Memória Episódica para Sistemas Baseados em GPT: Uma Implementação Prática

Lucas Matias Caetano
Programa de Pós-graduação em Informática (PPGI)
Universidade Federal do Espírito Santo (UFES)
Vitória, Espírito Santo, Brasil
lucas.caetano@edu.ufes.br

Abstract*—*This paper presents an implementation of **episodic memory** for GPT-based systems, enabling the model to retain and retrieve past experiences in a contextualized manner. The developed system uses **text embeddings** and a **vector database** (ChromaDB) to store temporal information, ensuring the relevance of retrieved memories in future interactions. The solution is built on three main pillars: (1) **temporal tagging**, (2) **efficient embedding storage**, and (3) **semantic retrieval**. The results demonstrate the feasibility of this approach for applications in chatbots, virtual assistants, and recommendation systems.

Keywords— Episodic Memory, GPT, Embeddings, ChromaDB, Natural Language Processing.

Resumo*—** Este artigo apresenta uma implementação de **memória episódica** para sistemas baseados em GPT, permitindo que o modelo retenha e recupere experiências passadas de forma contextualizada. O sistema desenvolvido utiliza ***embeddings de texto e um **banco de dados vetorial** (ChromaDB) para armazenar informações temporais, garantindo a relevância das memórias recuperadas em interações futuras. A solução aborda três pilares principais: (1) **tagueamento temporal**, (2) **armazenamento eficiente de embeddings**, e (3) **recuperação semântica**. Os resultados demonstram a viabilidade da abordagem para aplicações em *chatbots*, assistentes virtuais e sistemas de recomendação.

Palavras-chave: *Memória Episódica, GPT, Embeddings, ChromaDB, Processamento de Linguagem Natural.*

I. Introdução

A capacidade de sistemas de IA **lembrarem experiências passadas** é crucial para interações mais naturais e contextualizadas. No entanto, modelos como GPT tradicionalmente operam em um contexto estático, sem retenção de memórias entre sessões. Este trabalho propõe uma solução para esse desafio através de uma **memória episódica** que:

- **Armazena interações** com metadados temporais (data, hora, período do dia);
- **Recupera memórias relevantes** usando similaridade semântica.
- **Reinsere conhecimentos passados** no contexto atual de forma estruturada.

Inspirado em trabalhos como *Remembering Transformer* [3], o sistema combina técnicas de aprendizado contínuo com bancos de dados vetoriais, evitando o esquecimento catastrófico e otimizando a relevância das respostas.

II. Trabalhos correlatos

Diferentes Abordagens para memória em IA incluem:

- **Fine-tuning incremental** (Kirkpatrick et al., 2017): Limita-se a tarefas específicas e exige retreinamento.
- **Métodos baseados em memória** (Aljundi et al., 2017): Utilizam módulos especializados, mas com alto custo computacional.
- **Remembering Transformer** (Sun et al., 2024): Empregam adaptadores dinâmicos para reter conhecimento.

A solução proposta difere ao **integrar bancos de embeddings** (ChromaDB) com modelos de linguagem, oferecendo escalabilidade e baixo custo de armazenamento.

III. Metodologia

O sistema de memória episódica proposto é baseado em três componentes principais: **codificação temporal**, **armazenamento semântico** e **recuperação contextual**. A seguir, descrevemos a formulação matemática e o algoritmo proposto.

A. Codificação Temporal

Cada entrada de texto x_t no tempo t é associada a um vetor de metadados temporais τ_t:

$$\tau_t = (\text{timestamp}(t), \text{periodo\_dia}(t))$$

onde timestamp(t) é a data/hora da entrada, e periodo_dia(t) ∈ {manhã, tarde, noite} é determinado pela hora $h(t)$.

B. Armazenamento Semântico

- **Fragmentação do Texto**: Dado um texto x_t, divide-se em N segmentos $\{s_{t,1},...,s_{t,N}\}$, onde cada segmento $s_{t,i}$ tem no máximo L tokens (ex.: L=512).
- **Geração de Embeddings:** Cada segmento $s_{t,i}$ é mapeado para um *embedding* $e_{t,i} \in \mathbb{R}^d$ (ex.: d=384) usando um modelo de linguagem f_θ:

$$\mathbf{e}_{t,i}=f_\theta(s_{t,i})$$

- **Armazenamento no Banco Vetorial**: Cada tupla $(\mathbf{e}_{t,i}, \tau_t, s_{t,i})$ é armazenada em um índice de similaridade M.

- M suporta buscas por *k*-NN (vizinhos mais próximos) com base em distância de cosseno.

C. Recuperação Contextual

- **Consulta**: Para uma nova entrada x_q, gera-se um *embedding* de consulta $\mathbf{e}_q = f_\theta(x_q)$.
- **Busca por Similaridade**: Recuperam-se as *k* memórias mais relevantes de M com base em:

$$x \text{ similaridade}(\boldsymbol{e}_q, \boldsymbol{e}_{t,i}) = \frac{\boldsymbol{e}_q \cdot \boldsymbol{e}_{t,i}}{\|\boldsymbol{e}_q\| \|\boldsymbol{e}_{t,i}\|}$$

Opcionalmente, filtra-se por τ_t (ex.: apenas memórias da "tarde").

- **Fusão de Memórias**: As *k* memórias recuperadas $\left\{\left(\boldsymbol{e}_{\mathrm{t_j,i_j}}, \boldsymbol{\tau}_{\mathrm{t_j}}, \mathrm{s_{t_j,i_j}}\right)\right\}_{\mathrm{j=1}}^{\mathrm{k}}$ são agregadas em um contexto C_q:

$$C_q = \left\{\left(\boldsymbol{s}_{t_j,i_j}, \boldsymbol{\tau}_{t_j}, \text{similaridade}_j\right) \mid j = 1, \ldots, k\right\}$$

O contexto C_q pode ser posteriormente inserido no prompt do GPT para gerar uma resposta contextualizada.

D. Algoritmo Geral

- **Armazenar** (x_t): Para cada segmento $s_{t,i}$ de x_t:

$$\text{M.insert}(f_\theta(s_{t,i}), \tau_t, s_{t,i})$$

- **Recuperar**(x_q):
  - $\mathbf{e}_q \leftarrow f_\theta(x_q)$
  - $C_q \leftarrow \text{M.query}(\mathbf{e}_q, k)$
  - Retornar GPT($x_q \oplus C_q$)

IV. Experimentos

Os experimentos foram realizados a partir de um conjunto de testes, com 100 exemplos, gerado artificialmente pelo Modelo de Linguagem de Grande Escala (*Large Language Model* - LLM) **DeepSeek-V3**.

Esse conjunto de teste foi processado posteriormente pelo modelo de Embeddings de textos (f_θ) "paraphrase-multilingual-MiniLM-L12-v2" de 384 dimensões.

Os embeddings gerados foram salvos no banco vetorial (M) **ChromaDB,** otimizado para similaridade de cosseno.

O Código-fonte do sistema está disponível em https://github.com/lucasmatias/Episodic_Memory_for_GPT

A. Métricas

As métricas consideradas foram **Precisão@k,** que calcula a proporção de memórias recuperadas relevantes para x_q; **Recall@k,** que analisa a proporção de memórias relevantes recuperadas para x_q em relação ao total de memórias relevantes existentes; **Mean Reciprocal Rank (MRR),** que verifica a posição média da **primeira memória relevante** encontrada em uma lista de resultados; e a **AVG Similarity**, que mede a similaridade média (cosseno) entre as memórias recuperadas e a consulta.

V. Resultados

Conforme a **Tabela 01**, o valor baixo para **Precision@k (26%)** indica que muitas memórias irrelevantes são retornadas pelo modelo, sendo necessário ajustes no modelo de embeddings ou inclusão de filtros por metadados. Já o **Recall@k** moderado de 48% demonstra que o modelo encontra menos da metade das memórias relevantes, e o **MMR** baixo de 39% indica que o modelo não prioriza bem as memórias mais relevantes. Por fim, a **AVG Similarity** alcançou o valor de 54% nos testes, demonstrando que a memória tem uma relação moderada com a consulta, mas no limite para ruídos (quando o valor é abaixo de 50%)

Métrica	**Valor**
Precisão@3	26%
Recall@k	48%
MRR	39%
AVG Similarity	54%

VI. Conclusão

Com base nas métricas apresentadas, o sistema funciona, mas tem espaço significativo para melhorias. Como pontos positivos vale destacar o **Recall@k** razoável (48%) e uma base de testes (100 exemplos) diversificada, contudo, como pontos negativos, o **Precision@k** baixo (26%) evidencia que são retornados resultados irrelevantes da memória, o **MMR baixo (39%)** que os resultados relevantes retornados não aparecem nos primeiros resultados e o **AVG Similarity** moderado (54%) que as memórias recuperadas têm relação apenas parcial com a consulta.

Para trabalhos futuros, analisar melhorias no sistema a partir da aplicação de filtros na recuperação das memórias (por tópicos e data, por exemplo), fine-tuning do modelo de embeddings em dados específicos do domínio, e priorizar memórias com alta similaridade e uso recente (ex: adicionar um campo last_accessed e ponderar resultados).

References

[1] ALJUNDI, R. et al. Memory aware synapses: Learning what (not) to forget. Proceedings of the European Conference on Computer Vision (ECCV), p. 139-154, 2018.

[2] KIRKPATRICK, J. et al. Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, v. 114, n. 13, p. 3521-3526, 2017.

[3] SUN, Z. et al. Remembering Transformer: Adaptive memory integration for continual learning. Journal of Machine Learning Research, v. 25, p. 1-22, 2024.