promover uma busca mais exata, são investigados métodos de sumarização e extração de temas. A hipótese formada é que, ao centralizar o tema do diálogo, as memórias estarão mais espalhadas, distanciando-se entre si, facilitando a busca semântica. Desta forma, os vetores no banco de memórias podem representar tanto o diálogo em si (no método comum) quanto um resumo (no método de sumarização).

Com a construção do banco de memórias, é possível utilizar o mesmo modelo *encoder* para obter um vetor representativo da mensagem do usuário, então utilizando-o para encontrar os k vetores mais similares à mensagem no espaço do banco. E então, como cada vetor no banco de dados é relacionado a uma memória, é possível obter a memória original por meio de seu índice.

B. Geração de Respostas

Uma vez obtido um banco de memórias, é possível extrair os contextos semanticamente relevantes para a conversação. Entretanto, é necessário investigar que contextos são essenciais e como eles devem ser alimentados para o modelo generativo.

Primeiramente, são obtidas as memórias mais semanticamente próximas da mensagem do usuário. Desta forma, são obtidos os diálogos que podem ser relevantes para a conversa. Entretanto, também é necessário investigar o período citado pela mensagem do usuário. Por exemplo, caso o usuário pergunte "O que conversamos no domingo passado?", é necessário investigar memórias que aconteceram neste período. Desta forma, para filtrar o período em que memórias devem ser investigadas, o modelo generativo é instruído por meio de um *prompt*[1] a analisar a mensagem do usuário e retornar períodos que podem conter tal informação. Assim, é possível extrair todas as memórias neste período e investigar as suas relevâncias em relação à mensagem do usuário.

Com ambas as listas de memórias (uma obtida através da busca semântica e outra através da busca temporal), o modelo é instruído por meio de outro *prompt* a analisar cada memória e determinar aquelas necessárias para responder à mensagem do usuário. Com isso, as memórias são filtradas e somente as consideradas essenciais são mantidas. Além de reduzir a chance de alimentar informações irrelevantes ao modelo, tal filtro reduz significativamente o número de memórias, reduzindo o tamanho do *prompt* e, consequentemente, o consumo de memória. Ao fim, o modelo generativo recebe cada memória relevante por meio do *prompt* final, que o instrui a responder o usuário com base nas memórias passadas e na data atual.

[1]Todos os *prompts* utilizados neste trabalho são apresentados no apêndice deste relatório.

III. Metodologia Experimental

A. Dataset

O *dataset* de memórias utilizado neste trabalho, DailyDialog, foi encontrado na plataforma *Kaggle*. Tal *dataset* apresenta 1000 diálogos entre dois locutores, entretanto, sem nenhuma marcação de data associada às conversas. Com isso, cada conversa foi associada à uma data diferente de forma artificial, sendo cada item relacionado à um dia diferente entre 03/03/2025 e 08/06/2022. Alguns exemplos do *dataset* de memórias podem ser vistos na Tabela I.

TABLE I
Exemplos de Diálogos

Diálogo 1 (03/03/2025)
- The taxi drivers are on strike again. - What for? - They want the government to reduce the price of the gasoline. - It is really a hot potato.
Diálogo 359 (09/03/2024)
- Hello, I want to apply for the position of an usher in your restaurant. - Welcome. One of your duties is to know the number of the customer and accept reservation. - What else do I need to do? - You inform the people in charge and let them reserve dining tables for guests.

Para realizar os experimentos, um *dataset* de perguntas e respostas foi criado manualmente, em que cada pergunta é relacionada a uma conversa no *dataset* de memórias. No total, 51 itens foram feitos, e cada um contém: uma pergunta, a resposta esperada, o índice da memória no banco, e um marcador que determina se a pergunta infere corretamente o período correspondente à memória (levando em consideração o dia em que a pergunta foi realizada). Tal marcador tem o objetivo de facilitar a avaliação da extração de datas pelo modelo, uma vez que, se o marcador é verdadeiro, o período inferido pela pergunta do usuário contém a sua respectiva memória. Caso contrário, o período informado pelo usuário não contém a memória, ou seja, a extração de data obrigatoriamente deve falhar ao recuperar a memória. Alguns exemplos do *dataset* de perguntas e respostas podem ser vistos na Tabela II.

Para realizar os experimentos de forma a não enviesar a escolha de modelos e hiperparâmetros, o *dataset* de perguntas e respostas foi dividido em três partes:

- **Fold de Modelos**: 13 itens únicos que são usados para determinar que modelos generativos e *encoders* devem ser utilizados nos próximos passos;
- **Fold de Hiperparâmetros**: 13 itens únicos que são usados para determinar os hiperparâmetros do sistema;

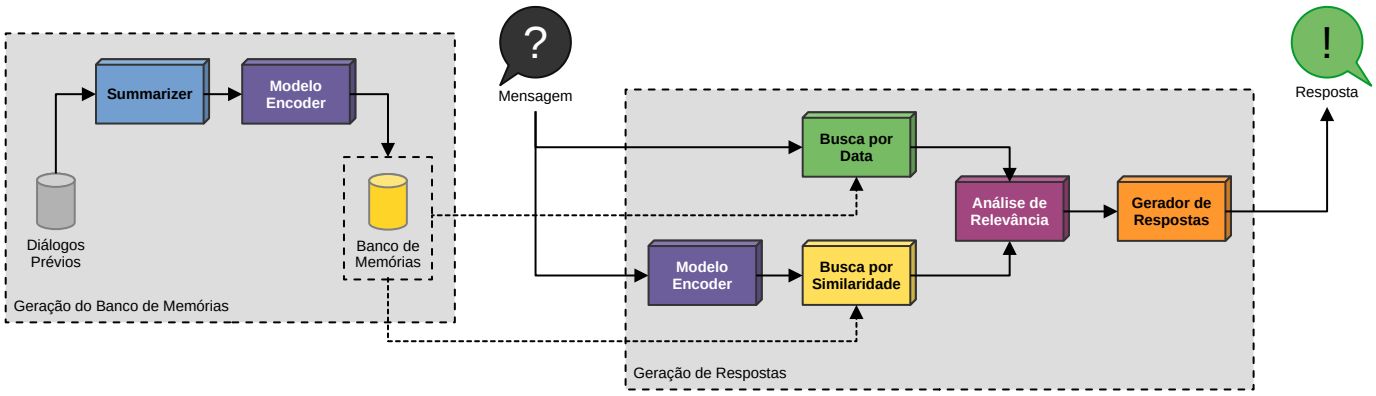

Fig. 1. Visão geral do sistema completo.

TABLE II
EXEMPLOS DE PERGUNTAS E RESPOSTAS

Pergunta: I remember you said something about gasoline last sunday.
Resposta: Yes. Taxi drivers were on strike to lower gasoline prices. **ID do diálogo**: 1 **Data correta?** Sim
Pergunta: About 12 months ago, I applied for a dish washer position in your restaurant, right?
Resposta: No. **ID do diálogo**: 359 **Data correta?** Não[a]

[a]Considerando a data de quando a pergunta foi feita, não fazem 12 meses que esta conversa aconteceu.

- **Fold de Teste**: 25 itens únicos que são usados para avaliar o sistema final.

B. Métricas

A avaliação de respostas de modelos generativos é comumente dita como um problema complicado em vários trabalhos [10]. Desta forma, métricas comuns, como Rouge [11], por exemplo, podem não ser suficientes para afirmar a qualidade da resposta de um modelo. Adicionalmente, como este trabalho explora a implementação de memória episódica, métricas que determinam se as memórias corretas foram encontradas podem melhor determinar se a abordagem escolhida é funcional.

1) Métricas para avaliação das respostas geradas: As métricas aplicadas neste trabalho para avaliar as respostas geradas pelos modelos generativos são: Rouge-L e BERTScore [12], com foco no *recall* e na *precision*, respectivamente.

$$\text{ROUGE-Recall} = \frac{|S \cap R|}{|R|} \tag{1}$$

$$\text{BERTScore-Precision} = \frac{1}{|S|} \sum_{s \in S} \max_{r \in R} \text{sim}(s, r) \tag{2}$$

Onde:

- S é o conjunto de palavras/n-gramas na predição
- R é o conjunto de palavras/n-gramas na referência.
- $|S \cap R|$ representa os elementos coincidentes entre predição e referência.
- $\text{sim}(s, r)$ é a similaridade de embeddings usada no BERTScore.

2) Métricas para avaliar a recuperação das memórias: Ademais, para avaliar os desempenhos da busca de memória comum (sem filtro de tempo) e busca de memória em um certo período de tempo, as métricas propostas são, respectivamente:

$$\frac{1}{N} \sum_{k=1}^{N} f(C', c_k)$$

$$\frac{1}{N} \sum_{k=1}^{N} g(C', c_k)$$

Onde:

- N é o número total de perguntas.
- c_k é o id da conversa correspondente a pergunta analisada.
- $C = \{c_0, c_1, \ldots, c_n\}$ é o conjunto de ids das conversas
- $C' \subset C$ é o conjunto de ids das conversas recuperadas para cada pergunta.

E as funções $f(C', c_k)$ e $g(C', c_k)$ são definidas abaixo:

$$f(C', c_k) = \begin{cases} 1 & \text{se } c_k \in C' \\ 0 & \text{caso contrário} \end{cases}$$

$$g(C', c_k) = \begin{cases} 1 & \text{se } c_k \in C' \text{ e a data correta} \\ 0 & \text{caso contrário} \end{cases}$$

C. Setup Experimental

Para realizar a busca de vetores, é empregado o FAISS (Facebook AI Similarity Search) [13], que promove uma busca rápida e eficiente devido ao uso da GPU para auxiliar no processo.

Para executar os experimentos, um computador equipado com uma GPU NVIDIA RTX 4090 com 24GB de VRAM foi utilizado para carregar as LLMs. Adicionalmente, para que os modelos testados pudessem rodar na máquina, a precisão foi reduzida para *float16*. A temperatura do modelo

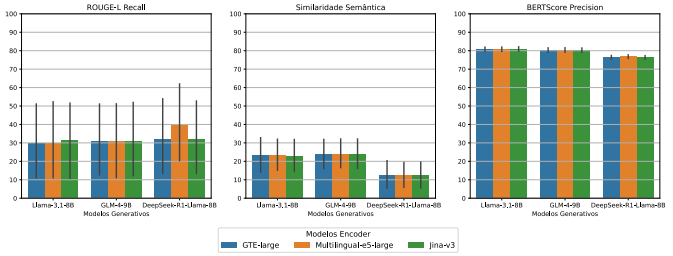

Fig. 2. Métricas aplicadas nas respostas geradas por cada configuração.

generativo foi configurada para *None* para manter os resultados determinísticos durante a experimentação. A penalidade por repetição (*repetition penalty*) e o número máximo de *tokens* novos (*max new tokens*) foram configurados para 1.1 e 1024, respectivamente.

O *GitHub* contendo o código e os *datasets* que foram utilizados está no seguinte link.

IV. Experimentos e Análises

Esta seção descreve os experimentos feitos, assim como apresenta uma breve discussão sobre os resultados obtidos. O método de busca gulosa foi aplicado para reduzir o tempo computacional necessário para executar todas as combinações possíveis de modelos e hiperparâmetros.

A. Experimento de Busca por Modelos

Com a grande gama de modelos disponíveis para o público, o primeiro passo torna-se reduzir o escopo por encontrar uma combinação de modelo generativo e *encoder* para ser experimentada posteriormente. Com isso, três modelos generativos e três modelos de *encoder* são testados no *Fold de Modelos*. Como os hiperparâmetros não serão configurados neste experimento, o número de vetores retornados pela busca semântica $k = 3$ e nenhum sumarizador é utilizado durante a fase de construção do banco de memórias. Os resultados das respostas podem ser vistos na Figura 2, em que não há muita discrepância entre os resultados. Entretanto, ao analisar a Figura 3, é possível ver o desempenho dos modelos durante o resgate de contextos relevantes para cada pergunta. Em geral, é importante notar que os modelos GLM 9B e Llama 8B obtiveram melhor resultado na extração de datas. Ademais, segundo a busca por data ou semântica, é possível ver um melhor desempenho tanto do DeepSeek 8B quanto do GLM 9B. Entretanto, durante a análise de relevância, o Llama 8B foi mais capaz de identificar os itens relevantes para a pergunta em questão. Em relação ao modelo de *encoder*, não houve discrepância significativa, entretanto, ao realizar a busca semântica, o *GTE-large* obteve melhor desempenho.

B. Experimento de Busca por Hiperparâmetros

Para encontrar um valor satisfatório de k e examinar se a aplicação de sumarização auxilia no processo de busca semântica, outro experimento é executado, agora no *Fold de Hiperparâmetros*. Adicionalmente, de acordo com as performances no experimento anterior, são utilizados os modelos Llama 8B e GTE-large como modelo generativo e *encoder*, respectivamente. O resultado da análise de respostas pode ser visto na Figura 4. Tais resultados mostram que o uso do sumarizador *RAKE* é tão eficiente quanto não utilizá-lo. Entretanto, segundo as pontuações mostradas na Figura 5, em uma ocasião específica, configurar $k = 9$ e utilizar o sumarizador *RAKE*, causou um aumento pequeno no desempenho das buscas.

C. Experimento Final

Levando em consideração as configurações encontradas anteriormente, o *Fold de Testes* foi utilizado para determinar e avaliar finalmente o desempenho do sistema. Com isso, as configurações testadas envolveram o uso do *encoder* GLM-large, $k = 9$ e o sumarizador *RAKE*. Como modelos generativos, foram testados ambos Llama 8B e GLM 9B para obter uma melhor visualização da importância na escolha de modelos generativos. O resultado das métricas aplicadas nas respostas geradas pode ser visto na Figura 6. Os gráficos mostram a superioridade das respostas geradas pelo Llama 8B, entretanto, isso pode ser devido à configuração de parâmetros do experimento anterior, que levou em consideração o Llama 8B como modelo generativo base. Ademais, o resultado das métricas aplicadas sobre a busca de memória pode ser observado visualmente na Figura 7. Nela, é possível observar que o Llama 8B possui melhor desempenho durante a extração de datas, assim como melhor desempenho durante a análise de relevância, que é um passo importante no sistema final. Em geral, o modelo Llama 8B corretamente extraiu 47.83% das datas, além de ter filtrado corretamente 52% das vezes as memórias que são relevantes para a pergunta do usuário.

V. Conclusões

Este trabalho apresentou uma possível implementação de um chat de conversação com memória episódica, em que o sistema, em hipótese, consegue resgatar memórias relevantes com base no tema e com base no período que é inferido pela mensagem do usuário.

Resultados mostram um sucesso razoável da implementação, com 52% de eficácia, mas promissor desempenho durante o resgate de contextos relevantes para o diálogo atual. Com isso, ao fazer mais reajustes, talvez seja possível integrar este sistema em um chat de conversação real, em que cada diálogo realizado também é adicionado ao banco de memórias, sendo possível retomar informações importantes trocadas nesta conversa em outras interações futuras com o modelo.

Futuramente, tal sistema pode ser mais explorado ao realizar *finetuning* ou utilizar de modelos mais robustos, com maior número de parâmetros.

References

[1] P. Lewis, E. Perez *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," 2021. [Online]. Available: https://arxiv.org/abs/2005.11401

[2] A. Dubey, A. Jauhri *et al.*, "The llama 3 herd of models," 2024. [Online]. Available: https://arxiv.org/abs/2407.21783

[3] T. GLM, A. Zeng *et al.*, "Chatglm: A family of large language models from glm-130b to glm-4 all tools," 2024.

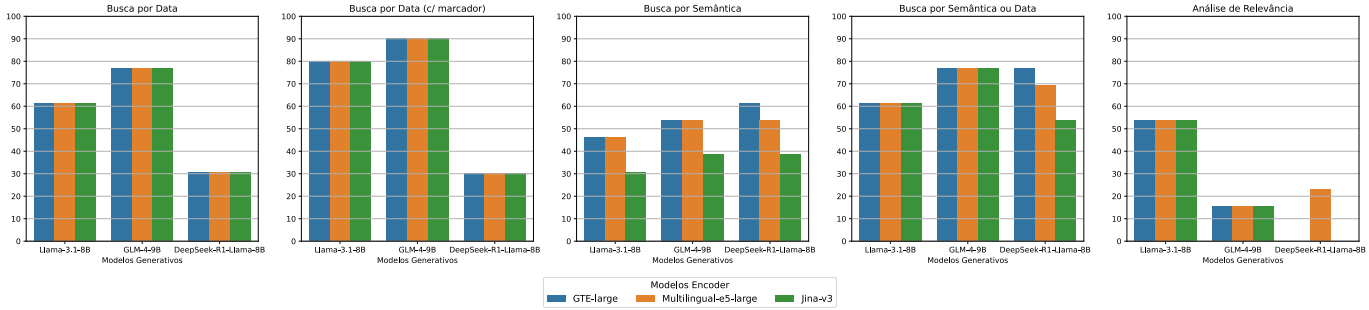

Fig. 3. Métricas aplicadas nos conjuntos de itens retornados durante a fase de busca de memórias.

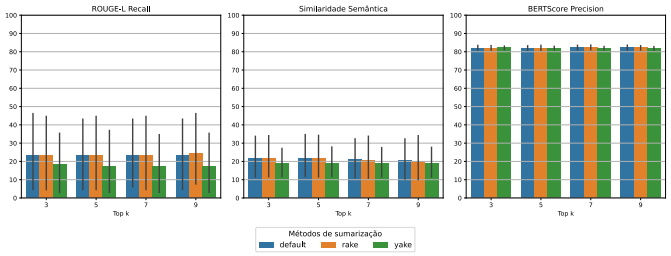

Fig. 4. Métricas aplicadas nas respostas geradas por cada configuração.

[4] DeepSeek-AI, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2501.12948

[5] Z. Li, X. Zhang *et al.*, "Towards general text embeddings with multi-stage contrastive learning," *arXiv preprint arXiv:2308.03281*, 2023.

[6] S. Sturua, I. Mohr *et al.*, "jina-embeddings-v3: Multilingual embeddings with task lora," 2024. [Online]. Available: https://arxiv.org/abs/2409.10173

[7] L. Wang, N. Yang *et al.*, "Multilingual e5 text embeddings: A technical report," *arXiv preprint arXiv:2402.05672*, 2024.

[8] R. Campos, V. Mangaravite *et al.*, "Yake! keyword extraction from single documents using multiple local features," *Information Sciences*, vol. 509, pp. 257–289, 2020.

[9] S. Rose, D. Engel, N. Cramer, and W. Cowley, "Rake: Rapid automatic keyword extraction," *Proceedings of the 2010 International Conference on Information and Knowledge Engineering (IKE)*, pp. 1–4, 2010.

[10] C. van der Lee, A. Gatt, E. van Miltenburg, and E. Krahmer, "Human evaluation of automatically generated text: Current trends and best practice guidelines," *Computer Speech & Language*, vol. 67, p. 101151, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S088523082030084X

[11] C.-Y. Lin and F. J. Och, "Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics," in *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, Barcelona, Spain, Jul. 2004, pp. 605–612. [Online]. Available: https://aclanthology.org/P04-1077

[12] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "Bertscore: Evaluating text generation with bert," 2020. [Online]. Available: https://arxiv.org/abs/1904.09675

[13] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

VI. Apêndice

A. Prompt de Data

Para extrair a data que está sendo inferida pela mensagem do usuário, o modelo é instruído a ler a analisar a mensagem, identificando quaisquer noções de tempo presentes. Ademais, alguns exemplos de mensagens e respostas aceitáveis são dados, utilizando da técnica de *few-shot* para auxiliar o modelo a interpretar o tempo presente nas mensagens.

Prompt de data

You'll receive a message from a user. Identify the date mentioned by the message based on today's date. Read the user's message and return a JSON with the period of time the user is refering to. If no time period is specified, DO NOT MAKE ONE UP.
Do not answer the user's message, only analyze the message to identify the time period it refers to and return the JSON.
The JSON must be formatted in the following format: { "dates" : [<items>] }, where
– Each item in <items> must be in the format { "start": <start_date>, "end": <end_date> }
– <start_date> marks the day in which the mentioned period starts
– <end_date> marks the day in which the mentioned period ends (if the period mentioned is longer than a day)
Please, format the dates only as following: month/day/year. And please, be sure that the date you mention corresponds to the weekday the user mentions, if any.

[exemplos de extração de datas]

Your turn! Return only the JSON with the dates.
Today's date: {today}
User:

B. Prompt de Relevância

Com os índices de memórias retornados pela busca de similaridade e de datas, o modelo agora deve ler e analisar cada conversa, determinando quais são relevantes para a mensagem do usuário.

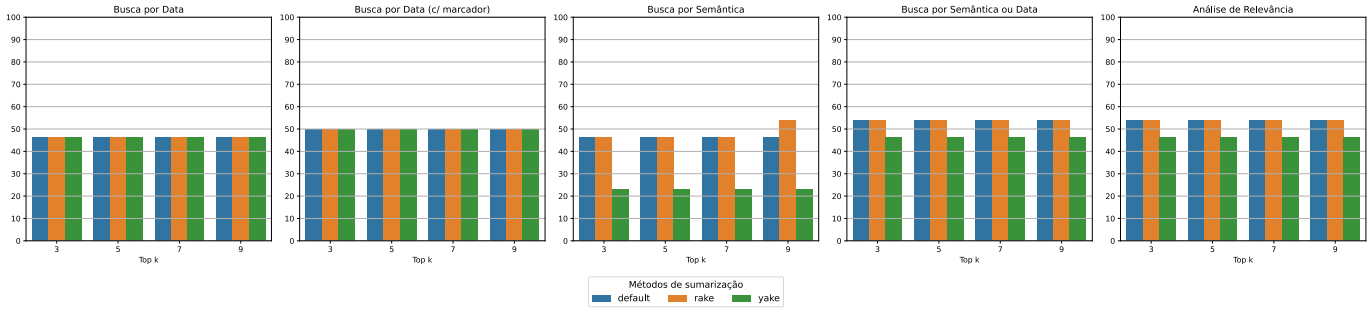

Fig. 5. Métricas aplicadas nos conjuntos de itens retornados durante a fase de busca de memórias.

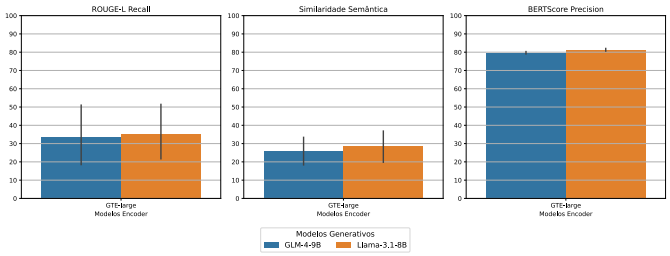

Fig. 6. Métricas aplicadas nas respostas geradas por cada configuração.

Prompt de relevância

You are going to receive some conversations between you and the user. Determine which conversations are relevant to answer the user's question givent their content and date.
At the end, return only a JSON with the indices of the relevant conversations.
Examples:
– if only conversations 2, 6 and 7 are necessary to answer the question, answer only with "{ "relevant_conversations": [2, 6, 7] }
– if no conversation is necessary to answer the question, answer only with "{ "relevant_conversations": [] }"

These are the conversations, with each one containing an index and the day they happened: {conversations}

Now, consider that today is {today}. Read the user's message below and determine which conversations are needed to respond to the user's message. Please respond only with the mentioned JSON format.
User:

C. Prompt de Chat

Após obter os índices filtrados, os quais representam as memórias que o modelo considera relevante para responder à mensagem atual do usuário, o modelo generativo é instruído a responder o usuário com base nestas conversações passadas, levando em consideração também a data atual.

Prompt de chat

You're an intelligent assistant. Interact with the user based on the previous conversations and today's date: {today}.
If the user's question is not related to a conversation, do not consider that conversation when answering the user. Do not mention anything that doesn't appear in the conversations.
Finally, analyze your answer to be sure your answer makes sense given the conversations below.
Here are the previous conversations between you and the user:
{conversations}
User:

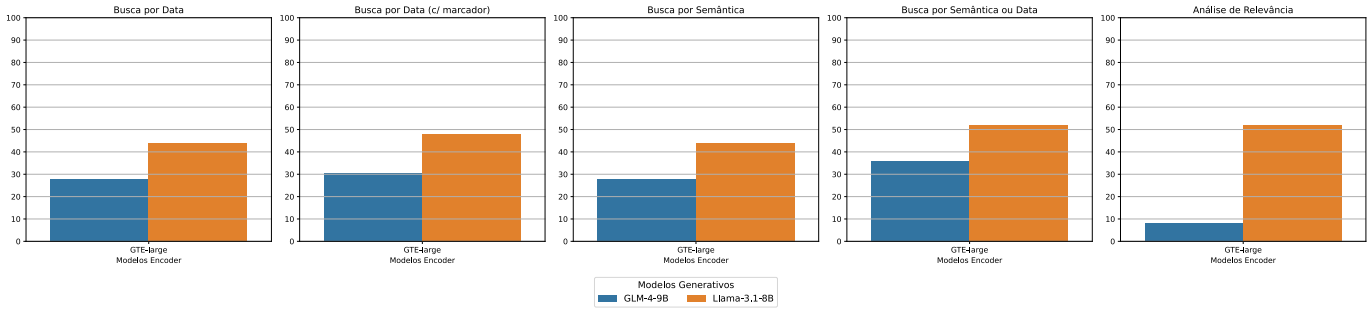

Fig. 7. Métricas aplicadas nos conjuntos de itens retornados durante a fase de busca de memórias.