



UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Vinicius Paulo Lopes de Oliveira

**MemoryGraph: uma proposta de memória para agentes conversacionais
utilizando grafo de conhecimento**

Goiânia
2024

17/10/24, 22:47

Sistema Eletrônico de Informações - Documento para Assinatura

Processo: 23070.042064/2024-35
Documento: 4906942



UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

**TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES
E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG**

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a [Lei 9.610/98](#), o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

Dissertação Tese Outro*: _____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Vinicius Paulo Lopes de Oliveira

3. Título do trabalho

MemoryGraph: uma proposta de memória para agentes conversacionais utilizando grafo de conhecimento

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento SIM NÃO¹

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:

- a) consulta ao(a) autor(a) e ao(a) orientador(a);
 - b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
- O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.



Documento assinado eletronicamente por Anderson Da Silva Soares, Professor do Magistério Superior, em 17/10/2024, às 10:54, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por Vinicius Paulo Lopes De Oliveira, Discente, em 17/10/2024, às 11:08, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador 4906942 e o código CRC 76A8AF37.

Vinicius Paulo Lopes de Oliveira

**MemoryGraph: uma proposta de memória para agentes conversacionais
utilizando grafo de conhecimento**

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação, do Instituto de Informática (INF), da Universidade Federal de Goiás (UFG), como requisito para obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação
Linha de pesquisa: Sistemas Inteligentes e Aplicações.

Orientador: Dr. Anderson da Silva Soares

Goiânia

2024

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Oliveira, Vinicius Paulo Lopes de
MemoryGraph: uma proposta de memória para agentes
conversacionais utilizando grafo de conhecimento [manuscrito] /
Vinicius Paulo Lopes de Oliveira. - 2024.
xvi, 167 f.

Orientador: Prof. Dr. Anderson da Silva Soares.
Tese (Doutorado) - Universidade Federal de Goiás, Instituto de
Informática (INF), Programa de Pós-Graduação em Ciência da
Computação, Goiânia, 2024.

Inclui siglas, abreviaturas, gráfico, tabelas, lista de figuras, lista
de tabelas.

1. Grafo de conhecimento. 2. Aprendizado de máquina. 3.
Processamento de linguagem natural. 4. Agente conversacional. 5.
Modelo de Linguagem Massivo. I. Soares, Anderson da Silva, orient.
II. Título.

CDU 004



UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
ATA DE DEFESA DE TESE

Ata nº 25 da sessão de Defesa de Tese de **Vinicius Paulo Lopes de Oliveira**, que confere o título de Doutor em Ciência da Computação, na área de concentração em Ciência da Computação.

Aos vinte e cinco dias do mês de setembro de dois mil e vinte e quatro, a partir das oito horas, via webconferência, realizou-se a sessão pública de Defesa de Tese intitulada “**Uma proposta de implementação de memória em agentes conversacionais utilizando grafo de conhecimento**”. Os trabalhos foram instalados pelo Orientador, Professor Doutor Anderson da Silva Soares (INF/UFG) com a participação dos demais membros da Banca Examinadora: Professora Doutora Nádia Felix Felipe da Silva (INF/UFG), membra titular interna; Professor Doutor Arlindo Rodrigues Galvão Filho (INF/UFG), membro titular interno; Professor Doutor Rodrigo Zempulski Fanucchi (COPEL Distribuição S.A.), membro titular externo; Professora Doutora Fabíola Souza Fernandes Pereira (UFU), membra titular externa. A realização da banca ocorreu por meio de videoconferência. Durante a arguição os membros da banca não fizeram sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Tese, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor Anderson da Silva Soares, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos vinte e cinco dias do mês de setembro de dois mil e vinte e quatro.

TÍTULO SUGERIDO PELA BANCA

MemoryGraph: uma proposta de memória para agentes conversacionais utilizando grafo de conhecimento



Documento assinado eletronicamente por **Vinicius Paulo Lopes De Oliveira, Discente**, em 25/09/2024, às 13:50, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Nadia Felix Felipe Da Silva, Professora do Magistério Superior**, em 25/09/2024, às 14:11, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Anderson Da Silva Soares, Professor do Magistério Superior**, em 25/09/2024, às 14:50, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Arlindo Rodrigues Galvao Filho, Professor do Magistério Superior**, em 25/09/2024, às 14:59, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **Fabíola Souza Fernandes Pereira, Usuário Externo**, em 25/09/2024, às 15:52, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



Documento assinado eletronicamente por **RODRIGO ZEMPULSKI FANUCCHI, Usuário Externo**, em 30/09/2024, às 21:30, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do [Decreto nº 10.543, de 13 de novembro de 2020](#).



A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4853547** e o código CRC **050DFEC9**.

RESUMO

Com o avanço dos modelos de linguagem massivos aplicados ao processamento de linguagem natural, muitas propostas se tornaram viáveis como, por exemplo, o uso de agentes conversacionais aplicado à diversas tarefas do cotidiano. No entanto, esses modelos ainda apresentam limitações tanto na integração de novos conhecimentos, quanto na representação e recuperação desse conhecimento, sendo limitados pelos custos, tempo de execução e treinamento. Além disso, sua característica de caixa-preta impede a manipulação direta do conhecimento, principalmente devido a representação vetorial que o representa de forma indireta, dificultando o controle e explicação de seus resultados. Em contraste, os grafos de conhecimento permitem uma representação rica e explícita das relações entre entidades do mundo real. Apesar dos desafios para sua construção, estudos indicam que estes podem se complementar para a produção de melhores resultados. Dessa forma, o objetivo desta pesquisa é propor um sistema de memória para agente conversacional baseado em modelos de linguagem massivos através da combinação de conhecimentos explícitos (grafos de conhecimento) e implícitos (modelos de linguagem) para melhor representação semântica e léxica deste. Essa metodologia foi chamada de MemoryGraph e é composto por três processos: construção do grafo, busca no grafo e representação do usuário. Foram propostos e comparados diversos fluxos de construção de grafo de conhecimento, considerando seus custos e influências no resultado final. O agente pode buscar informações nessa base através de diversas propostas de busca baseado em RAG, chamado aqui de GraphRAG. Esta metodologia de busca foi avaliada por humanos em cinco cenários de perguntas propostos, apresentando resultado médio superior nas cinco abordagens de buscas propostas (29% na melhor abordagem). Além destes, seis métricas de RAG, avaliadas por um modelo massivo, foram aplicadas ao resultado da aplicação proposta de dois datasets populares e um composto pelas diretrizes de diabetes, apresentando resultados superiores em todos os datasets. Além disso, foi proposto um método de representação do usuário a longo prazo, nomeada de *user_memory*, que demonstrou retenção de 93% das informações sobre o usuário. Para reforçar esse resultado, foram realizados estudos de caso que demonstram a capacidade do agente em personalizar a experiência do usuário baseado em experiências passadas, aumentando a velocidade de entrega da informação e satisfação do usuário. Os resultados demonstram que o paradigma do MemoryGraph representa um avanço em relação a representação vetorial em ambientes onde se faz necessário uma contextualização mais rica, temporal e mutável. Indicando, também, que a integração de grafos de conhecimento com modelos de linguagem massivos, especialmente na construção da memória de longo prazo e rica contextualização baseada em experiências passadas, pode representar um avanço significativo na criação de agentes conversacionais mais eficientes, personalizados e com uma capacidade aprimorada de retenção e utilização de informações ao longo do tempo.

Palavras – chave: Grafo de conhecimento; Aprendizado de máquina; Processamento de linguagem natural, Agente Conversacional, Modelo de Linguagem Massivo.

ABSTRACT

With the advancement of massive language models applied to natural language processing, many proposals have become viable, such as the use of conversational agents applied to various everyday tasks. However, these models still have limitations in both the integration of new knowledge and the representation and retrieval of that knowledge, being constrained by costs, execution time, and training. Furthermore, their black-box nature prevents the direct manipulation of knowledge, mainly due to the vector representation that indirectly represents it, making the control and explanation of their results more difficult. In contrast, knowledge graphs allow for a rich and explicit representation of relationships between real-world entities. Despite the challenges in their construction, studies indicate that these can complement each other to produce better results. Therefore, the objective of this research is to propose a memory system for conversational agents based on massive language models through the combination of explicit knowledge (knowledge graphs) and implicit knowledge (language models) to achieve better semantic and lexical representation. This methodology was called MemoryGraph and is composed of three processes: graph construction, graph search, and user representation. Various knowledge graph construction workflows were proposed and compared, considering their costs and influences on the final result. The agent can search for information in this base through various search proposals based on RAG, referred to here as GraphRAG. This search methodology was evaluated by humans in five proposed question scenarios, showing superior average results in all five proposed search approaches (29% in the best approach). In addition, six RAG metrics, evaluated by a massive model, were applied to the proposed application results from two popular datasets and one composed of diabetes guidelines, showing superior results in all datasets. Furthermore, a method for long-term user representation, called user_memory, was proposed, demonstrating 93% retention of user information. To reinforce this result, case studies were conducted, demonstrating the agent's ability to personalize the user experience based on past experiences, increasing the speed of information delivery and user satisfaction. The results demonstrate that the MemoryGraph paradigm represents an advance over vector representation in environments where richer, temporal, and mutable contextualization is necessary. It also indicates that the integration of knowledge graphs with massive language models, especially in the construction of long-term memory and rich contextualization based on past experiences, can represent a significant advance in creating more efficient, personalized conversational agents with enhanced capacity for retaining and utilizing information over time.

Keywords: Knowledge graph; Machine learning; Natural language processing; Conversational agent; Massive language models.

LISTA DE FIGURAS

Figura 1. Representação de um neurônio recorrente.....	18
Figura 2. Representação da arquitetura Transformer.....	19
Figura 3. Exemplo de utilização de janela de contexto para geração de <i>dataset</i> para treinamento de <i>word embedding</i>	21
Figura 4. Representação da arquitetura bidirecional utilizada no ELMo.....	22
Figura 5. Acurácia de diferentes modelos baseados em <i>Transformers</i> na tarefa de pergunta e resposta conforme a quantidade de dados utilizados no treinamento.....	23
Figura 6. Representação da capacidade do modelo T5 em ser treinado em multitarefas.....	25
Figura 7. Representação das etapas contidas na técnica de <i>Reinforcement Learning from Human Feedback</i>	28
Figura 8. Representação da diferença entre paradigmas de modelos de linguagem apresentado pelo RETRO.....	31
Figura 9. Abordagem geral do pipeline de Retrieval-Augmented Generation (RAG) para aplicações baseadas em Large Language Models (LLM).....	34
Figura 10. Diferentes arquiteturas que podem ser utilizadas para busca semântica (<i>retriever</i>).....	36
Figura 11. Representação de um exemplo em que a pesquisa por similaridade retorna informações duplicadas, enquanto outras informações relevantes podem ser ignoradas em decorrência de um baixo número K (três, neste caso) ou à distância semântica que pode não ser grande o suficiente para entrar no top K.....	44
Figura 12. Representação simplificada de um agente inteligente e sua interação entre seus componentes principais.....	46
Figura 13. Representação das métricas contidas na <i>Automated Evaluation of Retrieval Augmented (RAGAS)</i>	53
Figura 14. Exemplo de grafo de conhecimento.....	59
Figura 15. Representação de um exemplo de um documento segmentado e transformado em um grafo de conhecimento.....	61
Figura 16. Comparação entre as vantagens e desvantagens dos grafos de conhecimento (KG) e dos modelos de linguagem em grande escala (LLM).....	63
Figura 17. Comparação entre grafo direcional (RDF) e grafo de propriedade.....	65
Figura 18. Exemplo de inferência na tarefa de <i>Named Entity Recognition (NER)</i>	67
Figura 19. Exemplo de <i>pipeline</i> de <i>Named Entity Recognition (NER)</i> baseado em aprendizado profundo (<i>deep learning</i>).....	72
Figura 20. Exemplo de ambiguidade léxica e associação de entidades.....	73
Figura 21. Exemplo de desambiguação da entidade “pessoa” através do <i>link</i> com uma página <i>wiki</i>	74
Figura 22. Representação da modelagem das camadas de abstração do conhecimento.....	84
Figura 23. Exemplo de como as entidades se relacionam entre as camadas de abstração.....	85
Figura 24. Exemplo de como a taxonomia agrega à entidade uma nova camada de conhecimento geral.....	90
Figura 25. Representação de exemplo de subgrafo da estrutura hierárquica do documento em que este tem relação com todos os seus elementos e relação sequencial entre si.....	92

Figura 26. Fluxo de extração de informação de documentos e inserção na base de conhecimento.	93
Figura 27. Informações do parágrafo do texto sobre o descobrimento do Brasil transformado em um grafo.	94
Figura 28. Representação de exemplo de atualização da memória de curto prazo durante a conversa com o agente.	96
Figura 29. Representação da incorporação da experiência na memória de longo prazo.	98
Figura 30. Exemplo de uma conversa com os três tipos de informações extraídas: informação do usuário, resumo da conversa e conhecimento geral.	99
Figura 31. Exemplo de como é formatado a saída do grafo de conhecimento para utilização de contexto do agente.	102
Figura 32. Fluxo de execução do agente.	104
Figura 33. Gráfico de crescimento do número de nós a partir da variação do número de documentos inseridos.	112
Figura 34. Gráfico de crescimento do número de relações a partir da variação do número de documentos inseridos.	113
Figura 35. Gráfico de crescimento do número de <i>labels</i> dos nós a partir da variação do número de documentos inseridos.	113
Figura 36. Gráfico de crescimento do número de tipos de relações a partir da variação do número de documentos inseridos.	114
Figura 37. Gráfico demonstrando o tempo necessário para processamento e construção do grafo de conhecimento.	114
Figura 38. Exemplo de conversa entre o paciente fictício e o agente sem a utilização de <i>user_memory</i>	138
Figura 39. Exemplo da conversa descrita na figura 38 utilizando <i>user_memory</i> extraída das primeiras 30 primeiras conversas do agente com o paciente fictício.	139

LISTA DE QUADROS

Quadro 1. Paradigmas de aprendizado em processamento de linguagem natural...	26
Quadro 2. <i>Datasets</i> de <i>Named Entity Recognition</i> (NER) existentes na língua inglesa até o ano de 2020.	68
Quadro 3. Exemplo de anotação para os padrões IOB2 e IOBES.	69
Quadro 4. Ferramentas acadêmicas e industriais com modelos de <i>Named entity recognition</i> (NER) pré-treinados.	70
Quadro 5. Comparação entre diferentes técnicas de extração de relações.	77
Quadro 6. Comparação da busca local e global para <i>query</i> “qual a relação entre a Doença A e o Medicamento X?” considerando o processo, tipo de resultados esperados e exemplos possíveis de saída.	79
Quadro 7. Resultado da busca no Neo4J para a <i>Cypher query</i> apresentada.	101

LISTA DE TABELAS

Tabela 1. Exemplos de entradas do <i>dataset</i> contendo pergunta, resposta esperada e fontes utilizadas para a resposta apresentada.	111
Tabela 2. Estado final do grafo de conhecimento em cada cenário de inserção.....	115
Tabela 3. Custo necessário utilização do GPT-4o por dia via API da OpenAI para o cenário de 20.000 páginas por dia.	119
Tabela 4. Avaliação humana do resultado das buscas em diferentes cenários utilizando os dados do dataset diretrizes-diabete.....	125
Tabela 5. RAG vs GraphRAG nos datasets diretrizes-diabetes, RAGBench e Natural Questions.	129
Tabela 6. Exemplo de pergunta com a resposta esperada e a avaliação da métrica Answer Correctness para a execução com RAG e GraphRAG.....	132
Tabela 7. Resultado das perguntas simples realizadas utilizando <i>user_memory</i> . ..	134
Tabela 8. Resultado das perguntas complexas realizadas utilizando <i>user_memory</i>	135

LISTA DE ABREVIATURAS E SIGLAS

AI	Agente Inteligente
AIC	Agente Inteligente Conversacional
AM	Aprendizado de Máquina
AP	Aprendizado Profundo
AS	Aprendizado Supervisionado
ANS	Aprendizado Não Supervisionado
AR	Aprendizado Por Reforço
BTT	<i>Backpropagation Through Time</i>
BLEU	<i>BiLingual Evaluation Understudy</i>
CBOW	<i>Continuous Bag Of Words</i>
CoQA	<i>Conversational Question Answering</i>
DNLI	<i>Dialogue Natural Language Inference</i>
EI	Extração de Informação
EL	<i>Entity Linking</i>
ELMo	<i>Embedding from Language Model</i>
EM	<i>Exact Match</i>
EN	Entidade Nomeada
ER	Extração de Relação
GC	Grafo de Conhecimento
GRU	<i>Gated Recurrent Unit</i>
IA	Inteligência Artificial
IR	<i>Information Retrieval</i>
IRI	<i>Internationalized Resource Identifiers</i>
DL	<i>Deep Learning</i>
QA	<i>Question Answering</i>
KG	<i>Knowledge Graph</i>
LLM	<i>Large Language Model</i>
LoRA	<i>Low-Rank Adaptation</i>
LSTM	<i>Long Short Term Memory</i>
RC	Rede Convolucional
RDF	<i>Resource Description Framework</i>
RN	Rede Neural
RNA	Rede Neural Artificial
ROUGE	<i>Recall-Oriented Understudy for Gisting Evaluation</i>
ML	Modelos de Linguagem
MRR	<i>Mean Reciprocal Rank</i>
MS MARCO	<i>Microsoft Machine Reading Comprehension</i>
MultiRC	<i>Multiple-Choice Reading Comprehension</i>
NER	<i>Named Entity Recognition</i>
NLP	<i>Natural Language Processing</i>
PLN	Processamento de Linguagem Natural
PPO	<i>Proximal Policy Optimization</i>
PR	Pergunta e Resposta
QuAC	<i>Question Answering in Context</i>
RAG	<i>Retrieval-Augmented Generation</i>
RAGAS	<i>Automated Evaluation of Retrieval Augmented Generation</i>

RLHF	<i>Reinforcement Learning from Human Feedback</i>
RR	Redes Recorrentes
RDF	<i>Resource Description Framework</i>
RS	Redes Semânticas
SAS	<i>Semantic Answer Similarity</i>
SBERT	Sentence-BERT
<i>seq2seq</i>	<i>Sequence To Sequence</i>
SQuAD	Stanford Question Answering Dataset
SVM	<i>Support Vector Machine</i>
TDNN	<i>Time-Delay Neural Networks</i>
TFIDF	<i>Term Frequency–Inverse Document Frequency</i>
URI	<i>Uniform Resource Identifier</i>
WE	<i>Word embedding</i>
WOL	<i>Web Ontology Language</i>

AGRADECIMENTOS

Agradeço a minha esposa, por me suportar em todos os momentos.

Agradeço aos meus pais, por me concederem a oportunidade de chegar aonde estou.

Agradeço ao meu orientador, por confiar e apostar na minha proposta e carreira desde o início.

E agradeço ao Instituto de Informática da Universidade Federal de Goiás, ao Centro de Excelência em Inteligência Artificial e às empresas e órgãos parceiros, como Mr.Turing, Volkswagen, Americas Health, TCM, FUNAPE, COPEL e Embrapii, pelo apoio estrutural e financeiro aos projetos de pesquisa dos quais participei.

SUMÁRIO

1	INTRODUÇÃO	1
2	OBJETIVOS	13
2.1	Objetivo geral	13
2.2	Objetivos específicos	13
	Organização do Documento	14
3	A LINGUAGEM NATURAL COMO UMA INTERFACE PARA O CONHECIMENTO	15
3.1	Aprendizado de máquina no processamento de linguagem natural.....	16
3.2	Modelando a linguagem natural	20
3.3	<i>Retrieval-Augmented Generation</i> (RAG) nos Sistemas de Pergunta e Resposta (<i>Question-Answering</i>)	33
3.5	Agentes Conversacionais Inteligentes (AI Agents)	45
3.6	Avaliação de Pergunta e Resposta, <i>Retrieval-Augmented Generation</i> (RAG) e Agentes Inteligentes.....	49
4	GRAFO DE CONHECIMENTO	56
4.1	Grafo de conhecimento para codificação do conhecimento.....	56
4.2	Construção do grafo de conhecimento	63
4.2.1	Extração de Entidades.....	66
4.2.2	Extração de Relações	75
4.3	GraphRAG	77
5	METODOLOGIA.....	81
5.1	Codificação e Armazenamento do conhecimento	81
5.1.1	Camadas de abstração do conhecimento	82
5.1.2	<i>Pipelines</i> de incorporação de conhecimento	86
5.2	Recuperação de informação	100
6	EXPERIMENTOS, RESULTADOS E DISCUSSÃO	109
6.1	Construção do grafo de conhecimento e incorporação do conhecimento na memória de longo prazo.	109
6.2	Contextualização do agente utilizando grafo de conhecimento – acesso a memória	121
6.2.1	Avaliação Humana.....	121
6.2.2	Avaliação baseada em LLM.....	127
6.3	<i>user_memory</i> no contexto do agente.....	133
7	CONSIDERAÇÕES FINAIS	140
8	CONCLUSÃO.....	142
	REFERÊNCIAS.....	144

1 INTRODUÇÃO

A Inteligência Artificial (IA), um campo emergente dentro das Ciências da Computação, busca replicar a capacidade humana de raciocínio e resolução de problemas através de máquinas (RUSSEL, 2003). Com o crescimento exponencial do volume de dados disponíveis e o avanço dos recursos computacionais, tornou-se possível desenvolver algoritmos capazes de aprender com esses dados, originando o conceito de Aprendizado de Máquina (Machine Learning - ML). Dentro de ML, um dos métodos mais proeminentes é o uso de Redes Neurais Artificiais (RNAs), estruturas inspiradas no cérebro humano que consistem em camadas de neurônios artificiais interconectados (BISHOP, 1996). Essas redes são especialmente eficazes em identificar padrões complexos e realizar tarefas como reconhecimento de imagem e processamento de linguagem natural. Com o aumento da capacidade computacional e a disponibilidade de grandes conjuntos de dados, as RNAs evoluíram para modelos mais profundos e sofisticados, dando origem ao Aprendizado Profundo (Deep Learning - DL) (GOODFELLOW; BENGIO; COURVILLE, 2016). DL expande as capacidades das RNAs, permitindo o treinamento de redes com muitas camadas, conhecidas como redes neurais profundas, que são capazes de lidar com tarefas altamente complexas e fornecer resultados impressionantes em áreas como visão computacional, tradução automática e análise preditiva.

O DL foi responsável pelo avanço do estado da arte da maioria das áreas da computação relacionadas à predição como, por exemplo, visão computacional, processamento de linguagem natural e processamento de áudio (LeCUN; BENGIO; HINTON, 2015). O DL possibilitou, por exemplo, sintetizar voz de forma quase indiferenciável de uma pessoa (PRENGER; VALLE, CATANZARO, 2018; REN et al., 2020; DONAHUE et al., 2020), gerar rostos de pessoas de forma realista (GECER et al., 2020), identificar comportamentos em vídeos através da análise de objetos no tempo (ZHU et al., 2020), rastrear o movimento de centenas de pessoas simultaneamente (TANG et al., 2016), responder a perguntas complexas (BAKKER, 1987), traduzir simultaneamente em diversos idiomas (PYTORCH FAIRSEQ, 2020) e extrair sentimentos (DANG; MORENO-GARCIA; DE LA PRIETA, 2020).

Dentre os avanços possibilitados, ressalta-se que o processamento de linguagem natural (NLP, do inglês natural *language processing*) vem se destacando

porque a linguagem é uma parte fundamental da inteligência humana na medida em que permite às pessoas comunicar pensamentos e conceitos, expressar ideias, criar memórias e construir compreensão mútua. O NLP é uma área com ênfase nas interações entre computadores e linguagem humana por meio da compreensão de como o computador entende, processa e gera linguagem natural. Além de permitir a realização de várias tarefas baseadas em texto, como apresentado anteriormente, o NLP possibilita uma interação mais acessível entre humanos e computadores, permitindo um acesso maior e mais rápido ao conhecimento.

Um dos passos mais importantes em direção a um modelo com alta capacidade de compreensão linguística se deu pela arquitetura Transformer, introduzida em 2017 pelo artigo "*Attention is All You Need*", revolucionando o campo de NLP. Antes dos Transformers, modelos como LSTMs (*Long Short-Term Memory*) e GRUs (*Gated Recurrent Units*) dominavam o cenário, mas enfrentavam limitações ao lidar com dependências de longo alcance em sequências de texto. Os Transformers superaram essas limitações ao introduzir o mecanismo de *self-attention*, que permite que o modelo avalie a importância de cada palavra em uma sentença em relação a todas as outras, independentemente da distância entre elas. Isso é fundamental para capturar o contexto global e nuances de significado que são cruciais para a compreensão e geração de linguagem natural (VASWANI et al., 2017).

Os Transformers possibilitaram o desenvolvimento de modelos de linguagem como BERT (DEVLIN et al., 2018), GPT (BROWN et al., 2020) e T5 (RAFFEL et al., 2019), que definiram novos padrões de desempenho em tarefas como tradução automática, resumo de texto, geração de texto e pergunta e resposta. Um modelo de linguagem pode ser definido como um sistema de inteligência artificial treinado para compreender e gerar texto em linguagem natural. Ele aprende padrões e estruturas de grandes quantidades de dados textuais através da tarefa de predição da próxima palavra, o que exige do modelo a compreensão linguística do idioma, resultando em um modelo que representa aquela linguagem. Ao permitir que modelos sejam treinados em grandes quantidades de dados não supervisionados, os Transformers facilitaram o aprendizado de representações de linguagem extremamente ricas, permitindo que modelos de linguagem capturassem o contexto e o significado das palavras de maneira mais rica e eficiente, permitindo resultados mais coerentes e relevantes em uma ampla variedade de aplicações de NLP (VASWANI et al., 2017).

O aumento exponencial das capacidades de processamento e consumo de dados dos Transformers, levou ao desenvolvimento de modelos poderosos que podem ser utilizados como modelos base para solução de diversas tarefas de NLP. Isso levou a expansão do termo e da área de estudo de modelos de linguagem massivos (do inglês, *Large Language Models* - LLMs) (BROWN et al., 2020) e viabilizaram tarefas como, por exemplo, escrita e correção de código de programação (Microsoft co-pilot) (LI et al., 2023), aplicação práticas em ensino e educação (MALINKA et al., 2023) e robustos sistemas de recomendação (LIU et al., 2023).

Em novembro de 2022, o ChatGPT foi introduzido pela OpenAI e este foi apontado como o próximo passo em modelos de NLP pela sua alta performance em diversas tarefas de NLP, demonstrando a capacidade de compreensão e geração de linguagem de uso geral. Essa capacidade foi adquirida através do treinamento de um modelo Transformer, baseado no GPT-3, contendo bilhões de parâmetros e grandes quantidades de dados de texto (BROWN et al., 2020). Desde então, vários modelos LLM e técnicas relacionadas com diversas propostas foram apresentados, incluindo modelos focados em redução de tamanho de parâmetros, técnicas de *fine-tuning* mais eficientes e métodos de quantização dos pesos. Minaee et al. (2024) apresenta um *survey* que coleta cinquenta modelos oficiais publicados de LLM até fevereiro de 2024 e propõe três famílias de LLMs: GPT, LLaMA e PaLM.

Os sistemas conversacionais baseados em LLM se tornaram uma ferramenta de uso frequente no cotidiano de muitas pessoas, atuando como uma ferramenta central para resolver diversos tipos de problemas como, por exemplo, solucionar dúvidas, sumarizar textos, traduzir para diversos idiomas, corrigir e formatar textos, gerar e corrigir códigos e, até mesmo, gerar conteúdos (BOMMASANI et al., 2021). Porém, além do custo de treinamento, tamanho do modelo, LLMs tem seus conhecimentos restringidos aos dados que foram utilizados durante o treinamento. Dificultando a aplicação dessas ferramentas em um ambiente mais onde a informação é mais restrita como em empresas e corporações que possuem grande quantidade de dados sensíveis e com acesso privativo. Além disso, esse extenso volume de texto e dados, comumente não estruturados, tende a crescer diariamente e com isso, a necessidade de processamento e extração de informações desses dados (GANTZ; REINSEL, 2012). Como resultado, observa-se o crescimento continuado do tamanho dos modelos (número de parâmetros) e a quantidade de dados utilizados em seus treinamentos. Notadamente, pode-se citar o modelo GPT-3, que em 2020 possuía 175

bilhões de parâmetros e em sua atualização, o modelo GPT-4, passou a contar com mais de 1.76 trilhões de parâmetros e treinamento em uma base extraída de toda a internet e complementado com feedback humano, totalizando o custo estimado de 100 milhões de dólares (OPENAI, 2023). Assim, surge a preocupação com a acessibilidade desses modelos que demandam altos recursos, o que pode ser um obstáculo para a maioria das empresas e levar a um monopólio dos modelos mais avançados por parte das grandes corporações. Além desse questionamento, também há incertezas sobre a real capacidade de aprendizado e generalização desses modelos: se o resultado do modelo é realmente o aprendido, se o modelo está compreendendo a pergunta e procurando uma resposta em seu conhecimento paramétrico ou se está apenas memorizando o que foi apresentado durante o treinamento (MARCUS, 2020).

Ademais, o alto impacto ambiental gerado pelo treinamento desses modelos também é considerado uma preocupação. Em um estudo realizado em 2019, pesquisadores da Universidade de Massachusetts em Amherst estimaram que o treinamento de um grande modelo de aprendizado profundo produz 283.948.824 quilogramas de dióxido de carbono que aquecem o planeta de forma equivalente às emissões de cinco carros durante toda a sua vida útil (STRUBELL; GANESH; MCCALLUM, 2019). Apesar dessas estimativas e iniciativas para a redução do tamanho dos modelos que consideram estudos que demonstram que o uso de modelos especialistas podem ser uma solução para o mundo corporativo, há ainda grande necessidade de recursos humanos capacitados e de infraestrutura, o que é um impeditivo para a maioria das empresas terem seus próprios modelos de LLMs (OPENAI, 2023).

Assim, vê-se que, apesar das LLMs serem uma tecnologia essencial para solução de tarefas de NLP, se faz necessários caminhos para viabilizar essa tecnologia em ambientes mais diversos e que necessitam da mesma. Uma opção para essa viabilização seria o *fine-tuning*, uma técnica já concretizada e utilizada para ajuste de modelos para incorporação de novos conhecimentos ou especialização em tarefas. Essa técnica envolve processos nos quais um modelo que já foi treinado para uma determinada tarefa é ajustado para que ele possa executar uma segunda tarefa semelhante a primeira. Recentemente, novas técnicas para aumentar a eficiência do processo de *fine-tuning*, reduzindo o tempo e o custo desse processo, vem sendo desenvolvidas como, por exemplo, o LoRA (*Low-Rank Adaptation*), que atualiza

apenas uma pequena parte dos pesos do modelo, visando especificamente aqueles que têm o impacto mais significativo na tarefa em questão (HU et al., 2021). Porém, a técnica de *fine-tuning* ainda exige criação de datasets e carece de recursos humanos capacitados e de infraestrutura, mesmo considerando uma escala menor que o treinamento completo.

As principais provedoras de modelos de LLM apresentam ferramentas facilitadoras para o uso de *fine-tuning* ao utilizar os modelos das mesmas dentro de suas infraestruturas, reduzindo consideravelmente as barreiras para isto, como a OpenAI (OPENAI, 2024). No entanto, ao observar o cotidiano de empresas em que novas informações são geradas diariamente por meio de reuniões, contratos, manuais e outras interações e processos, os modelos podem se tornar obsoletos em poucos dias. Isso indica que o processo de criação de dataset e execução do *fine-tuning* a cada novo documento é um processo lento, com altos custos envolvidos e não escalável à rotina da maioria das empresas (DENG; LIU, 2018). Além disto, a abordagem do *fine-tuning* apresenta uma importante limitação devido a uma característica inerente os modelos de LLMs: a incapacidade de verificar a fonte da informação utilizada pelo modelo de LLM na geração de uma resposta. Mesmo os modelos mais expressivos e poderosos ainda estão suscetíveis a alucinações, ou seja, podem gerar informações falsas. Esse cenário pode ser crítico em alguns contextos como, por exemplo, em agentes de suporte a saúde (OPENAI, 2023; SRIVASTAVA et al., 2022).

Considerando esse cenário, estudos vêm atestando a capacidade dos modelos de LLMs de *fewshot learning* que podem produzir continuações coerentes sobre uma ampla gama de tópicos após a exposição à apenas alguns exemplos, demonstrando sinais de compreensão e raciocínio flexíveis. Contudo, análises mais recentes e aprofundadas revelam lacunas fundamentais nesses modelos. Fundamentalmente, os modelos de LLMs acumulam apenas padrões estatísticos que refletem correlações de termos apresentados a ela durante o treinamento. Apresentando, assim, a falta de quaisquer representações estruturadas ou modelação explícita de conceitos, relações ou regras. Como resultado, cometem frequentemente erros lógicos básicos, falhando na generalização de informações e tarefas que estão além da distribuição dos dados de treinamento e apresentam, também, dificuldades em sustentar sequências de raciocínios coerentes (BROWN et al., 2020).

Outra técnica importante neste cenário é o *prompt learning*, uma técnica que envolve a adaptação de modelos de linguagem pré-treinados para novas tarefas utilizando instruções ou "*prompts*". Ao invés de ajustar o modelo através de um processo intensivo de retreinamento, o *prompt learning* aproveita as capacidades já presentes no modelo fundamental, guiando-o a realizar tarefas específicas através de pequenos ajustes na entrada. Essa abordagem reduz significativamente o tempo e os recursos necessários para adaptar o modelo a diferentes aplicações, permitindo que ele seja eficaz em uma ampla gama de tarefas com mínimos esforços adicionais. O *prompt learning* representa uma evolução na forma como os modelos de linguagem podem ser reutilizados e refinados, destacando-se como uma solução eficiente para expandir as capacidades de IA em contextos variados (LIU et al., 2021). Porém, essa técnica apesar de ajudar no processo de adaptabilidade a novos cenários e controle dos LLMs, ele não resolve o problema da inserção e atualização do conhecimento da rede.

Considerando a necessidade de inserção e atualização rápida de informações, redução de alucinações dos modelos de LLMs e controle e transparência das fontes de informação utilizadas por eles, surge o conceito de RAG (do inglês *Retrieval-Augmented Generation*). O RAG é um processo de armazenamento e filtragem de fontes de informações relevantes para a geração dos modelos de LLMs, o que possibilita a solução das limitações e necessidades apresentadas anteriormente (LEWIS et al., 2020). A combinação entre o RAG e os modelos de LLM permitiu a ampliação do uso dessa tecnologia em uma maior gama de problemas e cenários, viabilizando às empresas menores a criação de suas próprias bases de conhecimento baseado em dados de acesso restrito e combinado à capacidade do LLMs providas por grandes empresas sem a necessidade de alterá-las ou mantê-las, reduzindo significativamente o esforço inicial para adoção dessas ferramentas e permitindo a solução de problemas relativos a seus dados (GAO et al., 2024).

A utilização de acesso a bases de conhecimento via RAG pode reduzir a obscuridade do processo e adicionar uma capacidade de armazenamento de informação mais dinâmica para esses sistemas. No entanto, é preciso destacar que isto não a elimina o problema totalmente, de forma que ainda não se conhece a origem e motivos da escolha entre um ou mais textos como parte da resposta pelo modelo. Além disso, as bases de conhecimento são pouco estruturadas e compostas por textos segmentados em parágrafos ou frases e com pouca ou nenhuma relação entre

os documentos ou partes do mesmo, muito diferente da forma complexa que seres inteligentes armazenam conhecimento.

Atualmente, RAG enfrenta uma série de limitações técnicas e operacionais que reduzem seu desempenho ideal em diversos cenários. Em primeiro lugar, sua dependência de um banco de dados externo impõe desafios relacionados à qualidade, atualidade e relevância dos dados recuperados. Se o banco de dados não for atualizado com frequência ou for de baixa qualidade, o sistema pode recuperar informações desatualizadas ou irrelevantes, resultando em respostas incorretas ou incompletas. Além disso, o processo de recuperação de documentos relevantes pode se tornar ineficiente em cenários com grandes volumes de dados não estruturados ou mal organizados, o que leva a um aumento no tempo de recuperação ou, em alguns casos, a falhas em encontrar informações úteis (BARNETT et al., 2024).

Outra limitação significativa do RAG é a integração entre o módulo de recuperação e o de geração. Embora a recuperação traga informações contextuais, o modelo de geração pode não ser capaz de integrar essas informações de maneira eficaz, levando a erros de coerência ou lógica. Isso ocorre porque o modelo de geração, ao combinar informações recuperadas com sua base de conhecimento inerente, pode confundir ou mesclar inadequadamente diferentes fontes, resultando em inconsistências no conteúdo final. Além disso, a capacidade de "ler" e processar os documentos recuperados de forma eficiente é limitada, especialmente quando múltiplos documentos são retornados, o que pode causar problemas no entendimento de longo contexto (LEWIS et al., 2020).

O RAG também apresenta desafios em cenários onde as perguntas são amplas, abertas ou ambíguas. Em questões de interpretação subjetiva ou onde múltiplos caminhos de recuperação são possíveis, o sistema pode não ser capaz de identificar de forma clara quais informações são mais relevantes, levando à recuperação de documentos irrelevantes ou apenas parcialmente corretos. O balanceamento entre exploração e especificidade na busca também é um desafio, já que, em alguns casos, o modelo pode recuperar documentos muito amplos ou excessivamente focados, sem a flexibilidade necessária para lidar com nuances (NOGUEIRA; CHO, 2019)..

Essas limitações sugerem que, enquanto o RAG oferece um avanço promissor na geração de respostas mais informadas, ele ainda carece de melhorias em termos

de integração eficiente entre recuperação e geração, otimização do processo de recuperação e melhor manejo de contextos ambíguos ou altamente dinâmicos.

Essa abordagem pode ser suficiente quando cada segmento de informação é relevante ou suficiente, porém, quando se faz necessário relacionar informações e a resposta depende de múltiplas segmentos, cenário mais real e mais condizentes com a capacidades de busca do ser humano, essa abordagem começa a apresentar suas limitações e a necessidade de uma melhor representação do conhecimento e de como buscá-la (BARNETT et al., 2024; LEWIS et al., 2020).

Por isso, Hawkins (2021) questiona sobre o quanto a área de IA realmente evoluiu em compreender e simular o comportamento da inteligência humana. Para que ocorra um avanço desses agentes inteligentes a fim de que possam ser empregados em solução de problemas reais de forma confiável, faz-se importante compreender melhor o funcionamento do aprendizado humano e computacional e tentar responder perguntas como: porque ainda existe uma diferença grande entre o aprendizado humano e o artificial? Que tipo de mecanismos pode ser implementados para reduzir essa diferença ou até superar em algum aspecto? O que pode ser feito para fornecer dados mais representativos para os modelos? Como melhorar a explicabilidade e interpretabilidade dos modelos através de representações e arquiteturas mais transparentes? Como representar melhor o conhecimento? Como aperfeiçoar a busca da informação armazenada nessas bases de conhecimento?

Estes estudos sugerem a necessidade da utilização de métodos que melhor expressão a complexidade do conhecimento. A representação do conhecimento por si só é um campo de estudos dentro da IA e tem como objetivo o desenvolvimento de representações de informações e conhecimentos sobre o mundo de forma que o computador consiga processá-los e utilizá-los. A representação do conhecimento incorpora conhecimentos de psicologia, neurociência, matemática e computação para modelar diversos comportamentos humanos como, por exemplo, o senso comum, aplicação de regras de conjuntos e subconjuntos, abstração de formalismos, descoberta de novos conhecimentos por indução e dedução e vários outros fenômenos gerados pela inteligência humana (RANDALL; SHROBE; SZOLOVITS, 1993).

Uma das primeiras formas de representação do conhecimento ocorreu por meio da utilização das redes semânticas (RS), nas quais é utilizada uma notação em grafo para representar os conhecimentos em padrões de nós interconectados

(HITZLER; KRÖTZSCH; RUDOLPH, 2009). Esta forma de representação foi seguida das ontologias, que são modelos de dados ou uma representação formal deles e que representa os conceitos e relações destes dentro de um domínio (BUITELAAR; CIMIANO; MAGNINI, 2005). Acompanhando estes, foi inserido o conceito de Web Semântica e sugerido que os padrões técnicos como *Uniform Resource Identifier* (URI), *Resource Description Framework* (RDF) e *Web Ontology Language* (WOL) deveriam ser promovidos e desenvolvidos. Isso contribuiu para o crescimento e avanço de pesquisas focadas em representação de conhecimento baseados em grafos (BERNERS-LEE; HENDLER; LASSILA, 2001). Em 2006, foi apresentado o conceito de Linked Data, que propôs ligar diferentes conjuntos de dados entre si na Web Semântica para serem tratados como um grande gráfico de conhecimento global. E finalmente, em 2012, o Google propôs uma nova tecnologia, chamada *Knowledge Graph* (KG, ou CG do português “grafo de conhecimento”) com o objetivo de utilizar o conhecimento semântico em pesquisas na web para identificar e reduzir a ambiguidade entre entidades no texto, enriquecer os resultados da pesquisa com links para entidades relacionadas na pesquisa exploratória e, dessa forma, melhorar a capacidade do motor de pesquisa e a experiência do usuário (CHEIN; MUGNIER, 2009). Esse marco levou ao desenvolvimento de vários CGs próprios por diversas organizações como, por exemplo, o Satori (KNOWLEDGE MINING API, 2016) que posteriormente foi incorporado ao BING (buscador da *Microsoft*), iniciativas OpenSource como DBpedia (BIZER et al., 2009), o YAGO (SUCHANEK; KASNECI; WEIKUM, 2008), Wikidata ou Freebase (BOLLACKER; COOK; TUFTS, 2007) e KBpedia (KBPEDIA, 2021).

O KG ou grafo de conhecimento (GC), em português, é uma estrutura que representa entidades do mundo real como, por exemplo, pessoas, lugares e coisas e a relação existente entre elas. Diferente dos bancos de dados vetoriais, vastamente utilizados em RAG, como bases de conhecimento (LEWIS et al., 2020), os GCs fornecem representações simbólicas estruturadas, codificando explicitamente conceitos como nós interligados dentro de uma rede. Uma semântica mais rica da informação é capturada pela modelagem de diversos relacionamentos entre entidades, incluindo conexões taxonômicas, lógicas, temporais e procedimentais. Os dois principais componentes do grafo são os vértices, ou nós, que as arestas que ligam os nós. As arestas são o que permitem representar a conexões do mundo real, enquanto os nós possuem etiquetas (labels) que trazem semântica pra cada a

informação. Além dessa capacidade de representar o conhecimento de forma mais próxima do mundo real, o grafo não impõe uma estrutura para os dados, permitindo que, com a evolução da base de conhecimento, possa ser inserido novos tipos de dados seja inserido, novas labels e novos tipos de relações (KEJRIWAL; KNOBLOCK; SZEKELY, 2021).

Um exemplo atual e cotidiano da aplicação de GC é no dispositivo Alexa. Segundo a *Amazon*, o uso de GC na Alexa tornou-a mais inteligente e gerou maior engajamento pelo aspecto mais natural e útil para o usuário (HAASE et al., 2017; AMAZON, 2018). Outro exemplo que pode ser citado que utiliza grafos de conhecimento é o assistente virtual e motor de busca do Google (SINGHAL, 2012). Além destes, uma aplicação importante atualmente ocorre em redes sociais como, por exemplo, o LinkedIn e Instagram. Nesses dois casos, a aplicação dessa estratégia permite relacionar pessoas e empresas de modo a aumentar as chances de estas possuírem interesses correspondentes (HE, 2016). O exemplo de GC como uma estrutura de armazenamento mais conhecida é o Wikipedia, que modela rica informação factuais de forma estruturada.

Outra característica importante dos GCs, é a possibilidade de combinação entre várias fontes de informações para a descoberta de novos conhecimentos, podendo ser compreendido como uma forma de raciocínio e se caracterizar como um conhecimento coletivo. Neste, o aparecimento de uma relação entre duas entidades presente em várias fontes reforça este entendimento, assim como ocorre com o conhecimento humano, no qual a compreensão e memórias que aparecem mais frequentemente formam o conhecimento coletivo, sendo esse um dos pilares da inteligência humana (SURAN; PATANAIK; DRAHEIM, 2020). Em contraposição ao ser humano, que possui alta capacidade em aprender continuamente, as redes neurais e conseqüentemente os modelos de LLMs, não possuem essa capacidade. A aprendizagem contínua é a capacidade de aprender novos conceitos e tarefas sequencialmente, sem perdas do conhecimento previamente adquirido. Nesse contexto, a utilização do GC como base de conhecimento do sistema de IA pode aumentar a capacidade apresentada e mitigar essa limitação (DONGMIN, 2019).

Marcus (2020), em seu estudo sobre o futuro da IA diz que, ao contrário dos modelos generalistas treinados com imensas quantidades de dados, uma abordagem híbrida, orientada para o conhecimento, baseada no raciocínio e centrada em modelos cognitivos, pode fornecer o substrato para uma IA mais rica e robusta do que é

atualmente possível. Dessa forma, a incorporação de técnicas simbólicas pode promover melhor controle, maior robustez e capacidade em raciocínio e explicação de seus resultados.

Com o surgimento de modelos de LLMs mais poderosos, como GPT-4 e Gemini, e a alta capacidade de compreensão e geração de linguagem baseado e um conhecimento de senso comum generalizado de um corpus extremamente geral e abrangente (OPENAI, 2023), as pesquisas com GC tem focado mais nas novas capacidades de armazenamento da informação, de forma mais rica, de domínios específicos e a possibilidade de buscas mais especializadas, considerando a base de conhecimento como um conhecimento interligado e explícito das entidades do mundo real, trazendo um novo nível de abstração do conhecimento para o contexto dos modelos de LLMs, visto a importância do mesmo para a resposta gerada. Contudo, tanto o processo de construção do grafo, quanto a formulação de consultas nessa estrutura requer experiência e apresentam um custo. Além disso, uma característica importante dos grafos de conhecimento é a sua incompletude e dispersão (dados esparsos), não possuindo a capacidade de raciocínio, adaptação ou generalização semanticamente que os modelos neurais, como os de LLM, exibem.

Dessa forma, percebe-se uma que LLMs e GCs possuem vantagens e desvantagens complementares, o que motiva a unificação dessas duas tecnologias. Pan et al. (2024), propõe um *roadmap* para unificação dessas duas tecnologias de forma complementar, consistindo em três frameworks: 1) LLMs aprimorados por GCs e que os incorporam durante as fases de pré-treinamento e inferência dos LLMs (ou para o propósito de melhorar a compreensão do conhecimento aprendido pelos LLMs); 2) GCs aprimorados por LLMs, que utilizam LLMs para diferentes tarefas de GC, como embedding, construção, geração de texto a partir de gráficos e respostas à perguntas; e 3) LLMs + GCs sinérgicos, nos quais os LLMs e os GCs desempenham papéis iguais e trabalham de forma mutuamente benéfica para aprimorar tanto os LLMs quanto os GCs para um raciocínio bidirecional impulsionado tanto por dados quanto por conhecimento.

Vários trabalhos vêm mostrando possibilidades da junção de LLM e GC dentro dessas três categorias, como Wang et al. (2024), que apresenta o DocGrapgLM, que altera a estrutura do modelo LLM com GC, otimizando o resultado em tarefas de extração de informação (EI) e QA. De forma similar, Chai et al. (2023) propõem o GraphLLM, uma abordagem que integra sinergicamente modelos de aprendizagem

de grafos com LLMs, permitindo às LLMs interpretar e raciocinar com em cima dos dados estruturados em um GC. Já Sequada et al. (2023), demonstra o impacto gerado pela modelagem dos dados de um banco SQL em um GC, alcançando 54% de acurácia na tarefa de QA ao utilizar GC, comparado a 16% quando utilizando os dados em banco SQL, enquanto Fatemi, Halcrow e Perozzi (2023), apresentam um estudo de como a codificação de dados estruturados em grafos como texto para consumo por LLMs afeta geração das respostas. Além desses estudos, Chai et al. (2023) ainda apresenta um framework baseado em três fases (*Extract-Define-Canonicalize*) de construção de grafo de conhecimento.

Além dos diversos trabalhos científicos, empresas provedoras de banco de grafos, como o Neo4j, Memgraph e Nebula Graph, tem impulsionado esse movimento de utilização de grafos em agentes inteligentes, como a inserção de buscas híbridas e índices vetoriais nos bancos de grafos (BRATANIC, 2024). ou implementando operação e busca em grafo em memória, de forma a possibilitar uma velocidade superior aos concorrentes (JAVOR, 2024), gerando extensões e ferramentas que simplifiquem e facilitem a criação e busca em grafo (NEBULAGRAPH, 2023). Porém, poucas pesquisas científicas podem ser encontradas mostrando a viabilidade da aplicação de GC em um contexto mais real ou apresentando as reais dificuldades entre o uso de GC ou bases vetoriais em sistemas conversacionais.

Dessa forma, a utilização de GC como representação do conhecimento pode ser uma alternativa e evolução natural do RAG baseado em banco de dados vetoriais, incorporando uma forma mais rica de representar o conhecimento, além de possibilitar novas abordagens de armazenamento e contextualização de sistemas conversacionais. Aplicações conversacionais que necessitam de constantes agregação e alteração de conhecimento podem se beneficiar de um sistema de memória e conhecimento mais robusto, além de proporciona uma melhor explicabilidade e interpretabilidade dos resultados. Ao explorar a sinergia entre GC e LLMs dentro de um ambiente empresarial, espera-se que seja possível compreender a vantagens e dificuldades de se construir uma um GC utilizando LLM, além das vantagens de se adotar GC como a representação do conhecimento, indicando um caminho para reduzir a lacuna entre a forma como ocorre o armazenamento e a busca do conhecimento humano e das máquinas.

2 OBJETIVOS

2.1 Objetivo geral

Propor e avaliar um sistema de memória baseado em grafo de conhecimento para agentes conversacionais.

2.2 Objetivos específicos

- Desenvolver um *pipeline* de construção de um grafo de conhecimento com foco na representação de diferentes níveis de abstrações do conhecimento.
- Comparar a resposta do agente proposto utilizando *Retrieval-Augmented Generation* (RAG) e *GraphRAG*.
- Avaliar a capacidade do grafo de conhecimento em simular as três características da memória humana: codificação, armazenamento e recuperação.
- Avaliar a capacidade do agente proposto em memorizar experiências.
- Avaliar a capacidade do grafo de conhecimento em integrar informação estrutura e não estruturada.
- Apresentar um base para compreensão das vantagens e limitações dessa tecnologia em ambientes corporativos.

Organização do Documento

O presente trabalho foi organizado da seguinte forma:

- O capítulo três abordará o campo de estudo de aprendizado de máquina (AM) e aprendizado profundo (AP) no contexto de Processamento de Linguagem Natural (PLN) e o caminho até os modelos de linguagem massivos e como isso impulsionou seu crescimento, bem como aprofundar na tarefa de pergunta e resposta (PR), como *pipeline* de *Retrieval-Augmented Generation* (RAG) auxilia na redução de suas limitações e eleva essa tarefa para criar os agentes conversacionais.
- O quarto capítulo abordará os conceitos de grafo de conhecimento e como é uma técnica complementar aos modelos de linguagem em grande escala (LLM), assim como técnicas de construção e avaliação dos mesmos. Além disso, o capítulo finaliza com uma seção sobre a utilização de grafo de conhecimento no contexto de *pipeline* de RAG.
- O quinto capítulo apresenta a proposta da pesquisa e a metodologia utilizada neste trabalho. Além desses, também aborda como o GC representa a memória de longo prazo de um agente, a codificação e armazenamento da informação em GC, incluindo uma seção sobre o uso da representação do usuário para personalizar a conversa e técnicas de recuperação de informação no GC.
- O capítulo seis apresenta os resultados dos experimentos e a discussão dos mesmos.
- O capítulo de considerações finais aborda sugestões para estudos futuros.
- E, por fim, o capítulo nove conclui o trabalho, apresentando como o estudo responde aos objetivos propostos.

3 A LINGUAGEM NATURAL COMO UMA INTERFACE PARA O CONHECIMENTO

A linguagem natural se refere à maneira como os humanos se comunicam uns com os outros, podendo ser expresso por escrita, fala e sinais. Diariamente, o ser humano lida com textos em várias atividades como, por exemplo, *e-mails*, jornais, menus, páginas *web*, letreiros e outros diversos. Por isso, faz-se necessárias formas de processar computacionalmente essa linguagem natural para permitir que programas sejam capazes de compreender e processar essa grande quantidade de informação presente no dia a dia dos seres humanos (JURAFSKY; MARTIN, 2008).

Antes do uso de computação para o estudo científico da linguagem, apenas o campo de estudo chamado de linguística, que incluía o estudo da gramática, semântica e fonética, era predominante. Muitos dos estudos se resumiam a formalismos matemáticos para descrever uma linguagem a partir da utilização de linguagens formais e teorias dos autómatos. Com a evolução da ciência da computação, algumas técnicas estatísticas surgiram e, na década de 1990, os métodos estatísticos e o AM estatístico começaram a substituir as abordagens clássicas de linguagem baseadas em regras, principalmente pelos melhores resultados, velocidade e robustez (MITKOV, 2005). A partir do desenvolvimento de métodos de processamento e o progresso de técnicas de IA em simulação do entendimento de linguagem aliados ao AP, tornou-se possível a construção de modelos que recebem textos criados por seres humanos como entrada e que possuem como saída textos que aparentam ser naturais, ou seja, também criados por seres humanos. Dessa forma, o ramo da ciência da computação conhecido como PLN foi consolidado e permitiu a evolução de tarefas que até o momento eram consideradas impossíveis (GOLDBERG, 2017).

De modo geral, o PLN tem como objetivo proporcionar aos computadores a capacidade de compreensão de textos da mesma maneira que os seres humanos executam. Por isso, o PLN está inserido em uma subárea de IA, visto que se tenta compreender, gerar e simular computacionalmente a inteligência dos seres humanos sobre textos escritos e/ou sonoros. Além disso, essa é uma área que está amplamente preocupada com a construção de ferramentas computacionais capazes em gerar utilidades por meio da linguagem como, por exemplo, tradução automática, resumo, respostas às perguntas e identificação de sentimentos. Ao não se limitarem somente

ao estudo da linguagem, essas ferramentas podem ser aplicadas em uma variedade de diferentes disciplinas científicas (JOHNSON, 2009).

3.1 Fundamentos de aprendizado de máquina para processamento de linguagem natural

O aprendizado de máquina (AM) é um campo multidisciplinar que combina diversas áreas de conhecimento, incluindo estatística, ciência da computação, inteligência artificial (IA) e otimização, com o objetivo de desenvolver algoritmos capazes de aprender a partir de dados. Ele se distingue da IA, que é um campo mais amplo que engloba várias técnicas e abordagens, sendo o AM apenas um subconjunto da IA (MITCHELL, 1997). O AM busca automatizar a capacidade de sistemas computacionais de realizar tarefas complexas, como classificação, regressão, reconhecimento de padrões e predição, sem ser explicitamente programado para isso. Esse campo é amplamente utilizado em diversas áreas, como medicina, processamento de linguagem natural (PLN), reconhecimento de fala, visão computacional, entre outras.

Os algoritmos de aprendizado de máquina podem ser divididos em três principais abordagens: o aprendizado supervisionado (AS), o não supervisionado (ANS) e o aprendizado por reforço (AR). No aprendizado supervisionado, o sistema é treinado com dados rotulados, ou seja, entradas com as saídas correspondentes, com o objetivo de aprender a mapear essas entradas para as saídas. As principais tarefas desse tipo de aprendizado são a classificação, que prevê a classe ou rótulo de novas entradas, e a regressão, que foca na previsão de valores contínuos, como preços de casas ou temperaturas. Já o aprendizado não supervisionado trabalha com dados sem rótulos, buscando identificar padrões ocultos. Suas principais tarefas são a clusterização, que agrupa dados com base em semelhanças, e a redução de dimensionalidade, que simplifica a análise reduzindo o número de variáveis. Por fim, o aprendizado por reforço treina um agente a interagir com um ambiente, aprendendo por tentativas e erros, recebendo recompensas ou punições conforme suas ações, sendo amplamente usado em áreas como controle de robôs, jogos e sistemas de recomendação (RUSSEL; NORVIG, 2003).

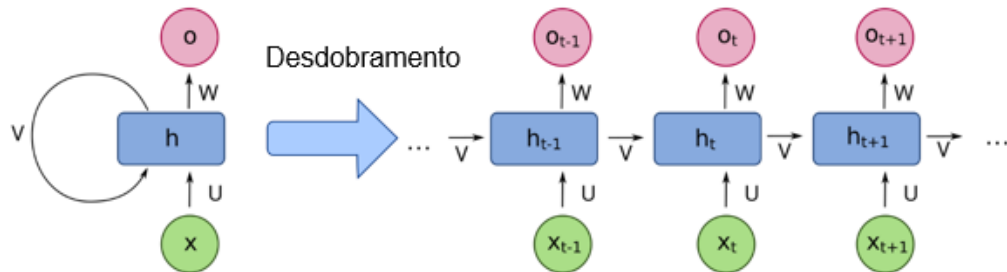
Como consequência do crescimento dos modelos de aprendizado, da quantidade de dados e do poder computacional, o aprendizado profundo (AP) surgiu

como uma extensão do AM, permitindo abordar problemas mais complexos e extrair padrões de dados com estruturas mais profundas. O AP utiliza redes neurais artificiais (RNA), que são inspiradas no funcionamento do cérebro humano, para modelar dados complexos e fazer previsões. O processo de aprendizado em redes neurais ocorre por meio de um algoritmo conhecido como *backpropagation* (retropropagação), que ajusta os pesos das conexões entre os neurônios para minimizar o erro nas previsões. O treinamento envolve duas fases principais: a propagação da entrada pelas camadas da rede (*feedforward*) e a retropropagação do erro para ajustar os pesos (GOODFELLOW; BENGIO; COURVILLE, 2016). Durante o treinamento, a função de erro é minimizada através de um processo iterativo, utilizando o gradiente da função de erro para ajustar os pesos da rede. Esse processo é repetido até que o erro seja minimizado ou até que um número predefinido de iterações seja alcançado (LINNAINMAA, 1976).

Uma das arquiteturas mais notáveis no AP são as redes neurais convolucionais (RC), que se destacaram inicialmente em tarefas de visão computacional. As RCs utilizam camadas de convolução, que atuam como filtros treináveis que extraem características específicas das entradas, como bordas ou texturas em imagens. Essa arquitetura foi popularizada pela AlexNet, uma RC que revolucionou a área de reconhecimento de imagens ao vencer a competição ImageNet em 2011 (KRIZHEVSKY; SUTSKEVER; HINTON, 2012). A capacidade das RCs de processar dados visuais com alta precisão abriu novas fronteiras para o aprendizado profundo em outras áreas.

Apesar das RCs serem altamente eficazes em dados espaciais, como imagens, surgiram desafios para o processamento de dados sequenciais, como texto e áudio. Para resolver esses problemas, foram desenvolvidas redes recorrentes (RR), que possuem a capacidade de "memorizar" informações passadas, utilizando o feedback interno de suas camadas (RUMELHART; HINTON; WILLIAMS, 1986). Nas RR, a saída de um neurônio em uma camada pode ser reutilizada como entrada para ele mesmo, o que permite modelar dependências temporais entre os dados, como pode ser observado na figura 1.

Figura 1. Representação de um neurônio recorrente.



Legenda: Onde "x" é a entrada, "o" é a saída e "h" é o estado interno do neurônio recorrente.

Fonte: adaptado de Goodfellow, Bengio e Courville (2016).

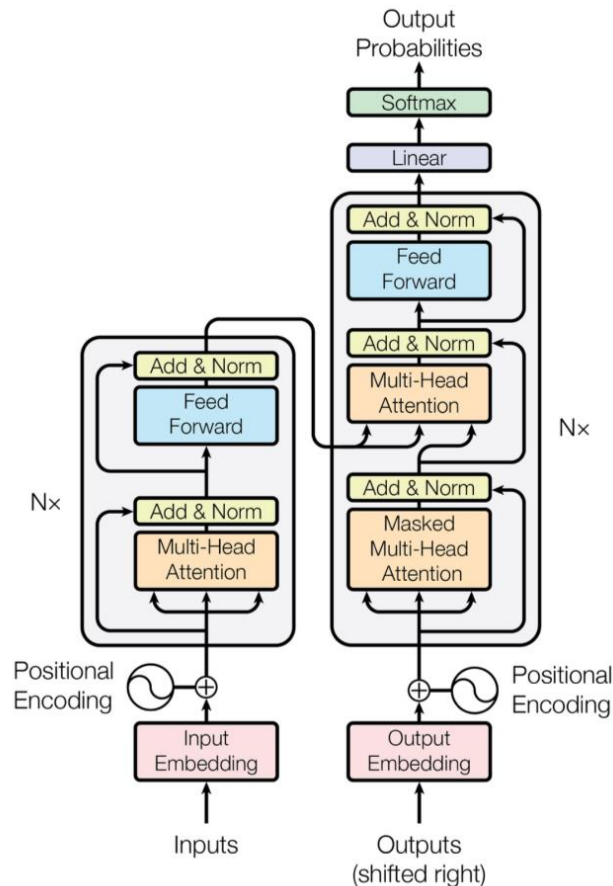
Contudo, as RR tradicionais enfrentavam dificuldades em aprender padrões de longo prazo, devido ao fenômeno do gradiente de desaparecimento. Para contornar esse problema, foi introduzido o conceito de *Long Short-Term Memory* (LSTM), uma arquitetura de RR capaz de armazenar informações ao longo de sequências mais longas, utilizando "portas" que controlam quais informações são mantidas ou descartadas na memória (HOCHREITER; SCHMIDHUBER, 1997). Esse avanço foi fundamental para o sucesso das RR em tarefas de PLN, como tradução automática, reconhecimento de voz e geração de texto.

Embora as RR e LSTM tenham sido amplamente utilizadas para PLN, surgiram novas abordagens que superaram suas limitações. Uma das mais revolucionárias foi o mecanismo de atenção, que permite que o modelo se concentre em partes importantes de uma sequência de dados, ignorando outras. Essa técnica foi fundamental para o desenvolvimento dos Transformers, uma arquitetura introduzida no artigo "*Attention Is All You Need*" (VASWANI et al., 2017). Diferente das RR, os Transformers processam a sequência de entrada de forma paralela, sem a necessidade de processar dados em ordem temporal, o que aumenta significativamente sua eficiência.

Os Transformers utilizam uma arquitetura baseada em codificadores e decodificadores, onde o mecanismo de atenção é aplicado tanto nas entradas quanto nas saídas, permitindo que o modelo considere o contexto completo de uma frase ou sequência. Esse avanço resultou em melhorias substanciais em várias tarefas de PLN, como tradução automática, sumarização de textos e respostas a perguntas.

Modelos baseados em *Transformers*, como o BERT e o GPT, tornaram-se a base para o estado da arte em PLN (VASWANI et al., 2017).

Figura 2. Representação da arquitetura *Transformer*.



Legenda: Na direita tem-se o encoder e na esquerda o decoder. "Nx" são módulos que podem ser empilhados para gerar modelos mais profundos e elas são constituídas basicamente de camadas de *Multi-Head Attention* e *Feed Forward*. Além disso foi adicionado uma codificação posicional para informar ao modelo a posição relativa de cada palavras, uma vez que não se tem mais recorrências. Fonte: retirado *ipsis litteris* de Vaswani et al. (2017).

Os avanços em redes neurais, particularmente os Transformers, permitiram que modelos de PLN alcançassem um desempenho sem precedentes em tarefas como tradução, reconhecimento de entidades, resumo de textos e geração de linguagem natural, se tornando, assim, formando a base para muitos dos avanços modernos em PLN.

3.2 Modelando a linguagem natural

A ascensão do AP transformou significativamente o campo do PLN nos últimos anos ao possibilitar a construção de modelos que apresentam resultados notáveis em diversas tarefas, destacando-se as RNs que executam um processo ponta a ponta. Apesar da redução da necessidade da extração manual de características, como era realizado antes do AP, as RNAs ainda necessitam de uma representação numérica para serem treinadas. Inicialmente, modelos estatísticos baseados em frequência de palavras foram utilizados para representar textos, como *bag-of-words*, porém a limitação relacionada aleatoriedade da ordem das palavras ainda permanecia (HARRIS, 1954).

Considerando-se isso, foi criado o TFIDF (*term frequency–inverse document frequency*) com o objetivo de incorporar a relevância de uma palavra para um documento em uma coleção de textos (*corpus*) (BENGIO et al, 2003). Contudo, estes métodos ainda apresentam limitações por representar o documento como um todo ou a palavra em relação a todo o *corpus*, o que resulta em representações esparsas que podem variar conforme o tamanho deste e do vocabulário, comprometendo a qualidade semântica e a aplicabilidade das representações. Para superar essas limitações, foi apresentado um método de representação de texto mais eficiente e de baixa dimensionalidade, chamado *embedding*.

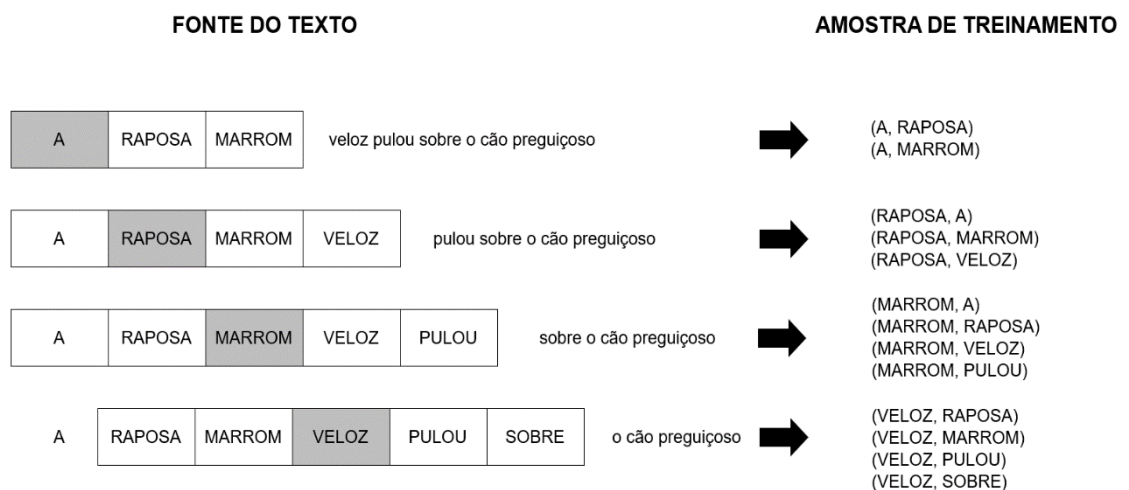
O *Word embedding* (WE) é uma forma de representação da semântica de uma palavra através de um vetor e é baseada na hipótese de distribuição, em que se considera que palavras que aparecem em contextos semelhantes possuem significados semelhantes. Para isso, WEs são pré-treinados em uma tarefa em que o objetivo é prever uma palavra com base em seu contexto utilizando uma RN (MIKOLOV et al., 2013a). Uma das vantagens do WE é sua capacidade em ser utilizado para a composição semântica como, por exemplo, os *embeddings* de “homem” + “real” = “rei”. Logo, as palavras são mapeadas para um espaço onde suas distâncias equivalem às suas distâncias semânticas. Esses vetores são comumente representados por *arrays* de 300 a 1000 posições compostos por números reais.

Apenas em 2013, com a publicação do *word2vec* é que o WE foi amplamente difundido e estabelecido como o padrão de representação de palavras. Mikolov et al. (2013a) propuseram os modelos *Continuous Bag Of Words* (CBOW) e *skip-gram* como abordagens neurais para construir *embeddings* de palavras em que o objetivo

é calcular a probabilidade condicional de uma palavra-alvo dada as palavras de contexto em um determinado tamanho de janela e para construir *embeddings* de palavras em que o objetivo é prever as palavras do contexto circundante dada uma palavra-alvo central (MIKOLOV et al., 2013a,b).

É importante destacar que essas abordagens apresentaram importantes limitações na representação de frases, pois a combinação de *embeddings* (combinação que pode ser feita por média, por exemplo) não é tão precisa para representar o contexto da frase e os resultados são inferiores quando se aumenta a extensão das frases. Outra limitação do método é que janelas de contexto (conforme ilustrado pela figura 10) curtas geram *embeddings* similares para palavras diferentes, de forma que nem sempre é possível utilizar uma janela longa (por exemplo, em textos onde as frases são curtas). Além disso, cada palavra possui apenas uma representação, ou seja, não considera a polissemia (capacidade de uma mesma palavra ter diferentes significados dependendo do contexto), o que dificulta a aplicação de um mesmo modelo em diferentes contextos e tarefas (MIKOLOV et al., 2013a).

Figura 3. Exemplo de utilização de janela de contexto para geração de *dataset* para treinamento de *word embedding*.



Legenda: A célula cinza é a que será predita e as células brancas são o contexto a ser utilizado (no caso do *Continuous Bag Of Words* e ao contrário no caso do *skip-gram*).

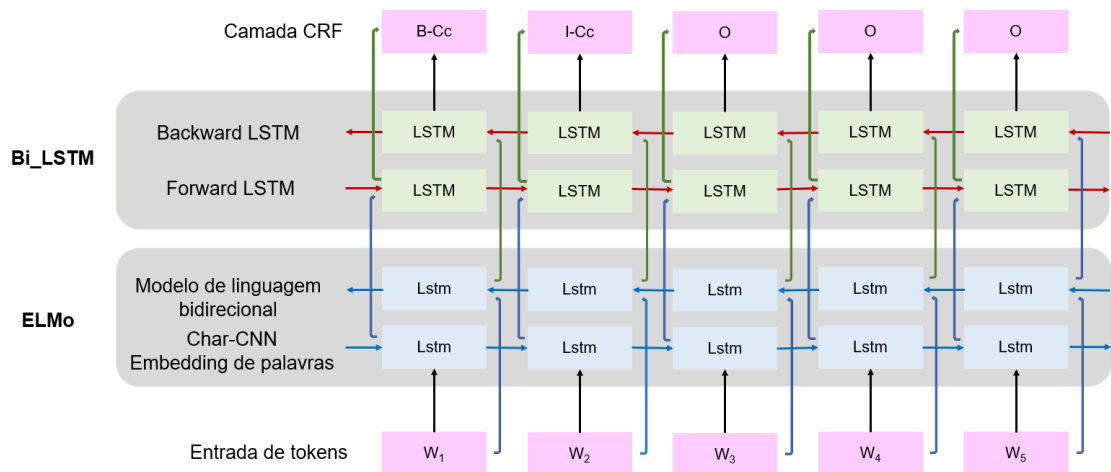
Fonte: adaptado de Mikolov et al. (2013a,b).

A fim de solucionar esses obstáculos, foi proposto o *Embedding from Language Model* (ELMo), no qual a representação de palavra apresenta características mais

complexas da linguagem como, por exemplo, sintaxe, semântica e variação entre contextos linguísticos, ou seja, há modelagem da polissemia. Uma característica do ELMo é o uso de RRs bidirecionais (LSTM ou *Gated Recurrent Unit* - GRU), conforme figura 4, nos quais os *embeddings* são funções aprendidas dos estados internos da rede. Dessa forma todas as sub-representações presentes nas camadas internas da rede são utilizadas para gerar a representação final, resultando em uma representação mais rica e que considera todo o contexto apresentado. Outra característica importante do ELMo é que suas representações são puramente baseadas em caracteres, permitindo que a rede use pistas morfológicas para formar representações robustas para *tokens* fora do vocabulário não vistos no treinamento.

Estes modelos, treinados em um grande corpus, se tornam modelos de representação razoável da linguagem no qual ela foi treinada. Assim, eles podem ser utilizados como entrada para outro modelo responsável por uma tarefa específica como, por exemplo, tradução e classificação. Outra forma de utilizá-los é como modelo pré-treinado, adicionando ao final um classificador ou regressor, dependendo da tarefa, e treinando a rede como um todo em uma nova tarefa (PETERS et al., 2018).

Figura 4. Representação da arquitetura bidirecional utilizada no ELMo.



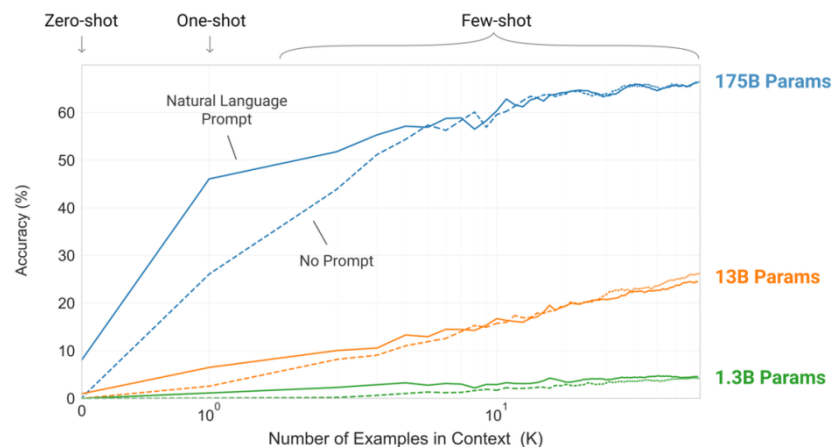
Fonte: adaptado de Peters et al. (2018).

Esse conceito representou o início dos modelos de linguagem (ML), conceito que a partir deste ponto foi se tornando cada vez mais comum. Dessa forma, pode-se definir ML como sendo um modelo da distribuição de probabilidade sobre palavras ou sequências de palavras. Na prática, um ML fornece a probabilidade de uma determinada sequência de palavras ser válida semanticamente naquele contexto. A

consequência desses modelos é a capacidade de modelar a compreensão de uma linguagem, possibilitando uma representação mais próxima da humana ao considerar o contexto (domínio do conhecimento, tempo, formalidade e outros) (VASWANI et al., 2017).

Devido aos problemas de paralelização no treinamento das RRs, torna-se difícil o treinamento de sequências muito grandes. De forma a contornar as limitações das RRs, o *Google* criou a arquitetura *Transformers*, que, como visto no capítulo anterior, retira as recorrências e se baseia apenas em mecanismos de atenção (JOHNSON, 2009). Essa nova arquitetura possibilitou um treinamento muito mais rápido e eficiente, permitindo a construção de modelos muito maiores e treinados em imensas quantidades de texto comparados aos modelos anteriores. Isto resultou em modelos com qualidades superiores, demonstrando relação entre os tamanhos dos modelos, quantidade de dados de treinamento e resultados positivos, como pode ser visto na figura 5.

Figura 5. Acurácia de diferentes modelos baseados em *Transformers* na tarefa de pergunta e resposta conforme a quantidade de dados utilizados no treinamento.



Legenda: GPT-1 (verde), GPT-2 (laranja) e GPT-3 (azul).
 Fonte: retirado *ipsis literis* de Brown et al. (2020).

A forma mais comum de utilização dessas arquiteturas para o PLN é através da realização de um pré-treino no modelo com uma grande quantidade de dados, normalmente um corpus de domínio geral, que serve como base para qualquer tarefa. Caso esse modelo necessite de ser aplicado à um domínio específico, é então

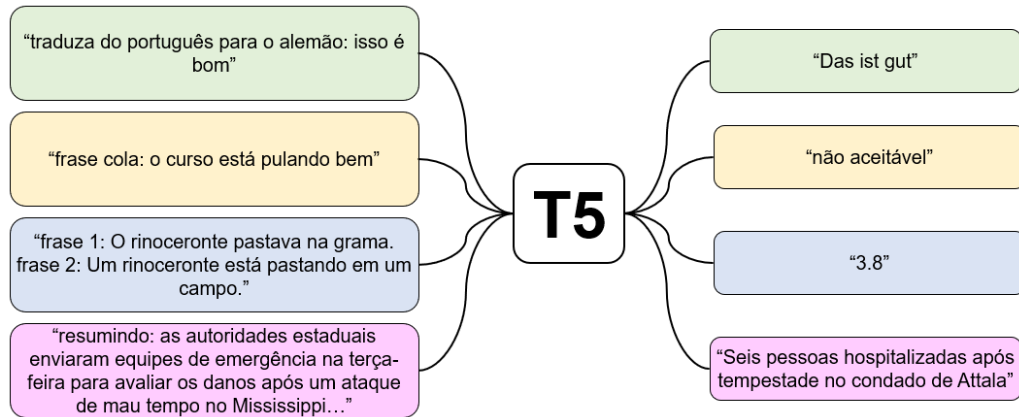
realizado um *fine-tuning* em um corpus do domínio. Assim, tem-se um modelo de linguagem pronto para ser treinado em qualquer tarefa de PLN.

Diferentes modelos implementaram estratégias variadas para treinar essa arquitetura como, por exemplo, o BERT do *Google* que utiliza uma técnica para mascarar palavras aleatórias do texto para que a rede possa prever qual palavra foi mascarada (DEVLIN et al., 2018). Outro exemplo de estratégia foi implementado no GPT da *OpenAI* que utiliza o método mais clássico de ML: previsão da próxima palavra dada uma sentença (RADFORD, et al., 2019).

Em seguida, modelos como o RoBERTa (RADFORD, et al., 2019), apresentaram alterações na arquitetura original do BERT (DEVLIN et al., 2018) para melhorar o desempenho, como aumento na velocidade de treinamento e assim, definiram um novo estado da arte em tarefas de comparação semântica entre pares de frases. No entanto, esses modelos necessitam que ambas as sentenças sejam alimentadas na rede, o que causa uma sobrecarga computacional. Por exemplo, para obter o par mais semelhante em uma coleção de 10.000 sentenças, requerem-se cerca de 50 milhões de cálculos de inferência com BERT, o que, em uma máquina comum para treinamento (com GPU), pode levar cerca de 100 horas. Por isso, o surgimento do Sentence-BERT, ou como ele é chamado SBERT, permitiu um importante avanço na tarefa de busca semântica. O SBERT é uma modificação da rede pré-treinada de BERT que utiliza estruturas de rede siamesas e triplas para derivar *embeddings* de frases. Esses *embeddings* podem ser armazenados e utilizados para comparação semântica fora do modelo como, por exemplo por um algoritmo de cálculo de distância cosseno otimizado. Dessa forma, a busca semântica que poderia demorar dezenas de horas pode ser reduzida para poucos segundos (REIMERS; GUREVYCH, 2019).

Enquanto a maioria dos modelos de linguagem anteriores eram utilizados em tarefas específicas, o T5 surgiu com a proposta de um só modelo atender a diferentes tarefas ao mesmo tempo. Com o objetivo de testar os limites da transferência de conhecimento (*transfer learning*), o T5 utiliza um modelo pré-treinado e faz um treinamento em diversas tarefas. Durante a inferência é utilizado um trecho de texto no início do texto de entrada para que o modelo consiga identificar qual é a tarefa desejada e responder de acordo, apresentando uma alta capacidade de abstrair e compreender a estrutura da linguagem natural (RAFFEL et al., 2019). A figura 6 apresenta um infográfico da capacidade da T5 de realizar múltiplas tarefas de PLN.

Figura 6. Representação da capacidade do modelo T5 em ser treinado em multitarefas.



Fonte: adaptado de Raffel et al. (2019).

Em cada período na academia um paradigma de aprendizado se sobressaiu, como apresentado no quadro 1. Inicialmente, teve-se os modelos de aprendizado supervisionados não baseados em RNs, como o TFIDF aplicados a modelos mais clássicos como SVM. Com o surgimento do AP, métodos supervisionados baseados em RNs foram predominantes, como RC e RR. Já a partir de 2017, a maioria dos avanços foram focados em executar o pré-treino em grandes modelos e em seguida realizar um ajuste fino (*fine-tuning*) do modelo em tarefas específicas. Por fim, os esforços atualmente se concentram em estudos e discussões sobre *prompt-learning*, que ao contrário dos paradigmas anteriores em que se tenta adequar o modelo os dados, nesse tenta-se ajustar os dados ao modelo pré-treinado por meio da reformulação das entradas utilizando uma função *prompt* (LIU et al., 2021). Assim, pode-se observar que o aprendizado tem caminhado no sentido de modelos mais capazes de lidar com diversas tarefas com o mínimo de alterações possível e, assim como os humanos, serem capazes de reter conhecimento de diversas tarefas simultaneamente.

Quadro 1. Paradigmas de aprendizado em processamento de linguagem natural.

Paradigma	Engenharia	Relação da tarefa
a. Aprendizagem Totalmente Supervisionada (Rede Não-Neural)	Características (por exemplo, identidade da palavra, classe gramatical, extensão da frase)	
b. Aprendizagem Totalmente Supervisionada (Rede Neural)	Arquitetura (por exemplo, convolucional, recorrente)	
c. Pré-treino, ajuste (fine-tune)	Objetivo (por exemplo, modelagem de linguagem mascarada, previsão da próxima frase)	
d. Pré-treino, prompt, previsão	Prompt (por exemplo, prefixo)	

Legenda: LM: modelo de linguagem; CLS: classificação; TAG: marcação; GEN: geração. Fonte: adaptado de Liu et al. (2021).

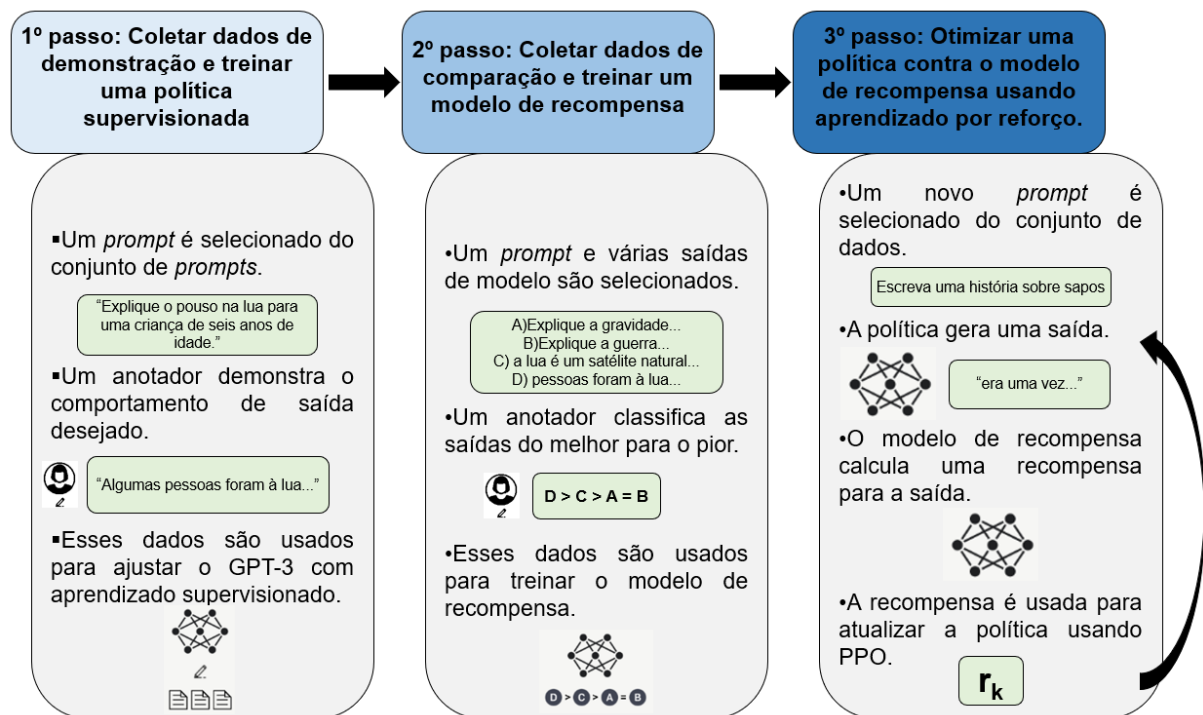
Em 2020, a OpenAi anunciou o GPT-3, um modelo LLM com capacidades inovadoras de busca, conversação e geração de texto, disponibilizada através de API. O GPT-3, assim como seu antecessor GPT-2, é um modelo que utiliza apenas o *decoder* de uma arquitetura *transformer*, porém houve um crescimento de 1.5 bilhões para 176 bilhões de parâmetros no modelo, enquanto GPT-2 foi treinado em 40 gigabytes de texto, o GPT-3 foi treinado em 570 gigabytes de textos diversos, retirados de livros, artigos, sites, blogs e outros. Isso permitiu ao GPT-3 se tornar um dos primeiros modelos a ter a habilidade de performar bem em diversas tarefas de PLN sem a necessidade de *fine-tuning* (BROWN et al., 2020; RADFORD et al., 2019). Além disso, ele foi introduzido em formato de serviço de API, iniciando um movimento de LLM como serviço e popularizando e viabilizando a tecnologia.

Apesar dos grandes avanços e novas possibilidades, o GPT-3 apresentou importantes limitações, como a dificuldade em seguir instruções, presença de alucinações e geração de conteúdo danoso e tendencioso, demonstrando

potencialidades negativas e nocivas (BROWN et al., 2020). Baseando-se nisso, houveram movimentos da OpenAI para o desenvolvimento de uma forma mais adequada para guiar o comportamento da LLM e ensinar o comportamento esperado a partir da inclusão do ser humano no treinamento. Sendo assim, o *InstructGPT* foi proposto em 2022 (OUYANG et al., 2022) e introduziu uma nova abordagem para incorporar o *feedback* humano no processo de treinamento a fim de aperfeiçoar o alinhamento da saída à intenção do usuário. Para que isso fosse possível, foi apresentado a técnica de *Reinforcement Learning from Human Feedback* (RLHF) que utiliza o *feedback* humano para a otimização da LLM (CHRISTIANO et al., 2017). Esse processo pode ser dividido em três etapas (figura 7):

- Coleta de dados e treinamento supervisionado de uma *policy*, onde *prompts* são extraídos de uma base pré-existente, um ser humano cria a saída esperada para cada um desses *prompts* e um LLM, no caso GPT-3, é treinado de forma supervisionada utilizando esses dados.
- Geração de um *dataset* comparativo para treinamento de um modelo de recompensa que seja capaz de distinguir não só entre certo e errado, mas quão melhor uma resposta é em comparação a outras respostas. O primeiro passo é a geração de saídas de diferentes modelos para um mesmo *prompt* e em seguida essas saídas são dadas a um ser humano para que ele faça o ranqueamento da melhor para a pior resposta. Esse passo é seguido da utilização desses dados no treinamento de um modelo capaz de realizar esse ranqueamento.
- Otimização do modelo de treinamento da primeira etapa utilizando o modelo de recompensa treinado na segunda etapa por meio de AR. Assim, cada *prompt* do *dataset* é fornecido como entrada para o LLM. Isso gera uma resposta que será avaliada pelo modelo de recompensa e, posteriormente, será utilizada para ajustar os pesos do LLM por meio do *Proximal Policy Optimization* (PPO).

Figura 7. Representação das etapas contidas na técnica de *Reinforcement Learning from Human Feedback*.



Legenda: PPO: *Proximal Policy Optimization*.

Fonte: adaptado de <<https://openai.com/index/instruction-following/>>. Acesso em: 20 jul.2024.

As novas capacidades desse modelo apresentaram a necessidade da criação de novas métricas e *benchmark* de avaliação. Assim, o InstructGPT foi avaliado em três aspectos principais: utilidade, veracidade e inofensibilidade. Em termos de utilidade, anotadores preferiram suas respostas em 85% ($\pm 3\%$) das vezes comparado ao GPT-3, demonstrando sua capacidade de seguir instruções do usuário. Quanto à veracidade, o modelo PPO apresentou pequenas melhorias, com resultados mais verdadeiros e informativos segundo o conjunto de dados *TruthfulQA*. Na inofensibilidade, testada com o *RealToxicityPrompts*, o modelo mostrou uma redução significativa nas respostas nocivas quando instruído a ser respeitoso, nenhuma alteração significativa sem essa instrução e um aumento na nocividade quando solicitado a fornecer respostas nocivas, sendo considerado mais prejudicial que o GPT-3 (OUYANG et al., 2022).

Esse processo gerou um modelo muito mais controlável, capaz de gerar respostas mais alinhadas ao *prompt* e respostas mais naturais. Porém, o *InstructGPT* é um modelo focado em seguir instruções e não conversas, dessa forma, o próximo

passo foi a utilização do mesmo método com um *dataset* focado em conversação. Esse aspecto conversacional do *dataset* foi gerado por seres humanos para simular os dois lados de uma conversa: tanto o lado do usuário, quanto o lado do LLM. Além disso, foi adicionado uma base de códigos fontes ao *dataset*, resultando no GPT-3.5 após treinamento, a base para o ChatGPT, uma das ferramentas mais disruptivas dos últimos anos (OUYANG et al., 2022).

Desde o seu lançamento em dezembro de 2022, o ChatGPT foi oferecido como serviço em decorrência da incorporação de conhecimentos de diversos domínios e idiomas e apresentação de capacidade conversacional inédita, possibilitando aplicações em diversas áreas e tarefas de PLN. Por isso, o ChatGPT recebeu ampla atenção e interesse, resultando em um número crescente de aplicações e pesquisas que exploram seu potencial.

Em 2023 a OpenAI anunciou o GPT-4, uma expansão do GPT-3.5. O GPT-4 é uma LLM multimodal, ou seja, aceita entradas de imagem e texto e apresenta desempenho em nível humano em vários testes profissionais e acadêmicos. Por exemplo, ele é capaz de ser aprovado em um exame simulado da ordem dos advogados com uma pontuação entre os 10% melhores. Em contraposição, a pontuação do GPT-3.5 estava entre os 10% piores. Além disso, o GPT-4 apresenta 82% menos probabilidade de responder a solicitações de conteúdo não permitido e 40% mais probabilidade de produzir respostas factuais do que o GPT-3.5, segundo reportado pela OpenAI (OPENAI, 2023). Progredindo, em 2024 foi apresentado pela OpenAI o GPT-4o (“o” de *omni*), capaz de receber como entrada uma combinação de texto, áudio, imagem e vídeo e, diferentemente do GPT-4 que gerava somente textos, este tem a capacidade de gerar saídas em texto, áudio e imagem. Além disso, foi apresentada melhoria no tempo de resposta, possibilitando o uso de LLMs poderosas em aplicações de tempo real. Isso resultou na capacidade de resposta a uma entrada de áudio em 232 milissegundos (média de 320 milissegundos), o que é similar ao tempo de resposta humana em conversas. Logo, seu desempenho é comparável ao do GPT-4 Turbo, uma versão mais rápida do GPT-4, em textos em inglês e códigos, apresentando também melhorias notáveis em textos em outras línguas (OPENAI, 2024).

É importante destacar o alto custo estimado de treinamento desses modelos: o treinamento final do GPT-3 é estimado entre 500 mil dólares e 4,6 milhões de dólares (APPENZELLER; BORNSTEIN; CASADO, 2023). O treinamento do GPT-4 pode ter

custado cerca de 50 milhões de dólares, mas o custo total de treinamento é, provavelmente, superior a 100 milhões de dólares, visto as várias tentativas prévias antes do treinamento final (COST ESTIMATES FOR GPT-4, 2023). Isso demonstra o quanto essa tecnologia, apesar de poderosa e disruptiva, pode ser inacessível para a maioria das empresas e pesquisadores pelo seu alto custo.

Em resposta a esse movimento da OpenAI, diversas outras grandes empresas lançaram suas versões em paralelo como, por exemplo, o *Google* (família *Gemini* em atual versão 1.5), a *Antropic* (*Claude*) e a *Meta* (versão *opensource* do *Llama* que possibilitou um rápido crescimento dos modelos *opensource* desenvolvido pela comunidade como *Alpaca*, *Vicuna*, *Mistral* e *Bloom*). A partir disso, observou-se o início a um momento ímpar não só para a área de PLN, mas para a computação como um todo. Os modelos LLMs se tornaram a base da maioria das aplicações de IA desde então (BOMMASANI et al., 2021), sendo chamados de modelos de fundação (*Foundation Models*) por possibilitarem o desenvolvimento de inúmeras aplicações a partir do uso destes.

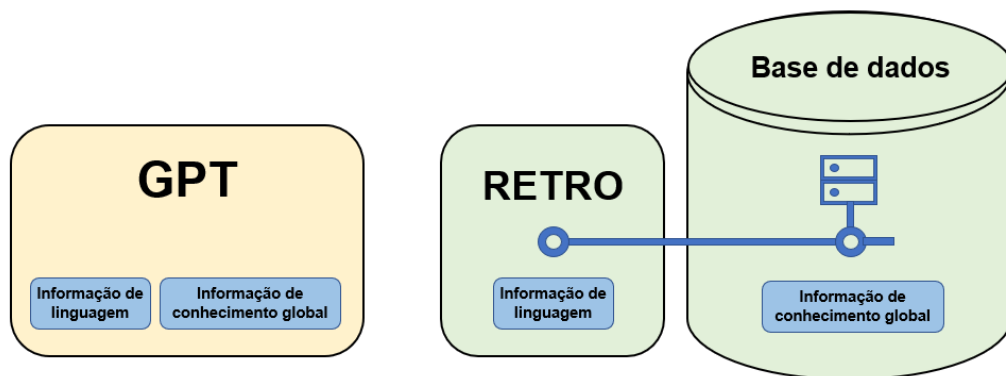
Apesar de todos os avanços e possibilidades obtidos através dos modelos LLMs, estas ainda são arquiteturas monolíticas, chamadas também de “caixa preta”, que retêm em seus parâmetros internos (pesos da rede neural) o conhecimento do idioma e todo o conhecimento do *dataset* utilizado no treinamento (OPENAI, 2023; KHATTAB; POTTS; ZAHARIA, 2021c). A natureza de “caixa preta” desses modelos leva a algumas importantes limitações:

- Ineficiência: o contínuo aumento desses modelos leva a ineficiências notáveis, de forma que modelos pequenos são capazes de ultrapassar um trilhão de parâmetros. Além de aumentar o impacto ambiental consideravelmente, os custos necessários à manutenção e ao treinamento dos modelos se tornam impeditivos à maioria das organizações.
- Opacidade: a codificação do “conhecimento” em pesos de modelo sintetiza o que é memorizado a partir de exemplos de treinamento. Como consequência, torna-se difícil discernir quais fontes foram utilizadas (se houver alguma) para realizar uma previsão. Isto se torna uma preocupação na prática pois, frequentemente, são geradas declarações que aparentam ser corretas dada a sua coerência gramatical, mas que são falsas.

- Estaticidade: alto custo para a atualização dos modelos. A atualização de modelos não ocorre de forma eficiente, de forma que é necessário um novo treinamento ou um ajuste fino no novo corpus com dados atualizados.

Baseando-se nisso, foi iniciado em 2021 um movimento com o objetivo de solucionar esses obstáculos, o que resultou em MLs que incorporam informações gerais de uma base externa e liberam os parâmetros do modelo para que o foco seja nas informações da linguagem e capacidade de raciocínio. Dentre as técnicas que utilizam essa abordagem, é possível destacar o *RETRO Transformer da DeepMind* (BORGEAUD et al., 2021) e o *WebGPT* da OpenAI (NAKANO et al., 2021). Essas técnicas demonstram que os MLs generativos menores podem apresentar desempenho semelhante aos de modelos maiores se forem otimizados com algum método que permita ao modelo a consulta a informações. Como exemplo, o modelo *RETRO* possui o desempenho semelhante ao GPT-3, apesar de ter somente 4% de seu tamanho (7,5 bilhões de parâmetros no *RETRO* vs 185 bilhões de parâmetros no GPT-3 Da Vinci) (BORGEAUD et al., 2021). A diferença entre paradigmas de ML apresentado pelo RETRO está ilustrada na figura 8.

Figura 8. Representação da diferença entre paradigmas de modelos de linguagem apresentado pelo RETRO.



Legenda: o RETRO incorpora informações recuperadas de um banco de dados para liberar seus parâmetros de serem um simples armazenamento de fatos e conhecimento gerais.

Fonte: adaptado de: <<http://jalamar.github.io/illustrated-retrieval-transformer/>>. Acesso em: 23 nov. 2023.

Essa abordagem foi amplamente difundida nos sistemas de pergunta e resposta e sistemas conversacionais como, por exemplo, o *REALM* (GUU et al., 2020), o *CoBERT-QA* (KHATTAB; POTTS; ZAHARIA, 2021a), o *Baleen* (KHATTAB; POTTS;

ZAHARIA, 2021b) e o RAG (LEWIS et al., 2020). Este último se difundiu de tal forma que passou a ser utilizado como o *pipeline* padrão para aplicações de LLM que necessitam de inserção de novas informações e busca em bases externas. Essa popularização indica que essa abordagem resolve, ou ao menos ameniza, os problemas citados anteriormente da seguinte maneira:

- Ineficiência: os modelos baseados em recuperação podem ser muito menores e mais rápidos. Ao contrário dos modelos de linguagem “caixa preta”, os parâmetros não necessitam mais armazenar uma lista cada vez maior de fatos, pois tais fatos podem ser recuperados. Ao invés disso, pode-se dedicar a utilização dos parâmetros para o aprendizado da linguagem e tarefas. Isso resulta em modelos menores e eficazes como, por exemplo, ColBERT-QA que atinge 47,8% *Exact Match (EM)* no *dataset OpenQuestion* de domínio aberto, enquanto um modelo T5-11B ajustado (com 24 vezes mais parâmetros) e um modelo GPT-3 (com 400 vezes mais parâmetros) atinge 34,8% e 29,9%, respectivamente (KHATTAB; POTTS; ZAHARIA, 2021a).
- Opacidade: os modelos baseados em recuperação oferecem um contrato transparente com os usuários. Ao produzir uma resposta, pode-se ler as fontes que ele recuperou e julgar sua relevância e credibilidade de forma manual. Isso é essencial para produzir maior confiança no usuário ao utilizar o modelo.
- Conhecimento estático: sendo os fatos armazenados como texto, o armazenamento de conhecimento de recuperação pode ser atualizado ou expandido com eficiência a partir da modificação do corpus do texto. Isso mantém constante a capacidade do modelo em encontrar e utilizar informações e reduz o custo computacional, acelerando a capacidade de alteração do conhecimento do sistema. Desta forma, os mantenedores das bases podem acrescentar um corpus específico de domínio e dispor daquele conhecimento para ser utilizado pelo sistema.

3.3 *Retrieval-Augmented Generation (RAG) nos Sistemas de Pergunta e Resposta (Question-Answering)*

A tarefa de pergunta e resposta (PR, em português ou QA, do inglês *Question-Answering*) é, de forma geral, a tarefa de retornar uma resposta à uma dada pergunta. Existem vários *pipelines* e técnicas para se resolver esse problema, desde a utilização de um único modelo geracional ajustado no domínio na qual a pergunta é realizada, ou métodos onde informações são recuperadas de bases de dados e utilizados como respostas diretamente, ou ainda sumarizadas em uma única resposta, ou até mesmo sistemas mais complexos que buscam informações de diferentes fontes e utilizam essas informações como contexto em um modelo geracional para entregar uma resposta mais natural e contextualizada (WANG, 2022).

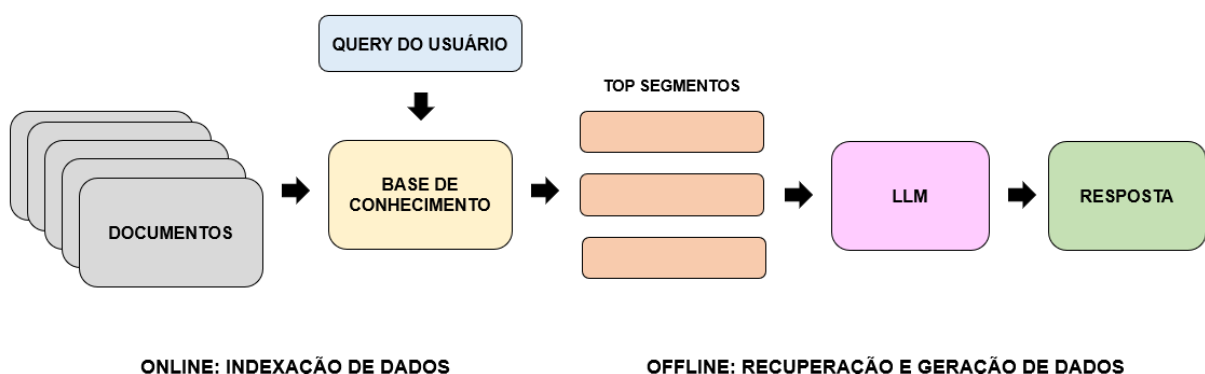
Devido às limitações dos modelos geracionais (LLMs) como, por exemplo, falhas no conhecimento do modelo, problemas de raciocínio para relacionar informações, alucinação, utilização de informações falsas na composição da resposta e dificuldade de inserção de novas informações, faz-se necessário uma estratégia para mitigar esses problemas (HUANG et al., 2023). O *Retrieval-Augmented Generation (RAG)*, é um *pipeline* que oferece uma solução para mitigar alguns desses problemas, fornecendo um melhor contexto às LLMs através de um conhecimento externo. O RAG é especialmente útil em cenários que exigem muito conhecimento ou em aplicações específicas de domínio que necessitam de informações constantemente atualizadas. Uma vantagem do RAG sobre outras abordagens é que o LLM não precisa ser retreinado para aplicações específicas (LEWIS et al., 2020).

Dessa forma, as duas principais abordagens para utilização de um sistema de PR, especialmente em um ambiente em que as respostas necessitam de acesso à informação não vista pelo modelo gerador, são: 1) *fine-tuning* de um modelo pré-treinado, como um LLM, e 2) RAG, utilizando um LLM como modelo gerador. Assim, pode-se traçar algumas comparações entre as duas abordagens: o RAG atualiza diretamente a base de conhecimento de recuperação, mantendo a informação atualizada sem necessidade de retreinamento frequente, enquanto o *fine-tuning* armazena dados estáticos e requer retreinamento para atualizações. O RAG é eficiente em utilizar recursos externos e requer processamento mínimo de dados, enquanto o *fine-tuning* depende da criação de conjuntos de dados de alta qualidade. A customização do RAG foca na integração de conhecimento externo, já o *fine-tuning*

permite o ajuste do comportamento e o estilo do modelo. O RAG proporciona maior interpretabilidade ao rastrear respostas até fontes de dados específicas, diferentemente do *fine-tuning* que pode ser uma "caixa preta". O RAG depende de recursos computacionais para estratégias de recuperação, o que pode levar a maior latência, enquanto o *fine-tuning*, após retreinamento, tem menor latência. RAG é menos propenso a alucinações, pois cada resposta é baseada em evidências recuperadas, ao contrário do *fine-tuning* que pode ainda exibir alucinações com entradas desconhecidas. Por fim, questões éticas e de privacidade no RAG surgem do armazenamento e recuperação de texto de bancos de dados externos, enquanto no *fine-tuning* surgem do conteúdo sensível nos dados de treinamento (GAO et al., 2024).

Pode-se observar que, apesar de existirem vantagens e desvantagens em ambas as abordagens, elas não são mutuamente excludentes e, se utilizadas juntas, podem se complementar. Porém, em uma área em que o conhecimento está em constante evolução, o custo de *fine-tuning* pode inviabilizar a sua implementação. Uma estratégia híbrida é realizar o *fine-tuning* com foco no comportamento (figura 9), em que o modelo segue um certo estilo e tom de resposta desejado e a agregação de conhecimento é responsabilidade para o *pipeline* de RAG.

Figura 9. Abordagem geral do pipeline de Retrieval-Augmented Generation (RAG) para aplicações baseadas em Large Language Models (LLM).



Legenda: LLM: large language models.
Fonte: próprio autor.

Com essa visão geral de técnicas resolver a tarefa de PR, pode-se definir o *pipeline* base de PR utilizando RAG e LLM em quatro principais etapas: indexação da base de conhecimento, processamento da *query* e criação do *prompt*, recuperação de

segmentos de texto que possam conter informações para a resposta (*retriever*) e Geração da resposta (*reader* ou *generator*) (LEWIS et al., 2020).

A primeira etapa, indexação da base de conhecimento, é o único processo que acontece de forma *offline*, ou seja, fora do fluxo de PR ou interação com o usuário. Essa etapa consiste no processamento de documentos, fragmentando-os em segmentos; traduzindo-os, se necessário; extraindo informações extras como, por exemplo, entidades, resumos; e outros pré-processamentos que vão depender do formato esperado pelo banco de dados em que serão salvas essas informações. Em seguida é realizado a inserção dessas informações em um banco de dados, que será posteriormente consumido pelo *retriever* durante o *pipeline* de PR (LEWIS et al., 2020).

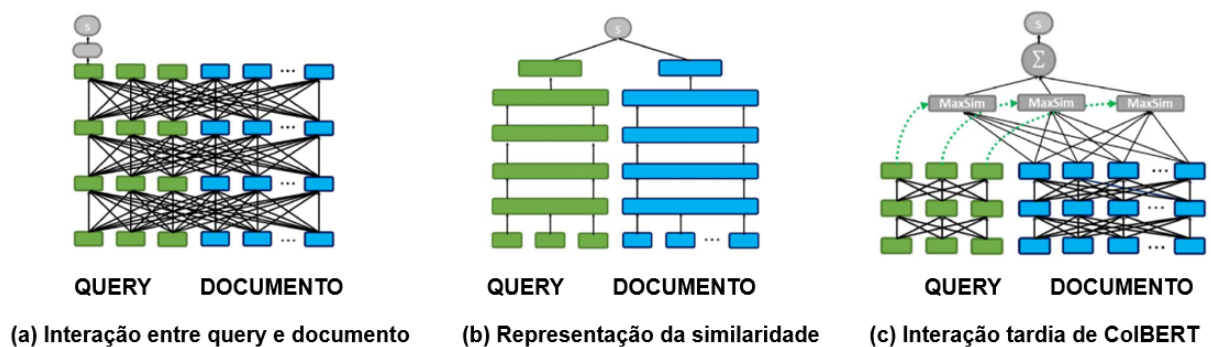
A segunda etapa, processamento da *query* e criação do *prompt*, ocorre assim que uma mensagem do usuário é recebida (*query*) e diferentes processamentos opcionais podem ser aplicados como, por exemplo, classificação da pergunta para melhor definição de processos específicos, tradução, parafraseamento, cálculo de *embedding*, entre outras. Uma vez processada, a *query* segue para dois caminhos: 1) inserção em um *prompt* pré-definido que contém todas as diretrizes de funcionamento do modelo LLM como, por exemplo, qual é o formato que ele deve responder, idioma, formalidade da resposta e todas as regras e personalizações que convêm para o negócio; ou 2) será enviada para o passo seguinte de *retriever* (LEWIS et al., 2020).

A terceira etapa, recuperação de segmentos de texto que possam conter informações para a resposta (*retriever*), ocorre quando a *query* é utilizada em um mecanismo de *Information Retrieval* (IR) que normalmente está incorporado no banco de dados onde foi indexado o conhecimento e os primeiros *k* documentos recuperados são retornados. Os documentos podem ser parágrafos, sentenças, páginas e/ou outras, formato que depende do processo utilizado para construir a base e do banco de dados utilizado. O *rank* é, então, definido pelo *score* do algoritmo de busca utilizado na busca semântica utilizando *embeddings*, o que resultando na distância cosseno entre cada documento e a *query* (XIONG et al., 2021; KARPUKHIN et al., 2020; LEWIS et al., 2020; MANNING; CHRISTOPHER; PRABHAKAR, 2008).

Tradicionalmente em IR, as tarefas de recuperação eram conduzidas usando modelos de *bag of words* como, por exemplo, o BM25 que buscam documentos que contêm os mesmos *tokens* da consulta (ROBERTSON; ZARAGOZA, 2009). A tarefa de *retriever* foi revolucionada com o BERT para o *ranking*. Na primeira abordagem

apresentada, cada documento é concatenado com a consulta e ambos são dados como entrada para um modelo BERT ajustado para estimar a relevância entre texto e consulta. Com isso, o BERT dobrou a métrica que avalia se a resposta está no top dez dos candidatos (MRR@10) sobre o BM25 em um dos *datasets* mais populares, o *MS MARCO Passage Ranking* (NGUYEN et al., 2016). Porém, apesar das melhorias de resultado, ele apresentou uma importante limitação de desempenho, pois pontuar cada par consulta-documento requer bilhões de operações computacionais. Como resultado, o BERT só pode ser utilizado na prática como um reranqueador, em que dado os top-k candidatos já extraídos por métodos mais simples como BM25, é gerado um novo *ranking* de respostas com maior acurácia, como pode ser observado na figura 10.

Figura 10. Diferentes arquiteturas que podem ser utilizadas para busca semântica (*retriever*).



Fonte: adaptado de Khattab, Potts e Zaharia (2021c).

A causa fundamental dessa limitação é a codificação conjunta de consulta e documentos candidatos. Por isso, a evolução natural foi resolver esse problema na arquitetura, originando uma abordagem que trata pergunta e resposta de forma independente. Assim, um *embedding* é gerado para consultas e para respostas separadamente que podem ser posteriormente comparados por operação entre vetores, como distância cosseno. Isso permitiu que uma base de *embeddings* dos documentos candidatos fosse criada previamente, de forma que sempre que uma pergunta é apresentada, apenas é necessário calcular o seu *embedding* e utilizar uma ferramenta de cálculo de distância vetorial para encontrar os top-k candidatos. Alguns

exemplos dessas arquiteturas são *SBERT* (REIMERS; GUREVYCH, 2019) e *DPR* (KARPUKHIN et al., 2020).

Porém, apesar da codificação independente entre a consulta e os documentos, isso também pode ser visto como uma deficiência, pois a comparação posterior se torna apenas um produto escalar, que não consegue relacionar tão bem quanto os modelos de ranqueamento. Por isso, fez-se necessário a criação de um modelo que fosse eficiente como os anteriores (*SBERT* e *DPR*), mas com uma qualidade de comparação semântica como os reranqueadores (*BERT*). Assim, foram geradas arquiteturas como o *CoBERT* (KHATTAB; POTTS; ZAHARIA, 2021b) que recebe os *embeddings* pré-calculados e utiliza uma camada de *MaxSIM* para relacionar os *embeddings* que são, então, somados à uma camada de saída para gerar uma pontuação de *match*.

Em 2022, OpenAI lançou um modelo que teve como objetivo substituir o uso de múltiplos *embeddings* especialistas utilizados em tarefas como, por exemplo, pesquisa, similaridade e pesquisa de código fonte, tornando-se o estado da arte na maioria das tarefas e apresentando um custo 99,8% menor que o antecessor. Esse modelo, intitulado *Ada02* é utilizado na atualidade como um dos melhores geradores de *embeddings* disponíveis (OPENAI, 2022).

É importante destacar que uma prática bastante comum na fase de *retriever* é a busca híbrida que utiliza tanto a busca vetorial (ou semântica) por distância cosseno nos *embeddings*, quanto a pesquisa baseada em palavra-chave, como BM25. Tanto a pesquisa baseada em palavras-chave, quanto a pesquisa vetorial retornam um conjunto separado de resultados, geralmente, em formato de uma lista de resultados de pesquisa classificados por sua relevância calculada por cada método. Existem várias técnicas para realizar a combinação desses resultados como apresentado por Benham e Culpepper (2017). Um dos métodos mais comuns é o ponderamento sobre o quanto o *score* de um método será incorporado em um *score* final, que permitirá um novo *rank* como é apresentado na fórmula matemática abaixo (fórmula 1).

Fórmula 1. Representação matemática do recálculo da fusão de múltiplos *retrievers* (denso e esparso).

$$hybrid_score = (1 - alpha) * sparse_score + alpha * dense_score$$

Legenda: Onde alpha é um valor entre 0 (considera puramente o resultado da busca por palavra-chave) e 1 (considera puramente o resultado da busca semântica).

Fonte: Benham e Culpepper (2017).

Uma abordagem de busca híbrida é considerada ideal para cenários em que é necessário tanto a busca semântica para uma experiência mais natural e semelhante à humana, quanto de correspondência exata de frases para termos específicos como, por exemplo, nomes de produtos, números, códigos de sistemas (HANEY, 2023). Enquanto *queries* curtas geram *embeddings* por falta de contexto, o que dificulta a abordagem de distância semântica, a busca por palavra-chave se beneficia de poucas palavras na *query*, pois reduz a quantidade de *match* na base. De forma complementar, uma sentença muito grande como *query* faz com que a busca por palavra-chave retorne muitos *matches* com diferentes significados semânticos. Quando comparado à busca semântica, observa-se que esta é capaz de codificar o contexto dessa sentença e encontrar documentos semanticamente mais próximos. Dessa forma, pode-se observar um complemento das duas técnicas, indicando que a busca híbrida é um caminho promissor para a otimização do *retriever*.

A quarta etapa do *pipeline* base de PR utilizando RAG e LLM, geração da resposta (*reader* ou *generator*), é a fase final para todas as passagens candidatas e estas são submetidas a um extrator ou gerador de resposta. Uma opção é utilizar um reranqueador para a escolha dos documentos candidatos com maior probabilidade de conter a resposta, normalmente um modelo *dual encoder* onde a *query* e cada documento é submetido e o modelo retorna um *score* representando o quanto o documento responde a *query*. Uma outra opção é a utilização dos documentos retornados para gerar um resumo dos mesmos e este ser retornado como resposta. Porém, com a evolução dos modelos geracionais e surgimento de LLMs como GPT4, *Gemini* e *Llama*, essa se tornou a ferramenta padrão de geração de resposta. Além disso, com o uso do pipeline de RAG é possível alimentar a LLM com o contexto relevante para a geração de uma resposta mais natural (LEWIS et al., 2020).

Retrieval-Augmented Generation (RAG) é uma arquitetura que combina modelos de recuperação de informações (retrieval) com modelos de geração de linguagem (generation), permitindo que sistemas de linguagem acessem informações externas para melhorar a qualidade e precisão das respostas geradas. A ideia principal é usar uma base de conhecimento externa para guiar a geração de respostas, proporcionando fatos e dados que complementam o modelo de linguagem.

Formalmente o RAG pode ser descrito como uma combinação de dois componentes principais (LEWIS et al., 2020):

1. Componente de Recuperação (Retrieval): Este componente recebe uma consulta q e utiliza um banco de dados de documentos D para recuperar os documentos mais relevantes. O processo de recuperação pode ser descrito como:

$$\text{Retrieval}(q, D) = \{d_1, d_2, \dots, d_k\}$$

Onde:

- q é a consulta (prompt ou entrada de usuário),
- D é o banco de dados ou coleção de documentos,
- $\{d_1, d_2, \dots, d_k\}$ é o conjunto de documentos relevantes recuperados pelo modelo de recuperação, com k sendo o número de documentos retornados.

2. Componente de Geração (Generation): Após a recuperação, o modelo de linguagem gera uma resposta baseada na entrada original q e nos documentos recuperados $\{d_1, d_2, \dots, d_k\}$. O processo de geração pode ser formalizado como:

$$p(y \mid q, d_1, d_2, \dots, d_k)$$

Onde:

- $p(y \mid q, d_1, d_2, \dots, d_k)$ é a probabilidade de gerar a sequência y (resposta) dado a consulta q e os documentos recuperados $\{d_1, d_2, \dots, d_k\}$.

O modelo RAG pode combinar o Retrieval e o Generation para recuperar os documentos relevantes e, em seguida, condicionar a geração do texto nas informações recuperadas. O procedimento completo pode ser expresso

matematicamente pela decomposição em dois passos:

1. Recuperação: Escolha de documentos relevantes a partir de uma distribuição de documentos condicionada à consulta q :

$$p(d_i | q) \propto \text{score}(q, d_i)$$

2. Geração: Geração da resposta y , condicionada à consulta q e aos documentos $\{d_1, d_2, \dots, d_k\}$:

$$p(y | q) = \sum p(y | q, d_i) * p(d_i | q)$$

Ou seja, a geração da resposta y depende de uma soma ponderada das probabilidades de gerar a resposta com base em cada documento recuperado. Dessa forma, pode-se resumir matematicamente o RAG como:

- Recuperação: $\text{Retrieval}(q, D) = \{d_1, d_2, \dots, d_k\}$
- Geração: $p(y | q, d_1, d_2, \dots, d_k)$
- Combinação: $p(y | q) = \sum p(y | q, d_i) * p(d_i | q)$

Apesar dessas etapas descreverem bem um *pipeline* de RAG, e ser possível a generalização para variações do mesmo, diversas propostas de otimização surgiram para esse *pipeline*, como pode ser visto na figura 18 (GAO et al., 2024). A maioria das propostas estão relacionadas aos dois principais componentes do RAG: *retriever* e *generator*. Alguns trabalhos focam no aperfeiçoamento na representação do conhecimento, melhor alinhamento entre *query* e documentos na base, melhor alinhamento entre a saída do *retriever* e a resposta gerada pela LLM. Além destes, também foram propostos métodos de *fine-tuning* de LLM considerando o RAG.

Atualmente, um terceiro componente do RAG tem recebendo bastante atenção e foco na pesquisa: os métodos de integração do contexto extraído no RAG. Estes podem ser dividido em:

- *Augmentation Stage*: os três estágios possíveis de incorporação são: 1) durante o pré-treino do modelo LLM como, por exemplo, o *RETRO* (BORGHAUD et al., 2021); 2) *fine-tuning*, similar ao pré-treino, mas com foco no *fine-tuning*; e 3)

durante inferência, que é o caso mais comum, através do módulo de *retriever*, em que segmentos de textos são buscados em uma base externa de forma a contextualizar a geração da resposta.

- *Augmentation Source*: a origem e formato da informação utilizada pelo RAG tem um impacto importante no resultado final, por isso, diferentes tipos são explorados nesses trabalhos como, por exemplo, dado não estruturado (texto não organizados, vídeos, áudios), dado estruturado (banco de dados SQL, planilhas *Excel*) e conteúdo gerado por modelos LLM (*datasets* sintéticos).
- *Augmentation Process*: alguns problemas carecem de múltiplos passos de busca de informação. Dessa forma, três métodos são mais frequentes na literatura, sendo eles: 1) interativo, permitindo a execução do módulo *retriever* múltiplas vezes (ARORA et al., 2023; BORGEAUD et al., 2021); 2) recursiva, onde a saída de um *retriever* guia a chamada do próximo (TRIVEDI et al., 2022; KIM et al., 2023); ou 3) adaptativa, onde o quando e qual base de conhecimento será utilizada pelo retriever é definido de forma dinâmico (JIANG et al., 2023; ASAI et al., 2023).

Além desses componentes do RAG, uma outra importante classificação apontada por Gao et al. (2024) é a de seus diferentes paradigmas baseados nas limitações de performance, custo e eficiência dos mesmos:

- *Naive RAG*: é o processo mais tradicional, apresentado anteriormente, que segue os passos de indexação, recuperação e geração. A *query* do usuário é utilizada para buscar documentos relevantes que são, em seguida, inseridos à um *prompt* e enviado ao modelo para geração de uma resposta final. Em diálogos de múltiplas interações, o histórico da conversa pode ser integrado ao *prompt*. Essa abordagem apresenta limitações como, por exemplo, baixa precisão e *recall* durante a fase de recuperação de informação (*retriever*), além de não apresentar ferramentas para lidar com a redundância de informação.
- *Advanced RAG*: esse método adiciona dois módulos ao processo anterior: 1) a *Pré-Retriever* que tem como objetivo melhorar a qualidade dos documentos recuperados na fase de *retriever*, agindo principalmente na *query* do usuário ao aplicar técnicas de expansão de *query*, parafraseamento, correção ortográfica, tradução e outros; e 2) a *Pós-Retriever* que tem como objetivo a redução do ruído que é retornado pelo *retriever*, gerando um contexto rico pra

produção da resposta. Esse processo é realizado por métodos como reranqueamento dos documentos, sumarização de documentos e *prompt* e fusão de informações relevantes em diferentes documentos.

- Modular RAG: é a abordagem mais flexível, tratando cada sub processo do RAG como um módulo que pode ser trocado, adicionado ou retirado do *pipeline*, de forma a possibilitar a reconstrução dessa abordagem com as abordagens anteriores. Alguns módulos podem utilizados ser para busca de informação, memória, fusão de conhecimento, predição, sumarização, e geração, de forma que cada módulo seja especialista em uma tarefa e ainda possibilita o rearranjo em diferentes *pipelines*. Por isso, essa abordagem é considerada flexível e adaptável à diferentes cenários e requisitos.

No entanto, mesmo com a capacidade dos modelos LLMs e a versatilidade que o RAG trouxe, algumas limitações importantes se mantiveram. A simples pesquisa de similaridade vetorial pode não ser suficiente quando o LLM precisa de informações de vários documentos ou mesmo apenas de vários segmentos do mesmo documento para gerar uma resposta. Por exemplo, dado a pergunta: “Quais dos empregados da empresa X fundaram suas próprias empresas?”, pode-se quebrar essa tarefa em duas: 1) “Quem são os empregados da empresa X?” 2) “Algum deles fundou sua própria empresa?” Perguntas como estas necessitam de buscas em diferentes partes de documentos e em diferentes documentos, também conhecido como *multi-hop QA* e ilustrado na figura 11 (MIN et al., 2019; YANG et al., 2018).

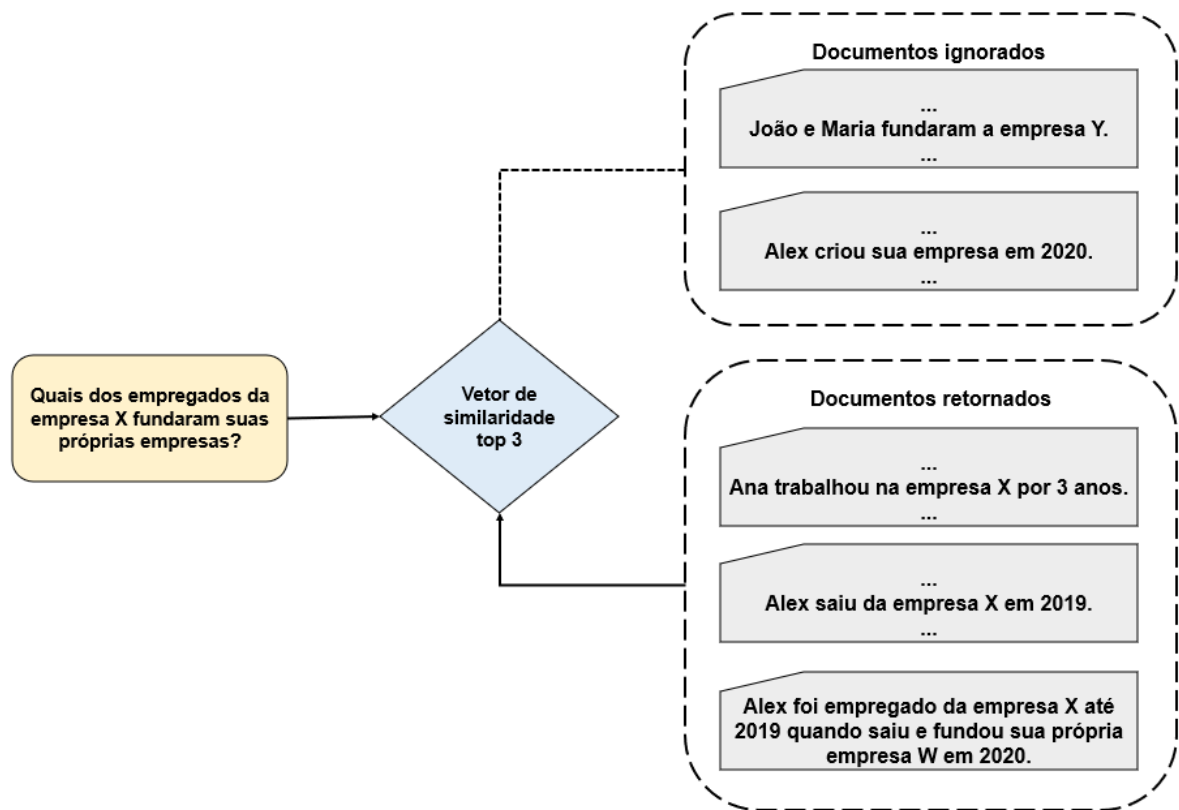
Os principais problemas que podem ser encontrados na busca por similaridades vetoriais que impedem a solução desse tipo de problemas são: repetição de informações nos top K documentos retornados pelo *retriever*; perda de referências de entidades através dos documentos; e dificuldade em definir a quantidade de documentos necessários para uma dada pergunta (LEWIS et al., 2020).

A repetição de informações nos top K documentos retornados pelo *retriever* ocorre em razão da distância semântica focar na busca de documentos mais próximos da pergunta e não informações complementares ou que tenha qualquer relação direta com outros documentos. Por exemplo, se um dos empregados da empresa X é mais presente nos documentos, pode ser que os top K documentos contenham informações sobre ele e sobre empresas que ele fundou depois, resultando na desconsideração de outros fundadores que aparecem em menor frequência nos documentos (LEWIS et al., 2020).

A perda de referências de entidades através dos documentos é decorrente do fato de que cada segmento do documento não possui uma relação direta ao banco. Dessa forma, a informação sobre uma mesma entidade, nesse caso de alguma pessoa, pode ser perdida no decorrer do documento devido a forma que ele foi segmentado. Por exemplo, considerando a existência de um documento em que a primeira página cita os empregados de uma empresa X e, em outra parte do documento, a ocorrência da informação de quais empresas foram fundadas pelos empregados, essa informação não pode ser relacionada entre elas durante a busca que procura explicitamente a relação da pergunta com cada segmento de forma isolada. Esse problema pode ser reduzido por meio de técnicas de resolução de correferência ou por meio da ampliação do tamanho dos segmentos (MANNING; RAGHAVAN; SCHÜTZE, 2008; SOON; NG; LIM, 2001). No entanto, estas técnicas normalmente solucionam somente casos onde as informações estão em um mesmo documento.

A dificuldade na definição da quantidade de documentos necessários para uma dada pergunta é considerada um obstáculo em razão da complexidade variável de cada pergunta e também da complexidade das informações dispostas na base de conhecimento. Por vezes, a resposta pode estar contida em um único parágrafo, mas em outras situações podem ser necessárias várias páginas de um documento para que uma resposta completa seja gerada. Vale ressaltar que, o aumento de documentos retornados pelo *retriever*, potencialmente aumenta a quantidade de informações irrelevantes para responder à pergunta (NOGUEIRA; CHO, 2019).

Figura 11. Representação de um exemplo em que a pesquisa por similaridade retorna informações duplicadas, enquanto outras informações relevantes podem ser ignoradas em decorrência de um baixo número K (três, neste caso) ou à distância semântica que pode não ser grande o suficiente para entrar no top K.



Fonte: próprio autor.

3.5. Agentes Conversacionais Inteligentes (AI Agents)

Com o avanço de modelos de IA, aumento da capacidade de solucionar problemas diversos de PLN e o aperfeiçoamento da capacidade de conversação, a ideia de um sistema inteligente com capacidades humanas se mostrou mais próximo da realidade. Logo, a evolução natural de sistemas de PR é a adição da capacidade para além da resposta às perguntas, criando interações de diversas formas e, promovendo assim, o conceito de agente inteligente conversacional (AIC). Tanto as pesquisas de IA com foco em agentes (DEEPGRAM, 2024) quanto ferramentas profissionais para construção de agentes cresceram devido as capacidades dos modelos LLMs (BUBECK et al., 2023).

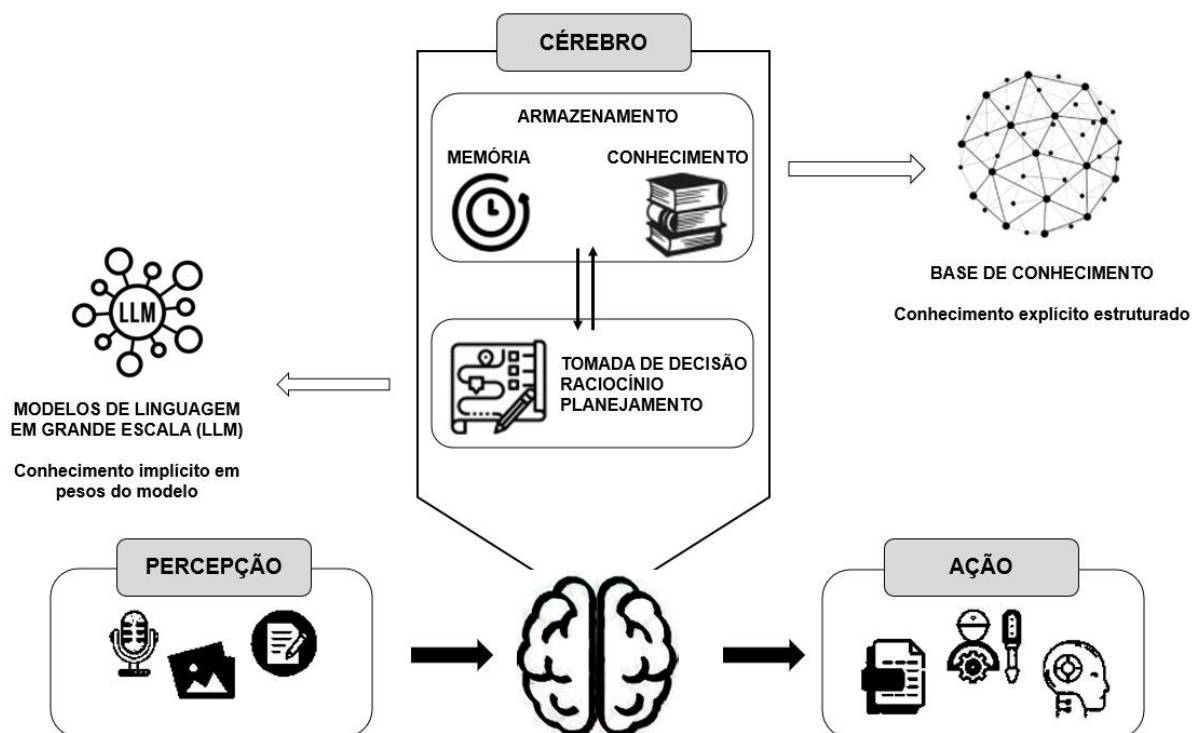
Teoricamente, um agente inteligente (AI) é uma entidade artificial, com capacidade de sentir o ambiente, tomar decisões e realizar ações, logo, seus três componentes principais podem ser divididos entre: percepção, cérebro e ações. A figura 12 apresenta um framework simplificado de um agente e a interação entre seus componentes principais, sendo eles:

- Percepção: é o processo de receber uma entrada do usuário, normalmente uma pergunta, mas pode ser qualquer tipo de interação com o ambiente como, por exemplo, uma requisição http, uma imagem, um áudio ou outros.
- Cérebro: local em que ocorre a tomada de decisões a cada interação baseado na sua memória e conhecimentos. Esse modulo contém dois submódulos principais, memória e decisor.
 - Memória: é o módulo responsável por armazenar conhecimentos e experiências passadas. A memória pode ser dividida em memória de curto prazo e memória de longo prazo. A primeira é responsável pelas informações da situação atual (conversa que está em curso) e é normalmente inserida no contexto do agente a cada interação que ele realiza com o ambiente. A segunda é responsável pelo conhecimento retido por um longo período de tempo, normalmente contido em um banco de dados externo e utilizado um *pipeline* de RAG para acesso em tempo real.
 - Decisor: é responsável por tomar decisões e planejar as ações futuras, auxiliando na decomposição da requisição em subtarefas e na definição das etapas necessárias em para atingir o objetivo. É representado por

um LLM de propósito geral que é capaz de receber um *prompt* com instruções de comportamento, lista de ações possíveis, personalidade, lista de ferramentas disponíveis e outros necessários. O processo de planejamento pode ser visto como um subprocesso a parte dentro do decisor, alguns exemplos de técnicas de planejamento são *Chain of Thought* (WEI et al., 2022) e *Tree of Thoughts* (YAO et al., 2023; LONG, 2023). Normalmente, este é um módulo separado, mas pode ser simplificado e integrado ao cérebro do agente.

- Ações: são, normalmente, representadas por ferramentas, que são módulos que permitem que o agente reaja e interaja com ao ambiente, seja através da geração de um texto, ou acesso a APIs externas, como por exemplo, *wikis*, interpretadores de códigos, bases de terceiros, sistemas de pagamento entre outros.

Figura 12. Representação simplificada de um agente inteligente e sua interação entre seus componentes principais.



Fonte: adaptado de Xi et al. (2023).

Apesar de grande esforço para desenvolvimento dos componentes necessários para um AI como, por exemplo, a otimização de *fine-tuning* de LLMs, algoritmos de busca e aperfeiçoamento em tarefas específicas, poucos trabalhos apresentam um modelo de propósito geral para servir como base para um AI que seja capaz de atuar em diversos cenários e unificar as pesquisas de alguma forma. Contudo, recentemente alguns trabalhos apresentam propostas para solucionar essa lacuna que se tem nos estudos e desenvolvimentos de AIs.

No trabalho apresentado por Xi et al. (2023), existe uma tentativa de redução nessa lacuna acadêmica que foi inspirada pela definição histórica de agente e apresenta um *framework* conceitual, composto por três componentes: percepção, cérebro e ações. A partir disso, são discutidas as aplicações práticas desses agentes e elucidado o futuro de ambientes multiagentes e as razões pelos modelos LLM serem os componentes principais que possibilitaram a existência destes agentes.

Vários trabalhos recentes apresentam um *survey* para auxiliar no mapeamento dessa área de pesquisa por meio de categorizações e definições formais de agentes, como Guo et al. (2024), que oferece uma discussão aprofundada sobre os aspectos essenciais dos sistemas multiagentes baseados em LLMs, incluindo os desafios, os domínios e ambientes que esses agentes simulam. Outro importante trabalho é o de Durante et al. (2024) que define um AI como uma classe de sistemas interativos que podem perceber estímulos visuais, entradas de linguagem e outros dados baseados no ambiente para produzir ações significativas e que podem alterar o ambiente. Com o a evolução das técnicas de AR, alguns trabalhos como o de Yuan et al. (2023) apresentam um estudo sobre o progresso de multiagentes cooperativos utilizando AR em ambientes abertos, aprofundando-se nos métodos de decomposição de MARL e outros conceitos matemáticos essenciais para esse ambiente.

Um importante aspecto dessa tecnologia emergente é a segurança e Chan et al. (2024) avaliam medidas para aumentar a visibilidade em AI, incluindo identificadores de agentes, monitoramento em tempo real e análise retrospectiva. Assim, os autores discutem sobre os desafios e oportunidades para garantir transparência e responsabilidade nas implantações de agentes de IA. Atualmente, ainda existe uma extensa lista de ferramentas disponíveis no mercado que facilitam o processo de criação do agente com foco em diferentes aspectos e customização como, por exemplo *Langchain*, *LlamaIndex*, *OpenAgents*, *crewAI*, *AutoGPT* e

outras. Isso tem difundido e popularizado ainda mais o uso de AI como ferramentas do cotidiano das pessoas.

Os AI apresentam benefícios não somente a nível de usuário final, mas também em nível gerencial, suporte administrativo, acompanhamento de pós-vendas e outros. Isso é decorrente dos AI aprimorarem a eficiência e produtividade ao lidar com solicitações complexas sem a necessidade de um agente humano, permitindo que as equipes de suporte se concentrem em trabalhos mais significativos. Os AIs coletam informações, organizam o *backlog* de *tickets* e enviam automaticamente pesquisas de CSAT, reduzindo o trabalho manual dos agentes humanos. Além disso, os AIs oferecem suporte sem interrupções, garantindo que os clientes sempre tenham assistência, o que pode impactar na satisfação e fidelidade aos serviços que utilizam AIs. A partir da integração aos sistemas de *backend*, os AIs utilizam os dados dos clientes para personalizar respostas e enviam detalhes importantes aos agentes humanos para que eles possam tomar as melhores decisões. Além disso, eles podem analisar conversas para identificar tendências e áreas de melhoria nas operações de suporte, sugerindo automações para atender proativamente às necessidades dos clientes. Ao possibilitarem essas tarefas descritas, são capazes de reduzir os custos operacionais associados ao solucionar as solicitações de forma independente do agente humano, permitindo que as empresas invistam recursos em outras áreas estratégicas.

Casos notáveis de agentes baseados em LLMs existentes no mercado incluem uma variedade de aplicações em diferentes setores. Ma et al. (2023) analisam a eficácia de agentes conversacionais para o suporte ao bem-estar mental, descobrindo que esses agentes podem ajudar a lidar com ansiedades, mas também podem produzir conteúdo prejudicial. Horton (2023) explora comportamentos econômicos humanos em cenários simulados usando agentes com dotação, preferências e personalidades. Generative Agents (PARK et al., 2023) e AgentSims (LIN et al., 2023) visam simular a vida diária em uma cidade virtual por meio da construção de múltiplos agentes. Blind Judgement (HAMILTON, 2023) emprega vários modelos de linguagem para simular processos de tomada de decisão de juízes, prevendo decisões da Suprema Corte com precisão superior ao acaso. Ziems et al. (2023) apresentam agentes que auxiliam pesquisadores na geração de resumos, roteiros e extração de palavras-chave. ChemCrow (BRAN et al., 2023) é um agente de química LLM que utiliza bancos de dados químicos para planejar e executar autonomamente a síntese

de repelente de insetos, três organocatalisadores e a descoberta orientada de um novo cromóforo. Boiko et al. (2023) combinam múltiplos LLMs para automatizar o design, planejamento e execução de experimentos científicos. Math Agents assistem pesquisadores em explorar, descobrir, resolver e provar problemas matemáticos. EduChat (DAN et al., 2023) e CodeHelp (LIFFITON et al., 2023) são exemplos notáveis de agentes LLM destinados à educação. Mehta et al. (2023) propõem um framework interativo que permite a arquitetos humanos interagir com agentes de IA para construir estruturas em um ambiente de simulação 3D. ChatDev (QIAN et al., 2023), TooLLM (QUIN et al., 2023) e MetaGPT (HONG et al., 2023) mostram potencial para automatizar codificação, depuração, testes e auxiliar em outras tarefas de engenharia de software. D-Bot (ZHOU; LI; LIU, 2023) é um administrador de banco de dados baseado em LLM que adquire continuamente experiência em manutenção de banco de dados e oferece diagnósticos e conselhos de otimização. IELLM (OGUNDARE; MADASU; WIGGINGS, 2023) aplica LLMs para desafios na indústria de óleo e gás. Dasgupta et al. (2023) apresentam um sistema de agentes unificado para raciocínio incorporado e planejamento de tarefas. Finalmente, o OS-Copilot (WU et al., 2024) é um framework para construir agentes generalistas capazes de interagir com elementos abrangentes de um sistema operacional, incluindo a web, terminais de código, arquivos, multimídia e várias aplicações de terceiros.

3.6 Avaliação de Pergunta e Resposta, *Retrieval-Augmented Generation* (RAG) e Agentes Inteligentes

Existem diversas métricas de avaliação de tarefas em PLN como, por exemplo, as tradicionais métricas de similaridade léxicas como, por exemplo, o *BLEU* (*BiLingual Evaluation Understudy*) (PAPINENI et al., 2002) que é métrica utilizada para avaliar a qualidade das traduções automáticas através da medida da sobreposição entre os n-gramas da *string* prevista e a esperada; o *ROUGE* (*Recall-Oriented Understudy for Gisting Evaluation*) que utiliza a sobreposição de n-grams (ROUGE-N) (LIN; HOVY, 2003) ou sobreposição de sequências mais longa (ROUGE-L) (LIN; OCH, 2004); e o *METEOR* (BANERJEE; LAVIE, 2005) que aborda a mesma tarefa, mas visa melhorar o BLEU ao utilizar uma média harmônica ponderada de *precision* e *recall* da sobreposição dos uni-grams.

Porém, a validação de sistemas de conversacionais pode ser um grande desafio, visto que além de avaliar a qualidade do *pipeline* como um todo (ponta a ponta), é importante avaliar os módulos individualmente, pois dependendo do módulo que está com desempenho baixo, as decisões são diferentes. Por exemplo, se o *retriever* está com desempenho baixo, talvez seja necessário realizar um ajuste fino no domínio dos dados ou alterar a representação das informações. Mas se é o *generator* que está apresentando desempenho baixo, talvez a qualidade do contexto retornado da base de conhecimento não esteja adequada ou pode ser necessário a adição de informações ao contexto como, por exemplo, qual é a relações entre diferentes documentos.

Os *datasets* de PR podem apresentar perguntas com uma única resposta correta ou múltiplas, assim como alguns *datasets* apresentam a opção de não ter resposta com o valor “None”. Atualmente, existem diversos *datasets* de PR, de domínios de conhecimento específicos e gerais com diferentes propostas de complexidade como, por exemplo, respostas curtas, respostas longas e entidades (KWIATKOWSKI et al., 2019).

A partir disso, a avaliação pode ser feita de duas formas: domínio aberto ou domínio fechado. Avaliação de domínio fechado oferece um resultado mais confiável, mas necessita que o conjunto com a resposta seja extraído do texto de referência, ou contexto, além dos índices de início e fim de onde foi extraída a resposta. Assim, é possível comparar se a resposta foi extraída de onde foi anotado, assim como do documento esperado. Apesar da confiabilidade no resultado, a subjetividade da tarefa pode julgar como “errado” respostas que poderiam ser consideradas como corretas por um ser humano. Por outro lado, a avaliação em domínio aberto permite que apenas a *string* de saída ou até partes dela sejam comparadas com a *string* esperada, ou seja, mesmo que a *string* tenha sido extraída de diferentes locais e documentos da anotação, o que é relevante é a sobreposição de *strings*. Isto permite uma avaliação menos restritiva e é indicada quando se sabe que uma resposta pode ser encontrada em diferentes partes da base (CHEN et al., 2017; VOORHEES; TICE, 2000).

Quando se deseja avaliar o *retriever*, tem-se que se observar o objetivo dele em retornar candidatos a respostas. Logo, deve ser definido uma quantidade de candidatos a ser considerado na avaliação, normalmente chamado de topK, onde K é a quantidade de candidatos e é avaliado se a resposta esperada está dentre esses candidatos. Dessa forma, utiliza-se a métrica *Recall* para medir quantas vezes o

documento com a resposta correta estava entre os documentos recuperados. O valor do *Recall* para pergunta é binário (contém a resposta ou não contém), mas em um *dataset* com as perguntas o recall retornará um valor entre 0 e 1 que mede a porcentagem de vezes em que o *retriever* retornou a resposta entre os candidatos. Porém, às vezes é desejado medir o quão alto na lista de candidatos o documento com a resposta correta se encontra e para isso se utiliza a métrica *mean reciprocal rank* (MRR). Para isto, utiliza-se o *score* que o *retriever* retorna, normalmente a distância cosseno entre a pergunta e o documento (POWERS, 2020).

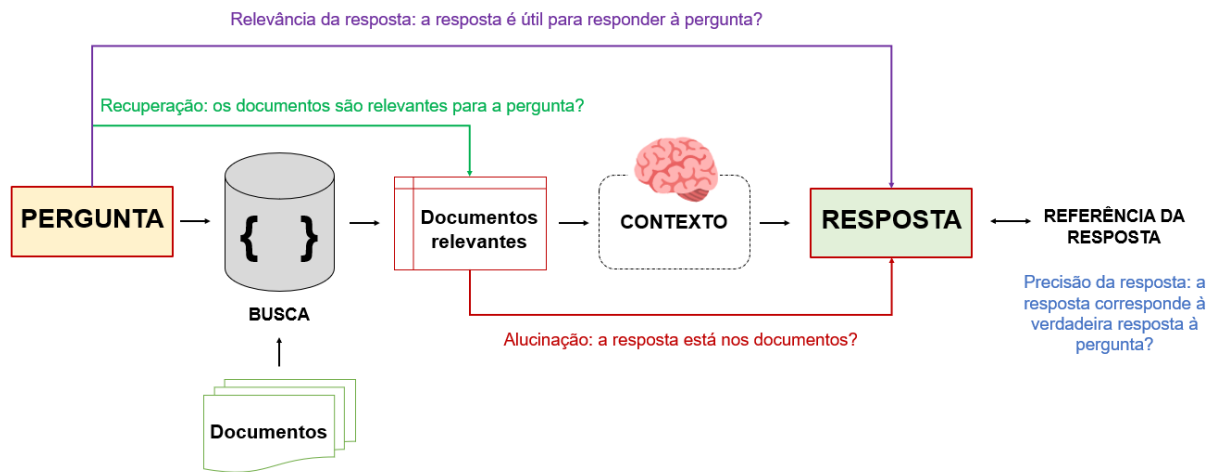
Se o objetivo é avaliar o *reader*, então é necessário saber o quanto a resposta retornada é semelhante a resposta esperada, que é o mesmo objetivo do *pipeline* todo e, por isso, as métricas aplicadas no *reader* podem ser utilizadas também para avaliar o *pipeline* completo. Uma das métricas mais utilizadas é a EM e que mede a proporção de documentos em que a resposta prevista é idêntica à resposta correta. Por outro lado, a métrica F1 é menos punitiva, pois ela mede a sobreposição de palavras entre a resposta esperada e a resposta prevista. Dessa forma, o exemplo anterior resultará em uma validação próxima de 100% (RAJPURKAR et al., 2016).

Apesar do F1 permitir a avaliação do quanto uma resposta é semelhante a outra, isso acontece a nível léxico e, por isso, não resolve o caso onde duas respostas escritas diferentes possuem o mesmo significado, ou seja, semanticamente são similares. Por exemplo, o trecho “cem reais” e o trecho “R\$100,00” não tem nenhuma palavra ou caractere igual, mas semanticamente são idênticos. Para resolver este problema é proposto a utilização de uma métrica chamada *Semantic Answer Similarity* (SAS) que utiliza uma arquitetura *Transformer* estruturada como um *cross-encoder* para avaliar a semântica entre a resposta esperada e a resposta prevista (RISCH et al., 2021).

Com o aumento da complexidade dos sistemas baseados em RAG, cuja arquitetura apresenta diversas dimensões e cada módulo interfere no resultado de outros, faz-se necessário avaliar cada módulo de forma independente, mas também de forma conjunta. Assim, fazem-se necessárias técnicas mais apropriadas como, por exemplo, o *Automated Evaluation of Retrieval Augmented Generation* (RAGAS) (ES et al., 2023), um método para avaliação de RAG que não utiliza referências (figura 13). Ao considerar que sistemas RAG são formados por um módulo de recuperação de informação e um módulo de geração baseado em LLMs, são apresentados quatro métricas, duas para cada módulo, sendo elas:

- Fidelidade (*faithfulness*): métrica que avalia a geração. Essa métrica mede a consistência factual da resposta gerada em relação ao contexto determinado. Ela é calculada a partir da resposta e do contexto recuperado. A resposta é dimensionada para o intervalo (0,1): quanto maior, melhor é o resultado. Quanto mais afirmações realizadas na resposta puderem ser inferidas a partir do contexto dado, maior a fidelidade da resposta.
- Relevância da resposta (*answer relevancy*): métrica que avalia o resultado da geração. Essa métrica se concentra em avaliar quão pertinente é a resposta gerada em relação a *query* fornecido. Respostas incompletas ou que contenham informações redundantes recebem pontuações mais baixas, enquanto pontuações mais altas indicam maior relevância. Essa métrica é calculada utilizando a *query*, a resposta gerada e o contexto, que pode ser o resultado do *retriever* ou simulado para manter consistente.
- Recuperação de contexto (*context recall*): essa métrica é aplicada no módulo de recuperação de informação (*retriever*) e avalia o grau de alinhamento entre o contexto recuperado e a resposta esperada, considerada como a *ground truth*. Essa medida é calculada com base na resposta esperada e no contexto recuperado, variando entre 0 e 1, em que valores mais altos indicam um desempenho melhor. Cada fato presente na resposta esperada é analisado para determinar se pode ser atribuída ao contexto recuperado ou não. Em um cenário ideal, todos os fatos da resposta devem ser atribuíveis ao contexto recuperado.
- Precisão do contexto (*context precision*): métrica aplicada na recuperação de informação (*retriever*) e que avalia se todos os itens relevantes presentes nos contextos fornecido (*ground truth*) estão classificados em posições mais altas ou não. Para cada segmento do contexto obtido, determina se ele é pertinente ou irrelevante para encontrar a resposta correta à pergunta proposta. Idealmente, todos os fragmentos relevantes devem aparecer nas primeiras posições do *retriever*. Essa métrica é calculada utilizando a *query*, o *ground truth* e o contexto, com valores variando entre 0 e 1, onde pontuações mais altas indicam melhor precisão.

Figura 13. Representação das métricas contidas na *Automated Evaluation of Retrieval Augmented (RAGAS)*.



Fonte: adaptado de ES et al. (2023).

Uma das vantagens dessas métricas é que não são necessárias referências, pois são métricas que utilizam a semântica e o entendimento do modelo LLM para julgar a qualidade dos resultados, mostrando mais uma aplicação da integração do modelo LLM no processo de desenvolvimento desse tipo de aplicação.

Além dessas métricas, pode-se citar algumas outras importantes adições a essas, como a relevância contextual (*contextual relevancy*) que é aplicada à informação recuperada (*retriever*) e avalia o quão relevante essas informações são para a dada *query* e a corretude da resposta (*answer correctness*) para a geração da resposta. Essa métrica mede a precisão da resposta gerada em comparação com a resposta esperada. Uma pontuação mais alta (próxima de 1) indica uma maior concordância entre a resposta gerada e a esperada, sinalizando uma melhor corretude. Assim como as anteriores, essas métricas podem utilizar métodos estáticos, como distância cosseno e também um modelo LLM como juiz. Uma das vantagens do uso de LLM nessas avaliações é a capacidade de explicar a nota atribuída a cada amostra, permitindo uma maior clareza nos resultados e ação mais assertiva diante dos resultados (inspirado em métricas apresentadas na ferramenta RAGAS. Disponível em:

<<https://docs.ragas.io/en/stable/concepts/metrics/index.html>>).

Além dessas seis métricas apresentadas, existem outras mais específicas, como, por exemplo, para quantificar inofensividade, nocividade, alucinação,

enviesamento, precisão, capacidade de sumarização, retenção de conhecimento (uma métrica experimental focada na capacidade conversacional) (ferramenta disponível em: <<https://docs.confident-ai.com/docs/metrics-knowledge-retention>>) e outros. Porém, estas não serão aprofundadas nesse trabalho por serem específicas para módulos, de forma que as seis citadas anteriormente são mais focadas nas interações e impactos dos dois principais módulos do RAG, que é o *pipeline* responsável pela interação do agente com a memória e que permite avaliar o impacto do grafo de forma mais direta.

Os *datasets* utilizados para validação de RAG são variados e a escolha depende do objetivo: alguns são focados em respostas curtas, outro longas, multietapas ou, ainda, multidocumentos. Alguns dos *datasets* que podem ser utilizados para avaliação de RAG são:

- *SQuAD (Stanford Question Answering Dataset)*: Um dos *datasets* mais populares para PR em que as respostas são trechos de textos do conjunto de dados fornecido, além de não exigir *multihop* (RAJPURKAR et al., 2016).
- *MS MARCO (Microsoft Machine Reading Comprehension)*: *dataset* popular criado pela *Microsoft* contendo perguntas reais feitas por usuários do *Bing* e que possui respostas baseadas em um único parágrafo (BAJAJ et al., 2016).
- *HotpotQA*: criado especificamente para perguntas *multihop*. Cada pergunta exige a combinação de informações de múltiplos documentos para chegar à resposta correta (YANG et al., 2018).
- *TriviaQA*: contem perguntas triviais extraídas de diferentes fontes. Embora não seja exclusivamente para perguntas *multihop*, contém perguntas baseadas em múltiplos documentos (JOSHI et al., 2017).
- *ComplexWebQuestions*: um *dataset* que é uma extensão do *WebQuestions*, contendo perguntas mais complexas que requerem a combinação de informações de diferentes documentos da *web* (TALMOR; BERANT, 2018).
- *Natural Questions*: desenvolvido pelo *Google* e inclui perguntas reais do *Google Search*. Toda pergunta possui uma versão de resposta curta e longa e pode ser utilizado *multihop* por possuir perguntas que exigem a combinação de informações de múltiplos documentos (KWIATKOWSKI et al., 2019).
- *RAGBench*: Um *dataset* construído especificamente para benchmark de RAG, contendo 100 mil exemplos. Ele abrange cinco domínios específicos de

diferentes indústrias e vários tipos de tarefas de RAG. Exemplos do RAGBench são provenientes de corpora da indústria, como manuais de usuário, tornando-o particularmente relevante para aplicações industriais (FRIEL; BELYI; SANYAL, 2024).

Para avaliar um AIC com RAG, muitos dos *datasets* mencionados anteriormente são relevantes e podem ser utilizados, mas é importante considerar a natureza interativa das conversas. *Datasets* como *HotpotQA*, *TriviaQA* e *Natural Questions* podem ser transformados para o formato de diálogo, criando múltiplos turnos de perguntas e respostas baseados no conteúdo dos documentos. Porém, alguns *datasets* são especialmente adequados para avaliação de AIC como, por exemplo:

- *Question Answering in Context (QuAC)* é um *dataset* projetado para modelar diálogos de perguntas e respostas, onde cada resposta deve levar em conta o contexto das perguntas anteriores. Isso é ideal para avaliar AIC que precisam entender e manter o contexto.
- *Conversational Question Answering (CoQA)* é um *dataset* que possui o formato de diálogo. Ele inclui uma série de perguntas relacionadas a um determinado texto, simulando uma conversa natural.
- *Doc2Dial* é um *dataset* que se concentra em diálogos que são baseados em documentos e em que o agente precisa fornecer respostas consultando documentos específicos, simulando um cenário de atendimento ao cliente, ideal para AI que utilizam RAG.
- *Dialogue Natural Language Inference (DNLI)* tem como foco a inferência no contexto de diálogos naturais, onde o agente precisa inferir respostas corretas baseadas em múltiplos turnos de conversação.

4 GRAFO DE CONHECIMENTO

O grafo de conhecimento (GC) é uma representação estruturada de informações que representa entidades do mundo real, como pessoas, lugares e objetos, e as relações entre elas. Ao contrário de bases de dados vetoriais utilizadas em sistemas de recuperação de conhecimento (LEWIS et al., 2020), os GCs oferecem uma representação simbólica estruturada, conectando conceitos como nós interligados em uma rede. Esses nós representam entidades, enquanto as arestas capturam as conexões entre elas. A flexibilidade dos GCs permite a inclusão de novos dados, rótulos e relações à medida que a base de conhecimento evolui, refletindo melhor a complexidade do mundo real (Kejriwal; Knoblock; Szekely, 2021). Uma das principais vantagens do grafo de conhecimento está em sua capacidade de integrar diversas fontes de dados, permitindo que as máquinas infiram informações que não estão explicitamente declaradas, algo essencial para aplicações como assistentes virtuais, sistemas de recomendação, e motores de busca avançados (HOGAN et al., 2021).

4.1 Grafo de conhecimento para codificação do conhecimento

Apesar das diferentes aparições do termo “grafo de conhecimento” na história e a inexistência de uma definição formal, pode-se considerar um grafo de conhecimento como um grafo de dados destinado a acumular e transmitir conhecimento do mundo real, cujos nós representam entidades de interesse, normalmente objetos do mundo real, como lugares e pessoas e as arestas representam as relações entre essas entidades. Essas relações entre nós podem, ou não, ser direcionais. A palavra “conhecimento” nesse caso se refere a algo que é conhecido, seja advindo de uma fonte externa ou inferido por algum modelo. Dessa forma, os GCs fornecem uma abstração concisa e intuitiva para uma variedade de domínios. Além disso, um GC é dinâmico, ou seja, pode fornecer novas inferências e se remodelar com a adição de novos dados ao longo do tempo e também pode ser heterogêneo, sendo capaz armazenar e relacionar diferentes tipos de informações.

Historicamente, pode-se separar o GC em dois momentos, antes e depois do anúncio do *Google Knowledge Graph* em 2012. Antes desse momento os primeiros

relatos de grafo de conhecimento, relevantes para área, foram na década de 70, utilizados principalmente para estudar comportamento dentro do ensino e modelando dependências de conhecimento entre alunos (SCHNEIDER, 1972; MARCHI; MIGUEL, 1974).

Durante os anos 80 alguns trabalhos aplicaram GC no domínio médico, sociológico, meteorológico (RADA, 1986; BAKKER, 1987; STOKMAN; DE VRIES, 1988). Neste momento algumas outras percepções e definições de grafos foram surgindo, inclusive a primeira noção mais clara de os nós do grafo serem entidades e arestas serem relações (SRIKANT; JARKE, 1989).

Nos anos 90, várias novas definições de GC surgiram, incorporando o conceito de relação direcional, relações com pesos, limitação do grafo não poder ser cíclico, entre outras. Nesse momento, eles foram utilizados também para gerar redes neurais para selecionar hipóteses a partir de observações de entrada (DIENG et al., 1992) e houve a utilização da lógica Bayesiana para construção do GC (SHIMONY; DOMSHLAK; SANTOS, 1997).

O início dos anos 00 foi marcado pela utilização de GC para análise de fluxo de processos e conhecimento (JIANG; MA, 2002). Já no final dos anos 00, foram apresentados grafos de conhecimento para ferramentas de busca (KANESCI et al., 2008; ELBASSUONI et al., 2009) e mapeamento *wikis* (COURSEY; MIHALCEA, 2009). Até 2012, os trabalhos consideravam grafos pequenos quando comparados aos atuais, provavelmente pela limitação computacional da época. Outra grande diferença é o conceito da relação, que definia uma dependência ou causalidade entre nós. Por exemplo, na relação “ $x \rightarrow y$ ”, “ x ” era considerado um pré-requisito para “ y ” ou que “ x ” levava a “ y ”.

Em 2012, foi anunciado o *Google Knowledge Graph* (SINGHAL, 2012) como uma ferramenta capaz de entender o mundo real e mapear a relação entre entidades. Foram apresentadas várias possibilidades de aplicações, gerando assim, um *momentum* nas pesquisas acadêmicas, fortalecendo o uso do termo GC como se utiliza hoje.

A maioria das definições foca em como o GC define a representação semântica da informação ou através de características essenciais. Dentre as definições mais importantes, está a de Paulheim (2016), que define quatro critérios para definir um grafo de conhecimento; Ehrlinger e Woß (2016) que analisaram várias definições existentes e propuseram uma definição que enfatiza o motor de inferência dos grafos

de conhecimento: “um grafo de conhecimento adquire e integra informações em uma ontologia e aplica um motor de inferência para derivar novos conhecimentos” que propuseram uma definição de grafo multirrelacional como sendo: “um grafo de conhecimento é um grafo multirrelacional composto por entidades e relações, que são considerados nós e diferentes tipos de arestas, respectivamente” (WANG et al., 2017).

Um grafo de conhecimento pode ser definido matematicamente como um grafo dirigido rotulado, ou multigrafo, formalizado como um par $G = (V, E)$, onde V é o conjunto de vértices (ou nós), que representam as entidades ou conceitos, $E \subseteq V \times R \times V$ é o conjunto de arestas (ou arcos), onde R é um conjunto de rótulos que descrevem os tipos de relações. Cada aresta $(v_i, r, v_j) \in E$ conecta dois vértices $v_i, v_j \in V$ e é rotulada com uma relação $(r \in R)$, descrevendo a natureza da conexão entre v_i e v_j .

Os vértices V representam entidades como objetos, pessoas ou conceitos, e podem ter atributos ou propriedades associadas que qualificam cada entidade. As arestas E representam as relações dirigidas entre os vértices. Cada aresta é uma tripla (v_i, r, v_j) , onde v_i é a entidade de origem (sujeito), v_j é a entidade de destino (objeto), e $r \in R$ é o rótulo que especifica a relação entre v_i e v_j (predicado). Os rótulos R formam o conjunto de tipos de relações que conectam as entidades, podendo incluir relações como "é amigo de", "escreveu" ou "parte de", entre outras.

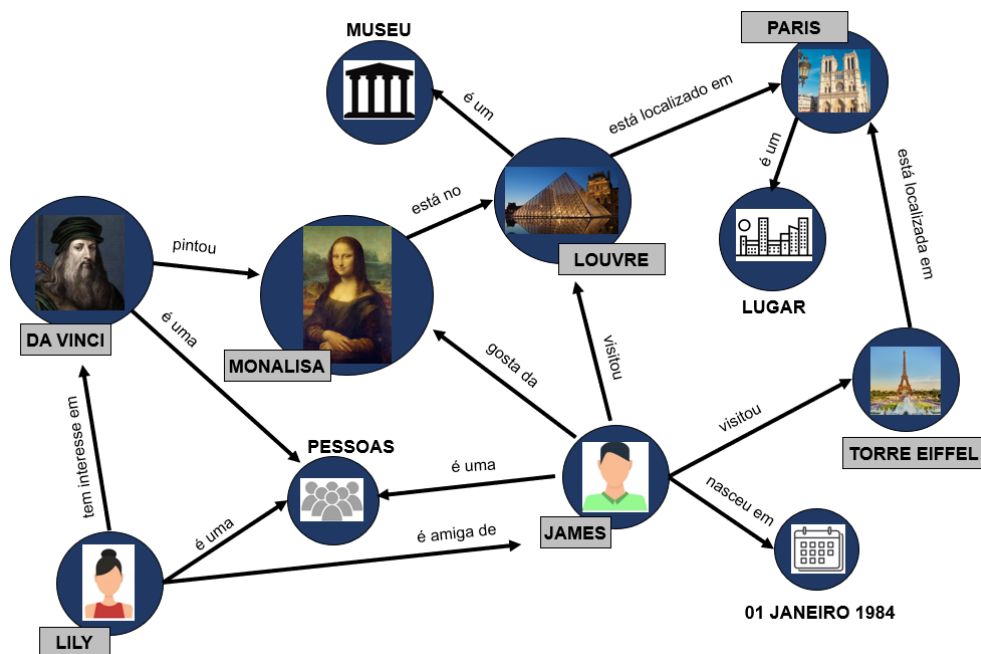
Além disso, grafos de conhecimento frequentemente incluem ontologias ou esquemas que especificam regras e restrições sobre como as entidades e relações podem ser organizadas, garantindo a semântica do grafo. Assim, matematicamente, pode-se considerar um grafo de conhecimento como uma extensão de um grafo orientado, onde as arestas são rotuladas, as entidades (nós) contêm propriedades e metadados adicionais, e as relações são potencialmente complexas e multilaterais.

Enquanto de um lado tem-se esse histórico de definições acadêmicas, do outro lado as grandes empresas tentavam criar uma definição mais comercial para grafo de conhecimento e, junto ao *Google, IBM, Facebook, eBay* e *Microsoft* (NOY et al., 2019) publicaram uma definição, que se tornou consenso entre essas empresas, que diz que um GC descreve objetos de interesse e conexões entre eles. Muitas implementações práticas impõem restrições aos links em GC definindo um esquema ou ontologia. GC e estruturas semelhantes geralmente fornecem um substrato compartilhado de conhecimento dentro de uma organização, permitindo que diferentes produtos e aplicativos utilizem vocabulário semelhante e reutilizem definições e descrições

criadas por outros. Além disso, eles geralmente fornecem uma representação formal compacta que os desenvolvedores podem usar para inferir novos fatos e construir o conhecimento.

Atualmente, podem ser encontrados GC *OpenSource* construídos utilizando bases de conhecimento de domínio gera como, por exemplo, *DBpedia* (LEHMANN et al., 2015), *Freebase* (BOLLACKER et al., 2007), *Wikidata*, *YAGO* (VRANDEČIĆ; KRÖTZSCH, 2014), *KBpedia* (FRANCIS et al., 2018), que são focados em fatos e *CommonsenseQA* (TALMOR et al., 2018) e *OpenBookQA* (MIHAYLOV et al., 2018) que são focados em senso comum. Em alguns casos existem GC de domínio específico construído em bases de conhecimento aberta como, por exemplo, mídia (RAIMOND et al., 2014), geografia (STADLER et al., 2012), turismo (MATURANA et al., 2018) e ciência (CALLAHAN et al., 2013). Porém, muitas empresas disponibilizam GC em versões *enterprise*, em forma de buscadores (*Google, Bing*), *e-commerces* (*Amazon, Airbnb, Ebay*), redes sociais (*Linkedin, Facebook, Instagram, Pinterest*) e outros. A figura 14 ilustra um exemplo de grafo de conhecimento.

Figura 14. Exemplo de grafo de conhecimento.



Legenda: o diagrama apresenta as relações (arestas) entre várias entidades (nós), como pessoas, lugares e objetos.

Fonte: adaptado de: <<https://aws.amazon.com/neptune/knowledge-graphs-on-aws/>>. Acesso em: 13 junho. 2024.

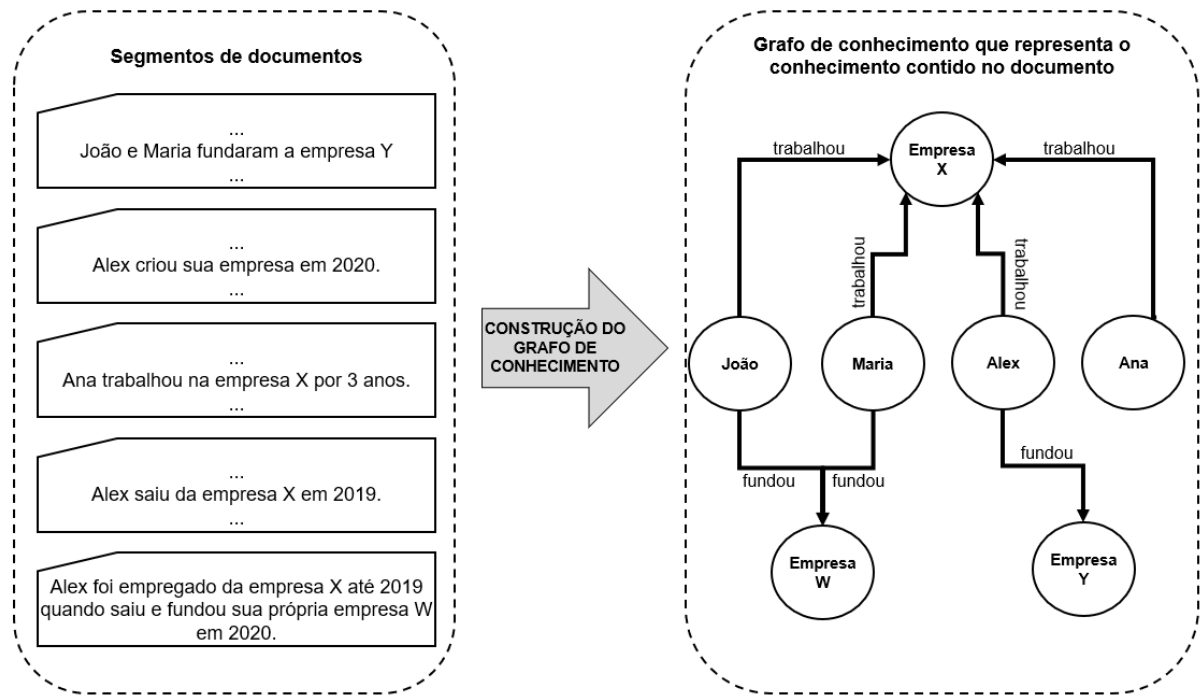
A vantagem fundamental de representar dados em formato de grafo é que eles modelam o mundo como objetos que têm propriedades e relacionamentos com outros

objetos, fornecendo o contexto para os dados através dessa correlação entre todos os dados. Dessa forma, o formato do grafo é considerado o mais próximo da forma como os seres humanos percebem o mundo quando comparados a técnicas que representam o conhecimento em vetores numéricos. Isso possibilita uma redução no custo da curadoria de banco de dados, assim como possibilita o avanço na direção ao entendimento das decisões dos modelos de AM.

Se comparado à estrutura textual, a estrutura do grafo proporciona uma visão mais completa das conexões entre os documentos, ao invés da visão localizada e isolada, como pode ser observado na figura 15. As dependências entre entidades são diretamente capturadas pelas arestas rotuladas com o tipo de relação. Além disso, as buscas podem ser realizadas através de diversas conexões, alcançando conjuntos de documentos antes improváveis com busca de texto ou *embedding*. Nós que possuem alta importância na base de conhecimento pode ser utilizados como ponto de entrada para subgrafos que possuem conhecimentos específicos, reduzindo a necessidade de busca completa e tornando a busca mais escalável. Os dados estruturados em grafos, em sua maioria, não necessitam de uma estrutura hierárquica (como JSON e XML) e, por isso, podem facilmente modelar relações cíclicas. O fato de a estrutura ser um grafo possibilita a aplicação de várias técnicas de processamento em grafos desenvolvidas durante décadas de estudo dessa estrutura, principalmente análises estatísticas que por si só já geram valor como, por exemplo, análise de melhor caminho, análise de nó mais influente, grau de conectividade de um nó, *PageRank* e detecção de comunidades (BRIN; PAGE, 1998).

Essas análises podem ser realizadas utilizando as linguagens para consulta em grafos que, ao contrário de outros modelos *NoSQL*, suportam não apenas operadores relacionais padrão (junções, uniões, projeções e outros), mas também operadores de navegação. Além disso, é possível aplicar ontologias (HITZLER et al., 2012; MUNGAL et al., 2012) e regras (HORROCKS et al., 2004; KIFER; BOLEY, 2013) para definir e descobrir novas relações dentro do grafo, consequentemente permitindo a descoberta de conhecimento. Entre as várias linguagens de consulta em GC estão *SPARQL* para gráficos *Resource Description Framework* (RDF) (HARRIS; SEABORNE; PRUD'OMMEAUX, 2013), *Cypher* (FRANCIS et al., 2018), *Gremlin* (RODRIGUEZ, 2015) e *G-CORE* (ANGLES et al., 2017) para consultar *property graphs*.

Figura 15. Representação de um exemplo de um documento segmentado e transformado em um grafo de conhecimento.



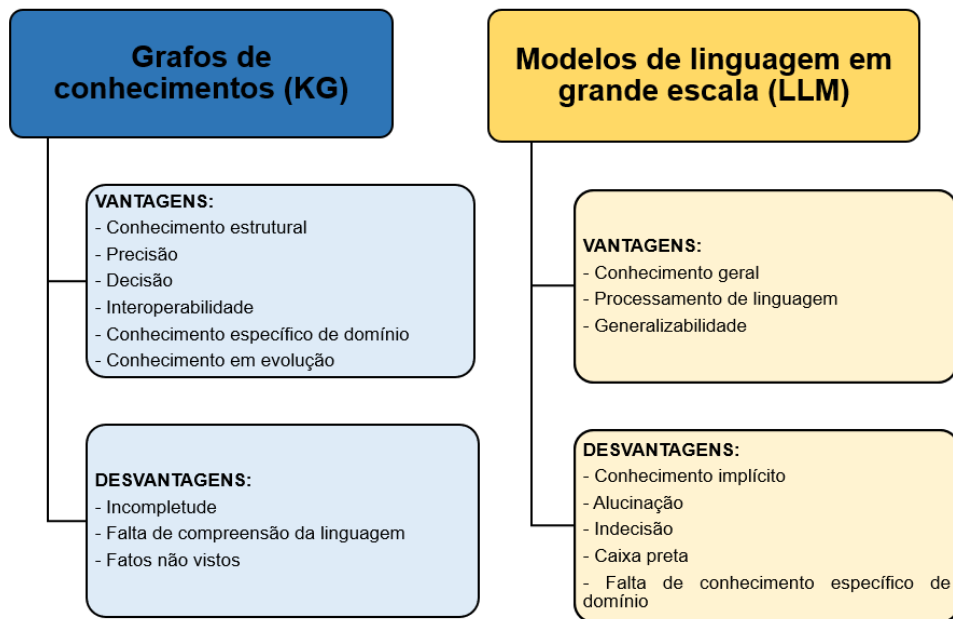
Fonte: próprio autor.

No GC fatos explícitos são diretamente capturados como nós e arestas, preservando detalhes importantes em vez de serem condensados em vetores. As entidades contêm atributos ricos, como descrições, *labels* e metadados que fornecem contexto crucial. A estrutura do grafo modela conexões do mundo real entre entidades, capturando regras, hierarquias e cronologias. Consultas podem percorrer relacionamentos para conectar fatos de fontes diversas, permitindo respostas que requerem inferência em múltiplas etapas. A resolução de entidades vincula referências ao mesmo objeto do mundo real, facilitando uma análise coletiva. A topologia do grafo fornece transparência sobre a relevância de certos fatos com base em suas conexões. Além disso, atributos do usuário, contexto e interações históricas são capturados para ajustar os resultados, possibilitando personalização. Esses componentes são fundamentais para sistemas que buscam fornecer respostas precisas e contextualmente relevantes, permitindo uma melhor compreensão e utilização das informações. Ao preservar detalhes explícitos e fornecer uma estrutura transparente para o raciocínio, é possível construir soluções mais robustas e

adaptadas às necessidades individuais dos usuários. Além disso, a flexibilidade do grafo permite que ele cresça com o tempo, representando a evolução do conhecimento do negócio (HOGAN et al., 2021).

Os GC oferecem vantagens significativas, e têm servido por muito tempo como base de numerosos produtos e aplicações intensivas em conhecimento (YANG et al., 2023; KERTKEIDKACHORN et al., 2023; XU et al., 2019), mas também apresentam limitações como, por exemplo, a complexidade na implementação e manutenção, desafios de escalabilidade, dependência da qualidade dos dados e dificuldade em resolver ambiguidades e inconsistências. As técnicas tradicionais para a construção de um GC geralmente envolvem o uso de diversos modelos especializados, o que requer mão de obra qualificada e, conseqüentemente, resulta em altos custos. Por outro lado, os modelos de linguagem de grande escala (LLMs) possuem uma alta capacidade de generalização, mas enfrentam dificuldades no acesso à informação. Nesse contexto, os GCs e LLMs podem apresentar melhores resultados se utilizados de forma complementar: os GCs aprimoram os LLMs ao fornecer conhecimento externo, interoperabilidade e representação de conhecimento factual e explícito, enquanto os LLMs podem ser utilizados para gerar novos fatos e representar conhecimento implícito (PAN et al., 2024). A Figura 16 ilustra as vantagens e desvantagens de cada uma dessas técnicas.

Figura 16. Comparação entre as vantagens e desvantagens dos grafos de conhecimento (KG) e dos modelos de linguagem em grande escala (LLM).



Fonte: adaptado de Pan et al. (2024).

4.2 Construção do grafo de conhecimento

Nessa sessão será aprofundado os conhecimentos sobre as técnicas e métodos necessários para a construção de um GC. Os componentes principais dessa construção podem ser resumidos em: pré-processamento, extração de entidade, vinculação de entidade ou desambiguação de entidades, extração de relação e montagem do grafo.

Para entendimento deste processo, são importantes algumas definições sobre grafos e sobre como é possível abstrair dados através deles. A primeira decisão ao se construir um grafo é qual o modelo de representação será utilizado, sendo as mais empregadas:

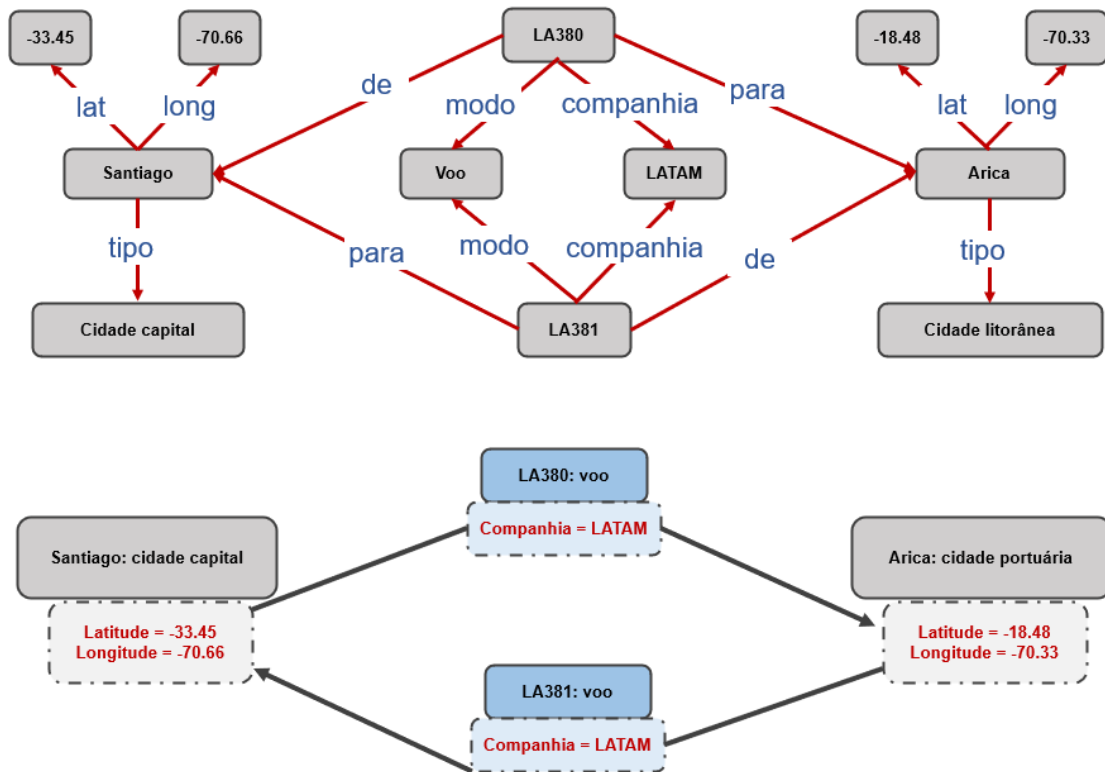
- *Directed edge-labelled graphs:* esse modelo descreve um grafo como sendo um conjunto de nós e um conjunto de arestas entre os nós. Quando o grafo representa um GC, os nós são entidades e as arestas a relação entre os nós. A principal ferramenta que utiliza esse modelo é o RDF (WISEMAN; RUSH; SHIEBER, 2016), recomendado pela W3C. O modelo RDF define diferentes tipos de nós, incluindo identificadores de *Internationalized Resource Identifiers* (IRIs) (BRICKLEY; GUHA, 2014) que permitem a identificação global de

entidades na web. Isto permite a representação de *strings* e outros tipos de variáveis (inteiros, datas e outros), e nós em branco que são anônimos aos quais não foi atribuído um identificador.

- *Property graphs*: esse modelo permite que um conjunto de pares de propriedade-valor e um rótulo sejam associados a nós e arestas. O grafo de propriedade foi proposto para oferecer flexibilidade adicional ao modelar dados como um grafo (ABADI et al., 2015). As ferramentas de banco de dados baseadas em grafos mais populares utilizam esse modelo como, por exemplo, o *Neo4j*. Uma vantagem é que esse modelo permite a tradução para *directed edge-labelled graphs* sem perda de informação. Dessa forma, o *directed edge-labelled graphs* apresenta um modelo mínimo para o grafo, enquanto os *property graphs* oferecem um modelo mais flexível.
- *Web Ontology Language (WOL)*: utilizado para descrever o modelo de dados e proporciona grande poder em termos de recursos de modelagem. No entanto, sua utilização deve ser cautelosa, pois pode resultar em um modelo de dados excessivamente complexo e de difícil compreensão.

A representação visual da comparação entre grafo direcional e grafo de propriedade está contida na figura 17.

Figura 17. Comparação entre grafo direcional (RDF) e grafo de propriedade.



Legenda: o primeiro grafo representa um grafo direcional, enquanto o segundo apresenta um grafo de propriedades, ambos os grafos representam o mesmo conteúdo.

Fonte: adaptado de Hogan et al. (2021).

A construção do grafo depende, também, de uma fonte de informação que será utilizada para extração e definição dos nós, arestas e atributos do grafo. Esse dado de origem pode estar em vários formatos como, por exemplo, texto corrido (livros, jornais, e-mails e outros), texto com marcações (HTML, *Wikitext*), texto estruturado (JSON, XML), regras (ontologias) ou até mesmo conhecimento humano direto (quando especialistas definem as entidades e relações principais do domínio). Porém, esses formatos podem ser combinados para obtenção de um grafo mais estruturado e rico, por exemplo, a utilização de ontologias para guiar a extração de triplas, e a utilização de wikis para enriquecimento da informação.

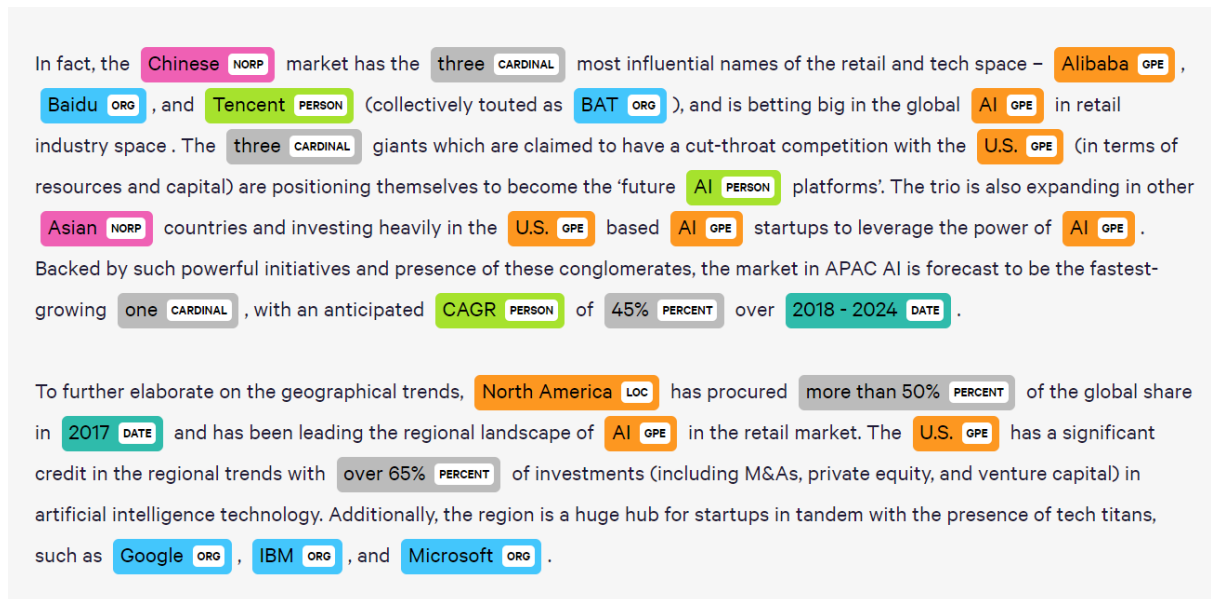
Com o advento das LLMs e sua capacidade de compreender texto em diversos estilos e formatos, esta se tornou uma ferramenta essencial para democratização e viabilidade da construção de GC, isso se deve a simplificação do pipeline de extração de entidades e relações. Para compreender esses avanços, e o caminho trilhado até o uso da LLM nessas tarefas, será apresentado a seguir as principais técnicas e métodos de extração de entidades e relações.

4.2.1 Extração de Entidades

A aquisição de informação baseada em entidades pode ser dividida em diversas subtarefas, como reconhecimento de entidades, desambiguação de entidades, classificação de entidades e alinhamento de entidades. A entidade é um termo que identifica claramente um item pertencente a um conjunto de itens com atributos semelhantes que, normalmente, variam sua definição de acordo com o domínio e contexto que elas se encontram. Exemplos de entidades são: nomes de organizações, pessoas e locais no domínio geral; nomes de genes, proteínas, drogas e doenças no domínio biomédico (NADEAU; SEKINE, 2007).

A extração de entidades nomeadas (NER do inglês, *Named Entity Recognition*), é a tarefa de identificar e classificar, ou tipificar entidades específicas em um dado texto. A tipificação inclui atribuir a uma entidade tipos gerais e específicos, sendo que esses últimos usam uma categoria de tipo estruturada em árvore e são tipicamente considerados como classificação multi-classe e multi-rótulo. O processo normalmente é realizado por modelo que recebem um texto como entrada e tem como saída uma lista de tuplas na forma “INÍCIO, FIM, CLASSE”, em que “INÍCIO” é o índice da *string* de entrada em que inicia a entidade nomeada, “FIM” é o índice da *string* de entrada em que termina a entidade nomeada e “CLASSE” é a classe ao qual essa *substring* pertence e são normalmente predefinidas (LI et al., 2020). Um exemplo de inferência na tarefa de NER pode ser observado na figura 18.

Figura 18. Exemplo de inferência na tarefa de *Named Entity Recognition* (NER).



Legenda: As entidades estão marcadas e cada classe está descrita na frente com uma cor diferente. Fonte: retirado *ipsis literis* de: <<https://meenavyas.wordpress.com/2018/06/10/spacy-named-entity-and-dependency-parsing-visualizers/>>. Acesso em: 05 jun. 2024.

O NER atua como uma etapa de pré-processamento importante para uma variedade de aplicativos, como recuperação de informações, resposta a perguntas, pesquisa semântica, tradução automática e outros.

O resultado do modelo de NER depende fortemente da qualidade do *dataset* de treino e validação, por isso é importante que a anotação dessas entidades seja realizada por especialistas do domínio desejado e que exista adequada curadoria para garantir a qualidade. Porém, por ser uma tarefa bastante consolidada, existem diversos *datasets*. O quadro 2 apresenta os principais *datasets* de NER até o ano de 2020 (LI et al., 2020).

Quadro 2. *Datasets de Named Entity Recognition (NER) existentes na língua inglesa até o ano de 2020.*

Corpus	Ano	Fonte de dados	#tags	URL
MUC-6	1995	Wall street journal	7	[https://catalog ldc.upenn.edu/LDC2003T13]
MUC-6 plus	1996	Adição de novas notícias ao MUC-6	7	[https://catalog ldc.upenn.edu/LDC96T10]
MUC-7	1997	New York Times news	7	[https://catalog ldc.upenn.edu/LDC2001T02]
CoNLL03	2003	Notícias	4	[http://www.clips.uantwerpen.be/conll2003/ner/]
ACE	2000-2008	Notícias, transcrições	7	[https://www ldc.upenn.edu/collaborations/past-projects/ace]
OntoNotes	2007-2012	Revistas, notícias, internet	18	[https://catalog ldc.upenn.edu/LDC2013T19]
W-NUT	2015-2018	Textos gerados por usuários	6/10	https://noisy-text.github.io
BBN	2005	Wall street jornal	64	[https://catalog ldc.upenn.edu/LDC2005T33]
WikiGold	2009	Wikipedia	4	[https://figshare.com/articles/Learning_multilingual_named_entity_recognition_from_Wikipedia/5462050]
WiNER	2012	Wikipedia	112	[https://rali.iro.umontreal.ca/rali/en/winer-wikipedia-for-ner]
WikiFiger	2012	Wikipedia	87	[https://github.com/xiaoling/figer]
HYENA	2012	Wikipedia	505	[https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/hyena]
N3	2014	Notícias	3	[https://aksw.org/Projects/N3NERDIEF.html]
Gillick	2016	Revistas, notícias, internet	5	[https://arxiv.org/e-print/1412.1820v2]
FG-NER	2018	Vários	200	[https://fgner.ait]
NNE	2018	Notícias	36	[https://github.com/nickyringland/nested_named_entities]
GENIA	2004	Textos clínicos e de biologia	34	[http://www.geniaproject.org/home]
GENETAG	2003	MEDLINE	29	[https://sourceforge.net/projects/jbio/files/]
FSU-PRGE	2017	PubMed e MEDLINE	3	[https://julielab.de/Resources/FSU_PRGE.html]
NCBI-DISEASE	2014	PubMed	3	[https://www.ncbi.nlm.nih.gov/CBBresearch/Dogan/DISEASE/]
BC5CDR	2015	PubMed	3	[http://bioc.sourceforge.net/]
DFKI	2018	Notícias de negócios e mídia social	7	[https://dfki-it-re-group.bitbucket.io/product-corpus/]

Fonte: adaptado de Li et al. (2020).

Diferentes formas de anotação são utilizadas na construção destes *datasets*. Esses padrões de anotação para NER se diferenciam nas *tags* de identificação de onde se inicia e onde termina uma entidade. Um documento anotado para a tarefa de NER normalmente consiste em uma palavra por linha com a anotação por palavra. Alguns exemplos de padrões de anotação são IOB (*Inside, Outside, Being*), IOBES

(*Inside, Outside, Begin, End, Single token*), BIOL (*Beginning, Inside, Outside, Unit, Last*). O quadro 3 apresenta um exemplo ilustrando a diferença entre IOB2 e IOBES.

Quadro 3. Exemplo de anotação para os padrões IOB2 e IOBES.

Palavra	IOB2	IOBES
Nasceu	O	O
Nova	B-LOCAL	B-LOCAL
lorque	I-LOCAL	E-LOCAL
,	O	O
nos	O	O
EUA	B-LOCAL	S-LOCAL
,	O	O
a	O	O
11	B-TEMPO	B-TEMPO
de	I-TEMPO	I-TEMPO
Janeiro	I-TEMPO	I-TEMPO
de	I-TEMPO	I-TEMPO
1842	I-TEMPO	I-TEMPO
.	O	O

Fonte: próprio autor

Em decorrência do amplo uso acadêmico e industrial, existem várias ferramentas com modelos de NER pré-treinados disponíveis *online*, como pode ser observado no quadro 4. Normalmente, esses modelos são treinados em *datasets* genéricos, baseados em *wiki* (como o *wikidata*) e possuem o objetivo de se adaptarem a aplicações de propósitos gerais.

Quadro 4. Ferramentas acadêmicas e industriais com modelos de *Named entity recognition* (NER) pré-treinados.

Sistema NER	URL
StanfordCoreNLP	https://stanfordnlp.github.io/CoreNLP/
OSU Twitter NLP	https://github.com/aritter/twitter_nlp
Illinois NLP	http://cogcomp.org/page/software/
NeuroNER	http://neuroner.com/
Nersuite	http://nersuite.nlpplab.org/
Poluglot	https://polyglot.readthedocs.io
Gimli	http://bioinformatics.ua.pt/gimli
spaCy	https://spacy.io/api/entityrecognizer
NLTK	https://www.nltk.org
OpenNLP	https://opennlp.apache.org/
LingPipe	http://alias-i.com/lingpipe-3.9.3/
AllenNLP	https://demo.allennlp.org/
IBM Watson	https://natural-language-understanding-demo.ng.bluemix.net
FG-NER	https://fgner.alt.ai/extractor/
Intellexer	http://demo.intellexer.com/
Repustate	https://repustate.com/named-entity-recognition-api-demo
Dandelion API	https://dandelion.eu/semantic-text/entity-extraction-demo
displayCy	https://explosion.ai/demos/display-ent
ParallelDots	https://www.paralleldots.cm/named-entity-recognition
TextRazor	https://www.textrazor.com/named_entity_recognition

Fonte: adaptado de Li et al. (2020).

Normalmente, a avaliação de NER é realizada pela comparação do resultado do modelo com a anotação feita por seres humanos. Além disso, o NER pode ser avaliado como duas subtarefas: detecção dos limites que definem uma entidade (INICIO e FIM) e a classe da entidade nomeada (CLASSE). As duas métricas mais comuns de avaliação de NER são:

- *Exact-match Evaluation*: essa métrica exige que tanto o INÍCIO, FIM e CLASSE sejam exatamente iguais ao anotado (SANG; DE MEULDER, 2003) Dessa forma, são utilizados como métricas *Precision*, *Recall* e *F-score*.
- *Relaxed-match Evaluation*: diferentes critérios podem ser utilizados para esse tipo de métrica. Por exemplo, no MUC-6 (YASUNAGA, 2021) um acerto é creditado sempre que uma entidade recebe a CLASSE correta, independentemente de seus limites, desde que haja uma sobreposição com os limites da verdade básica, independentemente da atribuição de tipo de uma entidade. Enquanto isso, o ACE (DODDINGTON et al., 2004) considera *matches* parciais e subtipos de entidades. Porém, essa falta de padronização na avaliação se torna problemática devido à dificuldade de comparação entre os estudos.

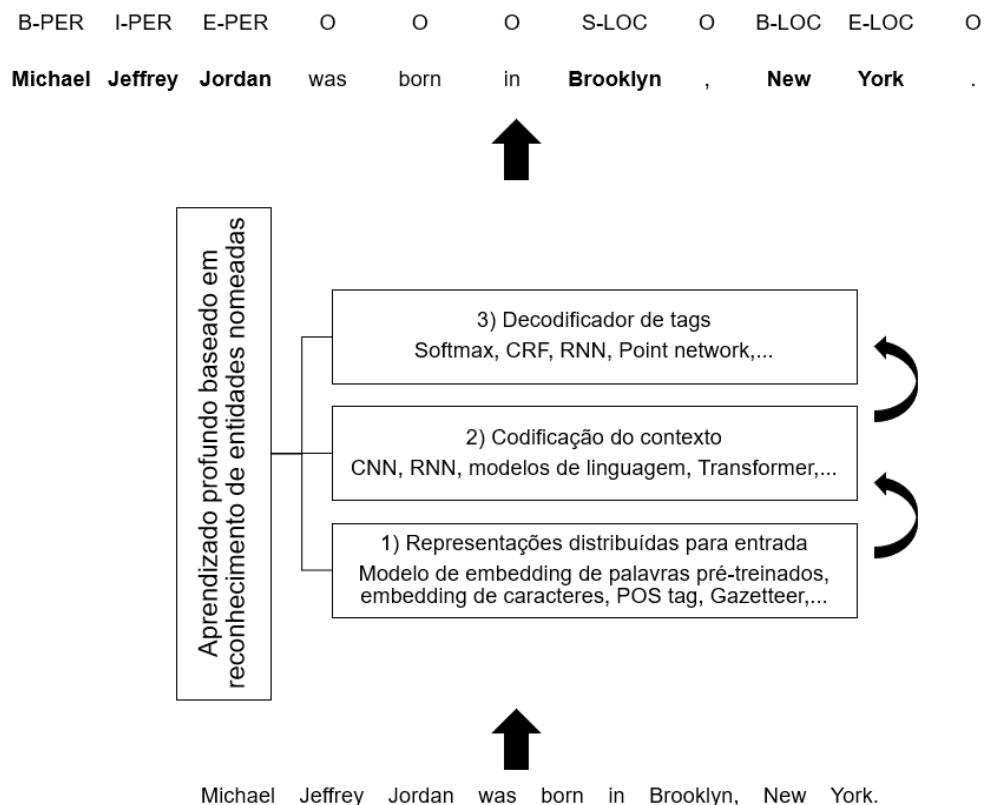
Por ser uma tarefa clássica de PLN, existem diversas abordagens em razão do avanço das técnicas, iniciando-se nas técnicas baseadas em regras até complexas RN profundas. A abordagem mais comum é a baseadas em regras, em que regras sintáticas e léxicas pré-definidas (HUDSON; MANNING, 2018) são utilizadas para identificar entidades. Para isso, pode-se utilizar dicionários de sinônimos para identificar menções de entidades (ZELLERS et al., 2018, 2019). Estes métodos são limitados em razão do grande número de regras necessárias para cobrir todas as variações de entidades, além de não considerar o contexto. Porém, pode funcionar muito bem em domínios muito específicos e com textos em formatos mais controlados. Uma combinação de regras e técnicas não supervisionadas pode ser utilizada para encontrar entidades através de clusterização (DEVLIN et al., 2019).

A abordagem mais utilizada atualmente é a utilização de modelos treinados via aprendizado supervisionado. Uma das principais necessidades dessa abordagem são as *features*, normalmente representadas por um vetor que é utilizado na entrada dos modelos. Essas *features* são uma abstração do texto e podem conter diferentes características em diferentes níveis (palavra, frase, documento). Estas podem ser extraídas por algum processo de engenharia de *feature* em que o vetor de *features* contém, por exemplo, resultado de *part-of-speech tag*, morfologia das palavras (se a palavra se inicia com letra maiúscula) e representação de posição relativa das palavras. Além desses, o vetor de *feature* pode ser gerado por outros métodos como, por exemplo, CBOW e *continuous skip-gram* (MIKOLOV et al., 2013b) ou métodos mais modernos como *Word2Vec* (MIKOLOV et al., 2013b), *GloVe* (PENNINGTON; SOCHER; MANNING, 2014), *fastText* (BOJANOWSKI et al., 2017) e ainda mais recentemente, os modelos de linguagem, como *BERT*. Outra forma de tratamento de NER é como um problema de classificação em que um modelo supervisionado é treinado para classificar se uma palavra ou conjunto de palavras em uma frase é uma entidade e qual entidade seria. Para isso, podem ser utilizados diversos modelos como, por exemplo, *Hidden Markov Models* (HMM) (RABINER; JUANG, 1986), *Decision Trees* (QUINLAN, 1986), *Maximum Entropy Models* (RATNAPARKHI, 1997), *Support Vector Machines* (SVM) (HEARST et al., 1998) e *Conditional Random Fields* (CRF) (LAFFERTY; MCCALLUM; PEREIRA, 2001).

O estado da arte para o problema de NER são modelos de AP que utilizam ML e RN profundas para localização de uma entidade e a qual classe a entidade pertence. As diversas camadas de abstração desses métodos, normalmente camadas de RNA,

permitem a descoberta de relações semânticas mais complexas entre as palavras de uma sentença, gerando representações latentes mais eficientes para o modelo de classificação (YADAV; BETHARD, 2018). Outra vantagem do DL é a capacidade de realizar treinamentos multitarefas, sendo esperado resultados melhores que modelos treinados em apenas uma tarefa (BIZER; HEATH; BERNERS-LEE, 2011), pois ao treinar duas tarefas ao mesmo tempo o modelo consegue identificar relações e representações internas úteis entre elas. Dessa forma, é possível treinar um modelo de regressão que tenha como saída os índices de início e fim de uma entidade e que também seja capaz de classificar as entidades encontradas, resultando em melhores resultados. A representação de como isso ocorre está contida na figura 19.

Figura 19. Exemplo de *pipeline* de *Named Entity Recognition* (NER) baseado em aprendizado profundo (*deep learning*).

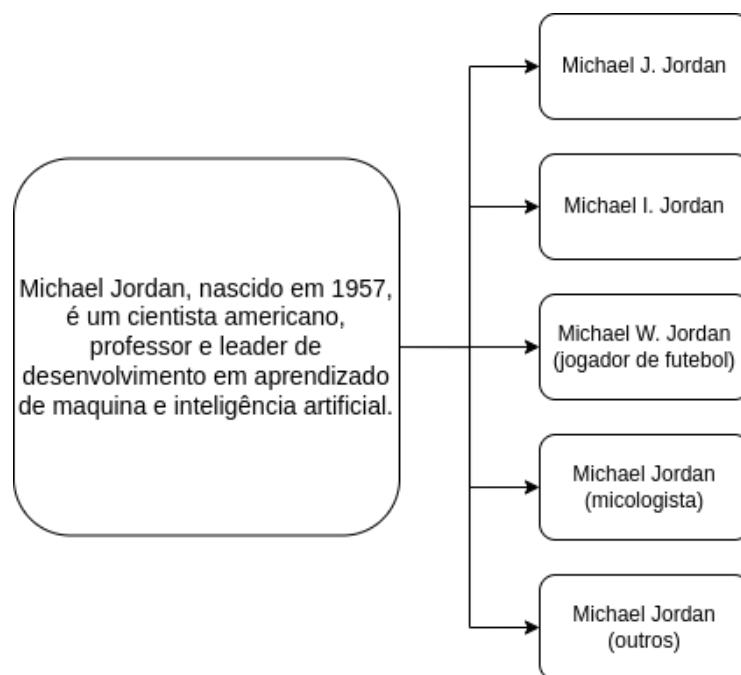


Legenda: como pode ser observado, uma frase é dada como entrada para o *pipeline* e então é realizada a geração do vetor de características (*embedding*). Este é passado para um modelo de codificação do contexto, responsável pelo entendimento semântico do *input*, e finalmente um outro modelo utiliza esse entendimento para realizar a tarefa de identificar e classificar as entidades na frase.

Fonte: próprio autor.

Um desafio na extração de entidades nomeadas é presença de ambiguidade léxica entre elas. Essa ambiguidade léxica pode ser observada em um exemplo na figura 20. Nesse caso, em uma aplicação que é desejável apenas recuperar todas as menções de “Michael Jordan” (jogador de basquete) e excluir menções de outras pessoas com o mesmo nome, como “Michael Jordan” (matemático), é necessário que ocorra a desambiguação ou associação de entidades.

Figura 20. Exemplo de ambiguidade léxica e associação de entidades.



Fonte: próprio autor.

A tarefa de associação de entidades, do inglês *Entity Linking* (EL), também conhecida como desambiguação de entidades, é o processo referente ao encontro de menções correspondentes de uma entidade em um contexto específico. No exemplo da figura acima, essa tarefa se daria por meio da descoberta de se o “Michael Jordan” que aparece em um dado texto é o jogador de basquete ou o matemático. Em outras palavras, é o processo de identificar a menção de uma entidade em um texto (não estruturado) e estabelecer uma ligação em uma base de conhecimento, comumente estruturada como, por exemplo, um grafo, um banco de dados ou um índice de uma *wiki*. Essa tarefa é considerada chave para solucionar o problema de ambiguidade entre as entidades nomeadas e permitir que a real informação seja extraída de um

texto. Por isso, é considerada essencial em qualquer tarefa de PLN que se faça necessário desambiguar entidades nomeadas e determinar o real significado delas no contexto, como em sistemas de extração de informação, sistemas de pergunta e respostas e outros (SHEN; WANG; HAN, 2015). Comumente, aplicações de domínio aberto utilizam uma base de conhecimento derivadas do *wikipedia*, sendo que as duas maiores e mais utilizadas são a *Wikidata* ou *DBpedia*. As técnicas de EL que mapeiam entidades nomeadas para entidades da *Wikipédia* também são chamadas de wikificação. A figura 21 representa como pode ser realizada a tarefa de desambiguação da entidade “pessoa” proposta.

Figura 21. Exemplo de desambiguação da entidade “pessoa” através do *link* com uma página *wiki*.

We know ‘Sebastian Thrun’ is a person
but do we know which person exactly?

When **Sebastian Thrun** **PERSON** started at **Google** **ORG** in **2007** **DATE**, few people outside of the company took him seriously. “I can tell you very senior CEOs of major **American** **WORD** car companies would shake my hand and turn away because I wasn’t worth talking to,” said **Thrun** **PERSON**, now the co-founder and CEO of online higher education startup Udacity, in an interview with **Recode** **ORG** **earlier this week** **DATE**.

A little **less than a decade later** **DATE**, dozens of self-driving startups have cropped up while automakers around the world clamor, wallet in hand, to secure their place in the fast-moving world of fully automated transportation.

About: **Sebastian Thrun**
An entity of type: **person**, from named graph: http://dbpedia.org/wiki/Sebastian_Thrun

Sebastian Thrun (born May 14, 1967) is an innovator, entrepreneur, educator, and computer scientist from Germany. He was CEO and cofounder of Udacity. Before that, he was a Google VP and Fellow, and a Professor of Computer Science at Stanford University. At Google, he founded Google X. He is currently also an Adjunct Professor at Stanford University and at Georgia Tech.

Property: **dbpedia:wikiPage** Value: http://dbpedia.org/page/Sebastian_Thrun

Legenda: A entidade “pessoa” com valor “Sebastian Thrun” não terá ambiguidade com outra entidade “pessoa” com o mesmo valor pois será realizado o processo de desambiguação através do *link* com uma página na *wiki* que define quem é essa entidade no contexto que ela foi encontrada.

Fonte: retirado *ipsis literis* de: <<https://www.aitimejournal.com/complete-tutorial-on-named-entity-recognition-ner-using-python-and-keras/7728/>>. Acesso em: 20 ago. 2024.

Alguns trabalhos consideram a tarefa de NER como uma etapa de EL, outros consideram apenas a desambiguação como sendo a tarefa de EL. Arquiteturas mais generalistas de EL, baseadas em métodos neurais, podem ser expressas em quatro etapas: geração de candidatos, codificação de menção de contexto, codificação de entidade e predição de menções prováveis. Apesar da literatura apresentar diversas abordagens para a arquitetura, todas elas podem ser consideradas alterações em uma ou mais dessas etapas (SEVGILI et al., 2020).

Porém, esse processo pode ser custoso e pouco escalável, impedindo o uso dessas abordagens neurais específicas. Com o advento das LLMs e sua capacidade

de compreender semântica, essa se tornou uma alternativa viável para desambiguação, onde o modelo atribui classes diferentes para a mesma palavra dependendo do contexto que ela se encontra. Ao combinar essa abordagem com a de wikificação, é possível obter um resultado satisfatório para o problema, com uma considerável simplificação do processo.

4.2.2 Extração de Relações

A extração de relação (ER) é uma etapa importante na construção de bases de conhecimento a partir de um grande número de textos não estruturados (TRISEDYA, 2019) e pode ser definida como a tarefa de extrair relações semânticas do texto que, geralmente, ocorrem entre duas ou mais entidades. Essas relações podem ser de diferentes tipos, por exemplo, na sentença “Paris está na França” pode-se entender como um relacionamento o trecho “está em” entre as entidades “Paris” e “França”. Isso pode ser denotado por meio de triplas, (Paris, está em, França) → (entidade, relação, entidade). Essa tripla pode ser ou não direcional.

Os métodos para extração de relação (ER) podem ser divididos em:

- *Rule-based*: em que as relações são extraídas com base em regras criadas manualmente. Normalmente é aplicado *part-of-speech* para extrair a estrutura gramatical da sentença e em seguida é aplicado um *regex* para extrair a relação. Apesar da alta da alta precisão dos padrões, a quantidade de possíveis variações linguísticas impossibilita a aplicação desses métodos em contextos gerais.
- *Weakly Supervised*: esse método se inicia com um conjunto de regras criadas manualmente e encontra automaticamente novas a partir dos dados de texto não rotulados por meio de um processo iterativo (*bootstrapping*). Apesar da capacidade em descobrir novos padrões, estes são baseados em outros pré-definidos, limitando assim as possibilidades. Além disso, esse método apresenta potencial para propagação de erro a cada iteração.
- *Supervised*: essa abordagem propõe um classificador binário para determinar se há uma relação específica entre duas entidades. Esses modelos recebem como entrada algumas características do texto como, por exemplo, palavras de contexto, *tags* de classes gramaticais, caminho de dependência entre entidades, *tags NER*, *tokens*, distância de proximidade entre palavras e outros.

Essa abordagem permite uma rígida supervisão, além da possibilidade de treinamento do modelo em modelos negativos. Porém, o alto custo no processo de anotação do *dataset* pode dificultar a generalização para outros domínios.

- *Distantly Supervised*: essa abordagem combina o uso de regras base (*seed*) como *Weakly Supervised* e o treinamento de um classificador, dos métodos *Supervised*. No entanto, ao invés de fornecer um conjunto de tuplas como *seed*, este é obtido de uma base de conhecimento existente como, por exemplo, *Wikipedia*, *DBpedia*, *Wikidata* ou outros. A partir disso, o trabalho manual para anotação do *dataset* é reduzido.
- *Few shot* com LLM: assim como na extração de entidades, essa abordagem permite a utilização do poder semânticos das LLM para extrair relações entre entidades. Para atingir um resultado satisfatório, é necessário apresentar alguns exemplos dentro do contexto do texto a ser extraído. Além disso, as relações podem ser extraídas de forma independente, ou juntamente com a extração de entidades, o que traz vantagens e desvantagens. No primeiro caso é possível informar as entidades na qual se deseja encontrar a relação, o que aumenta o controle do processo, aumentando também o tempo e custo do mesmo, no segundo caso o processo é mais rápido e menos custoso, e a LLM pode se beneficiar em fazer as tarefas ao mesmo tempo, visto a alta dependência que existe entre as duas, mas reduz o controle da extração.

A utilização de modelos de AP em extração de relações é observada principalmente com a abordagem supervisionada. Para fim de comparação, as técnicas clássicas mais comuns nessa abordagem, assim como as *features* de entradas e suas métricas de F1 podem ser observadas no quadro 5 (ZENG et al., 2014).

Quadro 5. Comparação entre diferentes técnicas de extração de relações.

Classificador	Conjunto de características	F1
SVM	POS, <i>stemming</i> , padrões sintáticos	60,1
SVM	Pares de palavras, palavras entre elas	72,5
SVM	POS, <i>stemming</i> , padrões sintáticos	74,8
MaxEnt	POS, morfológico, compostos nominais, thesauri, n-grams do Google, WordNet	77,6
SVM	POS, prefixos, morfológico, WordNet, análise de dependência, Levin Classed, ProBank, FrameNet, NomLex-Plus, n-gram do Google, paráfrases, TextRunner	82,2
RNN	POS, Reconhecimento de Entidades Nomeadas, WordNet	77,6
MVRNN	POS, Reconhecimento de Entidades Nomeadas, WordNet	82,4
CNN + softmax	Par de palavras, palavras ao redor do par de palavras, WordNet	82,7

Fonte: adaptado de Zeng et al. (2014).

Modelos baseados em *Transformers*, como o *BERT*, apresentam os melhores resultados na tarefa de ER. Isso acontece, porque estes modelos não produzem as representações de entidades no nível de *span*, de forma que eles precisam aprender tais representações com base em um conjunto de dados contínuos. Muitas entidades são divididas em vários *tokens* para se tornarem entradas para o modelo, mas a tarefa de pré-treinamento baseada em palavras não é adequada porque também divide entidades. Por exemplo, um modelo pode prever erroneamente "Anéis" na entrada "O Senhor dos [MASK]", onde na verdade, "O Senhor dos Anéis" é uma entidade.

4.3 GraphRAG

Uma vez construído o GC e o conhecimento estando codificado nessa estrutura de entidades e relacionamentos, tem-se uma base de conhecimento que apresenta uma maior facilidade de compreensão para o ser humano e codifica as várias relações existentes entre as entidades da base, uma representação que era inviável no banco vetorial. Porém, esse conhecimento deve ser acessado de forma dinâmica e, dependendo do que se deseja buscar (informações simples ou complexas), um ou vários nós e caminhos podem ser acessados. Por isso, faz-se necessário um método capaz de realizar esse processo de forma escalável e eficiente, visto que essa informação será utilizada em tempo real por um agente.

Pesquisas recentes têm explorado a tradução de grafos para a linguagem natural como, por exemplo, a transformação de nós e arestas em uma sequência de texto. No entanto, esse método enfrenta sérios problemas de escalabilidade, pois a

conversão de um grafo com milhares de nós e arestas em uma sequência de texto resulta em um número excessivo de tokens, ultrapassando a capacidade de entrada de muitos modelos LLMs (FATEMI; HALCROW; PEROZZI, 2023; ZHAO et al., 2023). Uma das formas de superar este problema é através da implementação do RAG em GC, o que permite a seleção, apenas, dos subgrafos relevantes para a pergunta. O uso de RAG com GC pode ser chamado de *GraphRAG*.

As principais características do GraphRAG são: utilização de grafos de conhecimento para a representação de informações, capturando entidades, relacionamentos e estruturas hierárquicas, o que permite uma compreensão do contexto semântico e dos relacionamentos e possibilita uma recuperação mais abrangente. O pré-processamento dos dados em um GC reduz o custo computacional e viabiliza uma recuperação mais rápida em comparação com abordagens RAG tradicionais. Além disso, o GraphRAG pode lidar com consultas complexas e multifacetadas ao sintetizar informações relevantes de várias partes do GC, oferecendo mais transparência e explicabilidade em comparação aos modelos de linguagem de caixa preta. O GraphRAG também possui a capacidade de aprendizado contínuo, expandindo seu conhecimento ao incorporar novas informações ao GC.

A principal alteração necessária ao agente está nos métodos de busca disponíveis que, a princípio, pode ser dividida entre busca global e local. A pesquisa global envolve a exploração de todo o CG para a identificação de entidades e relações pertinentes a *query*, oferecendo um conjunto de resultados mais completo. Utiliza-se a pesquisa global quando a consulta exige uma visão ampla e conexões de diferentes partes do grafo. Essa abordagem é vantajosa para consultas que não são muito específicas e que se beneficiam de um contexto mais amplo, mas pode levar um tempo maior. Por outro lado, a pesquisa local concentra-se em um subgrafo específico e restringe a pesquisa a uma região menor e bem definida do GC. Essa abordagem é útil para consultas que necessitam de informações detalhadas e precisas de um contexto específico. Por exemplo, considerando a *query* “Qual é a relação entre a Doença A e o Medicamento X?” e o resultado da comparação em termos de processo, tipos de resultados esperados e exemplos possíveis de saída, pode-se observar no quadro 6 como esses tipos de pesquisa se comportam.

Quadro 6. Comparação da busca local e global para *query* “qual a relação entre a Doença A e o Medicamento X?” considerando o processo, tipo de resultados esperados e exemplos possíveis de saída.

Tipo de busca/ parâmetro	Busca Global	Busca Local
Processo	Todo o GC será pesquisado para encontrar todas as conexões possíveis entre a Doença A e Medicamento X.	Apenas o subgrafo em torno das entidades Doença A e Medicamento X serão utilizados e o foco será em conexões imediatas (grau 1).
Resultado	A busca pode revelar não apenas as interações diretas entre a Doença A e o Medicamento X, mas também conexões indiretas por meio de outras doenças, medicamentos, sinais ou sintomas relacionados.	O resultado nesse caso deve revelar apenas as interações diretas que existem entre a Doença A e o Medicamento X, gerando um resultado mais direto e focado nas duas entidades da <i>query</i> .
Saída	<ul style="list-style-type: none"> - O medicamento X é utilizado no tratamento da Doença A, Doença B e Doença C. - Durante o doutorado do Dr. M, ele descobriu o medicamento X para a doença A. - Doença A também pode ser tratado com Medicamento Y. 	<ul style="list-style-type: none"> - Medicamento X é utilizado para tratar Doença A. - Medicamento X foi desenvolvido 10 anos depois da descoberta da Doença A. - Quem tem a Doença A deve tomar o Medicamento X por 1 ano.

Fonte: próprio autor.

Ao avaliar as diferenças principais entre o RAG e o GraphRAG, percebe-se vários aspectos incomuns, dentre eles, observa-se que o RAG estabelece relacionamentos entre passagens por meio de similaridade de vetores semânticos, enquanto o GraphRAG se baseia em diversas relações entre entidades como, por exemplo, taxonômicas, lógicas e tempo. A forma de raciocínio do RAG é baseada em cadeias de similaridade, ao passo que o GraphRAG utiliza inferências multi-hop sobre os caminhos do grafo. A inferência no RAG se dá através de projeções neurais (*embeddings*), já no GraphRAG ocorre uma interpretação explícita sobre caminhos do grafo. A dinâmica de exploração no RAG é baseada em deriva semântica, enquanto no GraphRAG é focada em travessias entre entidades-relação. Em termos de modularidade, o RAG carece de suporte nativo na maioria dos bancos, ao contrário do GraphRAG, que permite subgrafos personalizados com restrições únicas. A evolução no RAG requer mudanças externas no modelo, visto que a semântica é representada por um *embedding* que é calculado por uma rede neural externa ao banco. No GraphRAG ocorre um aprendizado contínuo da estrutura a partir dos dados pela característica do banco em permitir aumento e mudanças na estrutura, *labels* e propriedades. A confiabilidade no RAG pode ser questionável, visto que a informação

é representada por sinais de relevância que, normalmente, formam um rank, o que pode variar conforme a qualidade do modelo utilizado para calcular o *embedding*. No entanto, ressalta-se que no GraphRAG as relações são validadas e os rastros são explicáveis. Por fim, o RAG tem sua precisão limitada por associações implícitas entre a semântica da *query* e documentos, enquanto o GraphRAG apresenta um equilíbrio entre profundidade e escala.

Apesar da possibilidade de utilização dos *benchmarks* para PR em RAG, há escassez de *benchmarks* abrangentes para tarefas de raciocínio baseadas em grafos, sendo que a maioria dos existentes se concentram em tarefas mais básicas como, por exemplo, grau de nó, existência de arestas e cálculo de caminho mais curto, dificultando, assim, a comparação de propostas de GraphRAG (FATEMI; HALCROW; PEROZZI, 2023; WANG et al., 2023).

5 METODOLOGIA

Neste capítulo serão apresentadas o MemoryGraph e as metodologias utilizadas. Será apresentado os diferentes processos propostos para um agente conversacional com foco na capacidade de codificação e armazenamento do conhecimento em um GC, recuperação de informação de forma mais rica e contextual, e como o agente interage com sua memória de curto e longo prazo para personalizar personalização da experiência do usuário.

Sendo assim, a metodologia é dividida em duas grandes partes:

1) codificação e armazenamento do conhecimento: apresentação do processo de extração de triplas e construção do GC, considerando diferentes estratégias de incorporação do conhecimento, além das propostas de modelagem da base em diferentes níveis de abstração do conhecimento. Além disso, como é possível utilizar o conhecimento e experiências mantidas na memória de curto prazo do agente e concretizá-las na memória de longo prazo (GC) como novos conhecimentos e como uma representação dinâmica do usuário para usos em conversas futuras;

2) recuperação do conhecimento: apresentação é realizada a criação do contexto e incorporação do conhecimento do usuário em uma nova conversa. É apresentada diversas formas de realizar a busca da informação no GC, abordando as várias oportunidades que o mesmo apresenta para enriquecer o contexto do agente.

5.1 Codificação e Armazenamento do conhecimento

Inicialmente, tem-se que definir a estrutura de memória que se pretende guardar as informações, como será codificado a informação que será recebida, quais são essas informações, quais as limitações e outros aspectos. O ambiente que será considerado para esse estudo é um AI capaz de receber perguntas e interações de um usuário humano e, de forma natural, respondê-las da melhor forma possível. Para isso, ele poderá utilizar as experiências prévias com esse usuário por meio de conversas anteriores e, também, poderá consultar uma base de conhecimento previamente criada com informações relevantes ao contexto.

5.1.1 Camadas de abstração do conhecimento

Inspirado por diferentes trabalhos que apresentam vantagens em modelar a estrutura hierárquica do documento, outros sobre a modelagem do conhecimento como fatos, as vantagens da busca híbrida (simbólico e subsimbólica) (HIMMELHUBER et al., 2021), estudos do funcionamento da memória humana que apresentam vários níveis de abstração do conhecimento e, finalmente, considerando o cenário de uma ferramenta empresarial, que possua vários clientes, a proposta se sustenta em um banco de dados que modele as informações em vários níveis de abstração e com o maior grau de conectividade. As figuras 22 e 23 apresentam essa modelagem, considerando cinco camadas de abstração e apresentando como as entidades se relacionam entre elas. Importante notar que ele modela tanto a informações lexicais (conexões entre estruturas do documento e estrutura organizacional, como qual documento pertence a qual usuário, qual segmento pertence a qual documento), como informações semânticas (como as entidades se relacionam semanticamente entre elas), permitindo diferentes tipos e níveis de busca, tanto para a nível de contexto semântica para responder perguntas, quanto para organização e isolamento de informação. Cada camada é descrita a seguir:

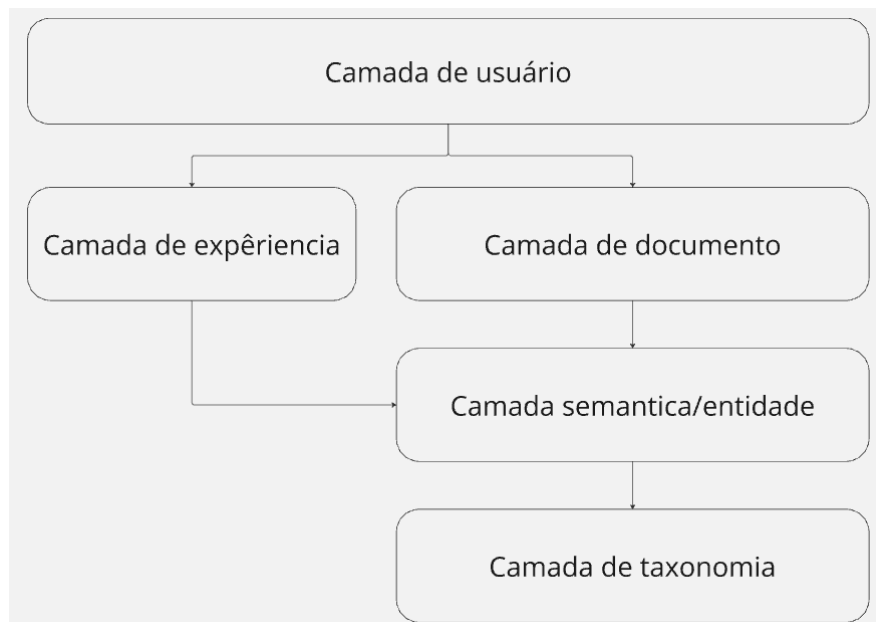
- Camada de usuário: responsável por armazenar as informações do usuário como, por exemplo, identificadores, nomes, *status* e quaisquer informações relevantes ao sistema.
- Camada de documentos: essa é uma camada léxica, que possui uma estrutura similar a um banco vetorial em que são salvos os documentos de forma hierárquica e sequencial. Como pode ser visto na figura 23, nessa camada foi criado um nó que representa o documento como um todo, incluindo em suas propriedades as informações gerenciais (como, por exemplo, o caminho onde se encontra o documento, nome, tipo, resumo do documento para acesso e busca a nível de documento), recebendo a *label* “document”. Esse nó está relacionado ao usuário a partir de uma relação de pertencimento que permite um futuro controle de acesso. Cada parte do documento, chamado de forma mais genérica de segmento, estão salvos em um nó, que recebe a *label* “segment”. Qualquer documento que possa ser salvo como texto pode ser codificado dessa forma. Por exemplo, um áudio pode ser transcrito e essa transcrição dividida em segmentos, seguindo a regra que fizer sentido para a

aplicação. Esses nós estão todos relacionados ao nó do documento e também estão relacionados de forma sequencial entre eles, indicando a ordem que eles aparecem no documento. Essa relação foi criada como forma de facilitar a recuperação de informação sequencial, processo essencial da memória humana. Além disso, esses nós de segmento recebem um ou mais nós relacionados a eles com o *embedding* que pode ser utilizado para buscas semânticas. A escolha de colocar o *embedding* em um nó separado e não no próprio nó foi realizada para permitir que um mesmo segmento tenha vários *embeddings*. Isso permite várias representações e menor esforço na troca de *embedding*, visto que o *Neo4j* permite a criação de múltiplos *vector-index* baseados no nó especificado.

- Camada de semântica/entidades: essa camada possui a semântica extraídas dos documentos e experiências do agente. Cada entidade, representada pelo nó, é relacionada a um ou mais segmentos, visto que uma mesma entidade pode ocorrer em vários locais diferentes e até mesmo se relacionar à outras entidades, formando um fato (entidade, relação, entidade). Dessa forma, é possível relacionar diferentes segmentos que pertencem a diferentes documentos em decorrência das entidades em comum entre eles. Esses nós recebem a *label* definida pelo processo de extração, mas também uma *label* “entity” para rápido acesso e isolamento das informações dessa camada.
- Camada de taxonomia: aqui estão salvos os dados extraídos da *wikidata* que classificam essa entidade. Esse processo foi realizado utilizando o serviço de busca da *wikidata* por meio do *sparql* (disponível em: <<https://query.wikidata.org/sparql>>) e automatizado por meio do código *python* a partir da biblioteca *SPARQLWrapper* (disponível em: <<https://sparqlwrapper.readthedocs.io/en/latest/>>). Essa informação é utilizada para trazer um conhecimento geral sobre as entidades extraídas, como por exemplo, a informação de que Brasília é uma cidade e a capital do Brasil, esse tipo de informação permite uma nova forma de relacionar entidades, aumentando a conectividade da base e ajudando na desambiguação das entidades.
- Camada de experiência: essa é uma camada relativo ao armazenamento, de forma resumida, das experiências que o agente teve com o usuário. Nesse estudo, essa experiência se resume a conversas com o usuário, em que uma

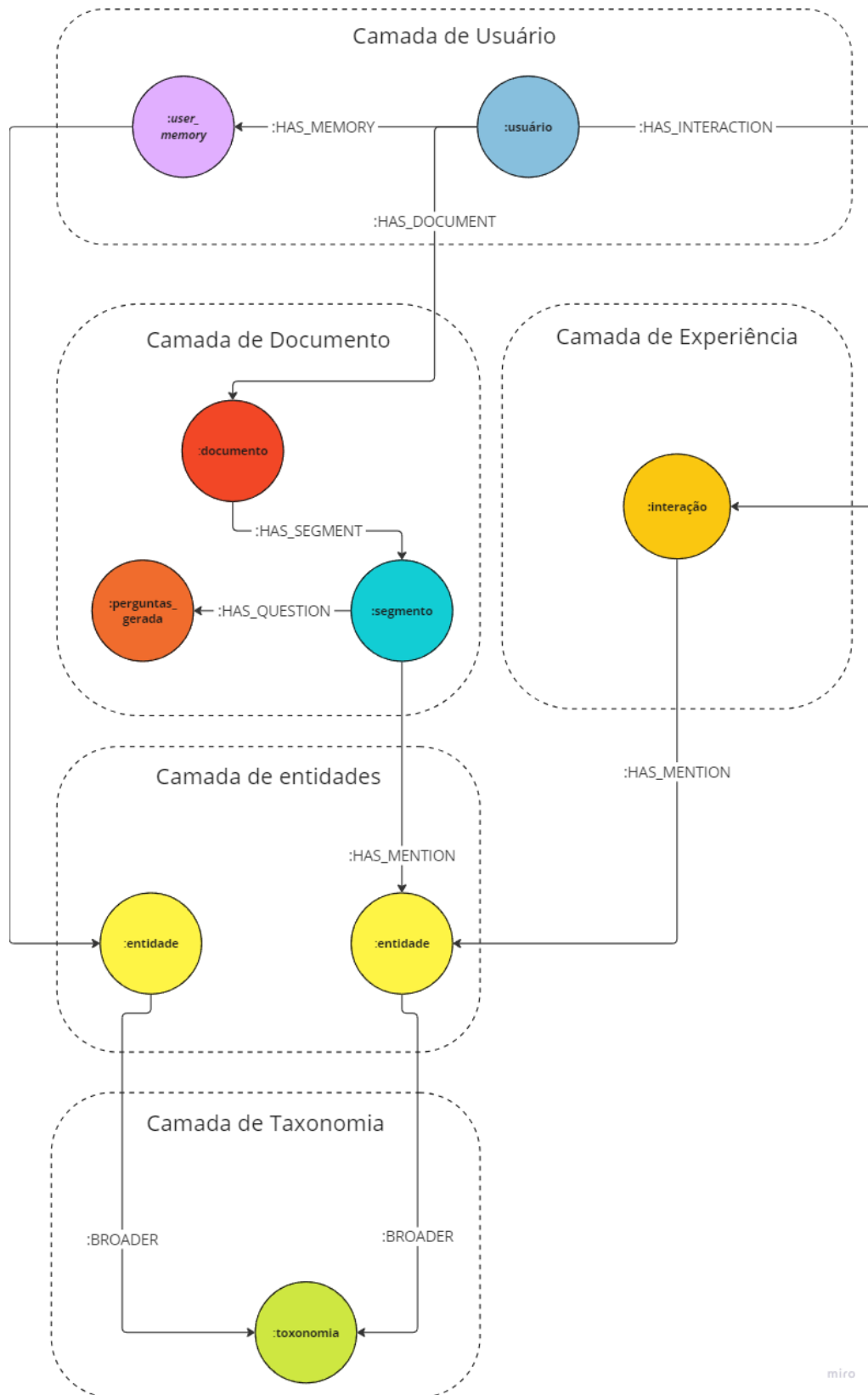
conversa termina sempre que o usuário ultrapassa dez minutos sem responder ou interagir com o agente. Dessa forma, toda a memória dessa conversa (considerada como memória de curto prazo) é concretizada através de um processo na base conhecimento (se tornando memória de longo prazo). Qualquer novo fato descoberto durante essa interação é inserido na camada semântica, de forma que seja possível referenciar essa interação com base no conhecimento extraído e até relacionar interações. Esse processo será melhor detalhado e discutido ao apresentar a etapa de incorporação conhecimento de experiência.

Figura 22. Representação da modelagem das camadas de abstração do conhecimento.



Fonte: próprio autor.

Figura 23. Exemplo de como as entidades se relacionam entre as camadas de abstração.



Fonte: próprio autor.

5.1.2 Pipelines de incorporação de conhecimento

Após a definição da estrutura da base de conhecimento (memória do agente), definiu-se o processo de recebimento, codificação e armazenamento de novas informações para transformá-las em conhecimento adquirido. Dessa forma, são duas as formas de incorporação do conhecimento: por meio de documentos fornecidos ou por meio de interações com o usuário.

5.1.2.1 Pipelines de incorporação de documentos na memória de longo prazo

Nesta seção será apresentado o *pipeline* de incorporação de documentos na base de conhecimento. Esta parte do princípio de que o sistema recebeu um documento e é composto pelas seguintes etapas:

- 1) Segmentação do documento: cada documento recebido foi fragmentado em segmentos. Essa segmentação objetivou a manutenção da semântica do bloco de informação e também considerou o tamanho adequado para a codificação dessa informação em *embedding* (subsimbólico), assim como para a extração de triplas (simbólico). Blocos muito grandes podem gerar *embeddings* pouco representativos, assim como blocos pequenos podem gerar informação com pouco contexto. Dessa forma, foi utilizada a segmentação por parágrafos e em seguida estes foram agrupados por número de *tokens* para evitar que um parágrafo fosse fragmentado ao meio ou fossem parágrafos muito curtos. Os experimentos apresentaram como o GC cresce e se comporta com a variação de *tokens*.
- 2) Descontextualização de segmentos: cada segmento foi submetido ao processo de descontextualização ou resolução de correferência, em que cada parágrafo foi alterado de forma que não fosse necessária outra parte do texto para ser compreendido. Por exemplo, dado um parágrafo com a frase: “Maria foi ao supermercado” e dado outro parágrafo com a frase: “Ela aproveitou que foi ao supermercado e comprou pão”, observa-se que no segundo caso a informação não pode ser totalmente compreendida sem a contextualização com a frase anterior, pois não está claro a quem se refere o pronome “ela”. Por isso, a segunda frase deve ser reescrita como: “Maria aproveitou que foi ao supermercado e comprou pão”. Esse processo permite

que os segmentos do texto possam ser compreendidos de forma independente, viabilizando o processamento de texto muito grandes.

- 3) Extração de triplas: a tripla é composta por uma entidade (origem), uma relação e uma entidade (destino): (origem, relação, destino). Cada entidade se tornará um nó no grafo que possui um ou mais *labels* e um conjunto de propriedades que são extraídas no mesmo processo. Essa extração pode ser realizada de forma livre de restrições ou a partir de uma ontologia sendo informada. Essa ontologia pode ser fixa e dependente do domínio ou pode ser dinâmica e baseada banco já existente. Apesar de suportar ambos os casos, nesse estudo foi utilizada uma versão híbrida em que foi extraído o *schema* do banco atual e utilizada suas *labels* sempre que possível. Caso isto não fosse possível, foi também utilizada uma nova *label*. Dessa forma, houve a manutenção da liberdade de crescimento da base com novos termos e classes de informações ao mesmo tempo o controle do crescimento com normalização dos termos e *labels* já existentes, reduzindo a probabilidade de nós duplicados. Além disso, a extração foi realizada de duas formas:

- a) extração de tripla em um único passo em que foi realizada a definição das entidades, sua classe (representada pela *label*) e suas propriedades, além de todas as relações contendo essa entidade.
- b) extrações das entidades de forma isolada à extração das relações. Nesse caso as tarefas foram divididas para reduzir a complexidade das tarefas e permitir que o processo de extração de relações fosse realizado com entidades já processadas e possivelmente com mais contexto para melhorar a qualidade das extrações.

Ambos os casos e suas influências nos resultados serão apresentados na fase de experimentação. Vale ressaltar que, nem sempre, a entidade extraída faz parte de uma tripla e que existem entidades que aparecem no documento sem a geração de um fato (entidade, relação, entidade). Porém, estas foram mantidas visto que podem aparecer em diferentes documentos e permitir uma relação indireta.

Dois princípios de extração de triplas podem ser empregados dependendo da característica desejada: 1) um cujo foco é na extração somente dos nós mais

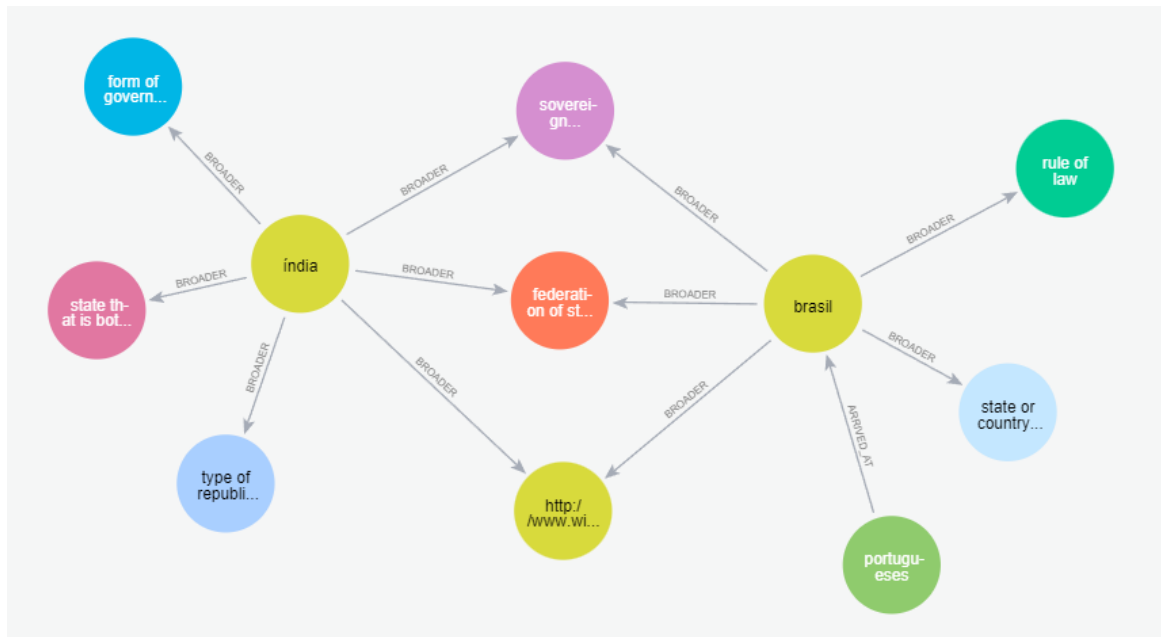
importantes e o maior número de propriedades da entidade, de forma a manter o grafo mais organizado com cada nó o mais representativo possível, se assemelhando mais a um grafo de propriedade (esse tratamento favorece uma abordagem híbrida em que a camada de entidades complementa os segmentos de texto e possibilita a obtenção de mais informações em caminhos mais curtos ao explorar as proximidades e considerar a densidade de informação presente em cada nó); e 2) e o segundo cujo foco é na extração do maior número de informações possíveis em forma de entidades ao invés da extração de características da entidade como propriedades dos nós (se assemelhando, dessa forma, a um RDF). Como consequência da representação de toda informação do documento em triplas, o potencial aumento no número nós seremos acompanhados de maiores relações entre os nós. Essa escolha se deu a fim de se obter relações menos óbvias e favorecer buscas em maior profundidade em razão de possibilitar uma maior cobertura do conhecimento e simplificar a criação de *queries*.

- 4) Relacionamento das entidades com *wikidata*: nessa etapa foi executada a busca de todas as entidades na *wikidata* por meio do sistema de busca de *sparql* query que possibilita a obtenção da página mais relevante para cada entidade. Isso permitiu a obtenção de diversos fatos relativos a cada entidade, além do subgrupo ao qual ela pertence. Essas *labels* foram utilizadas tanto para enriquecer a informação, quanto para normalizar a *label*, utilizando aquela que foi obtida na *wiki* ao invés da extraída no processo de descoberta, além de permitir realizar a resolução de entidades através da URL da *wikidata*, entidades iguais possuem a mesma URL, isso é possível devido ao sistema de busca proporcionado pelo serviço da *wikidata*. Esse processo de relacionar as entidades a uma base de conhecimento geral e pré-existente permitiu a normalização da base e integração com outras bases que passaram pelo mesmo processo. Ademais, como será apresentado posteriormente, essas informações adicionais auxiliaram na relação de informações extraídas e no agrupamento das mesmas, criando relações antes não existentes, como pode ser visto na figura 24. O seguinte processo foi executado para cada entidade extraída:

- a) Busca da entidade por meio do sistema de busca do *wikidata* para a obtenção da página com maior *match-score*.

- b) Como a página possui um ID único, este foi utilizado para a busca na *wikidata* de todas as subclasses e instâncias daquela entidade. Para isso foi utilizado a relação P31 (*instance of*) e P279 (*subclasse of*) da *wikidata* (disponíveis, respectivamente, em: <<https://www.wikidata.org/wiki/Property:P31>> e <<https://www.wikidata.org/wiki/Property:P279>>). Por exemplo, se considerarmos a entidade “K2”, o resultado será que “K2” é uma instância de “montanha”, “vulcão” é uma subclasse de “montanha” e uma instância de “relevo vulcânico”. Esse processo pode acontecer de forma recursiva e extrair vários níveis de taxonomia. Porém, nas primeiras experimentações foi relatado que primeiro nível é suficiente para informar as principais classes dessa entidade, visto que se houver aprofundamento nas instâncias e subclasses do resultado encontrado, a taxonomia se torna muito genérica. Por exemplo, para a entidade “Vinicius de Moraes”, tem-se que ele é uma instância de “humano”, mas “humano” também é subclasse de “onívoro”, “mamífero”, “pessoa” e “animal”. Essas informações são consideradas gerais e de alto nível e não agregam conteúdo para decisões do cotidiano.
- c) Em seguida, os resultados dessa taxonomia foram relacionados às respectivas entidades. A taxonomia com maior relevância para a entidade foi atribuída como *label* nesta para normalizar a base. Além disso, a descrição encontrada na página da *wiki* referente a entidade foi acrescentada como propriedade ao nó dessa entidade, trazendo uma descrição comum entre as bases que utilizaram a *wiki* e aumentando o contexto agregado a essa entidade.

Figura 24. Exemplo de como a taxonomia agrega à entidade uma nova camada de conhecimento geral.



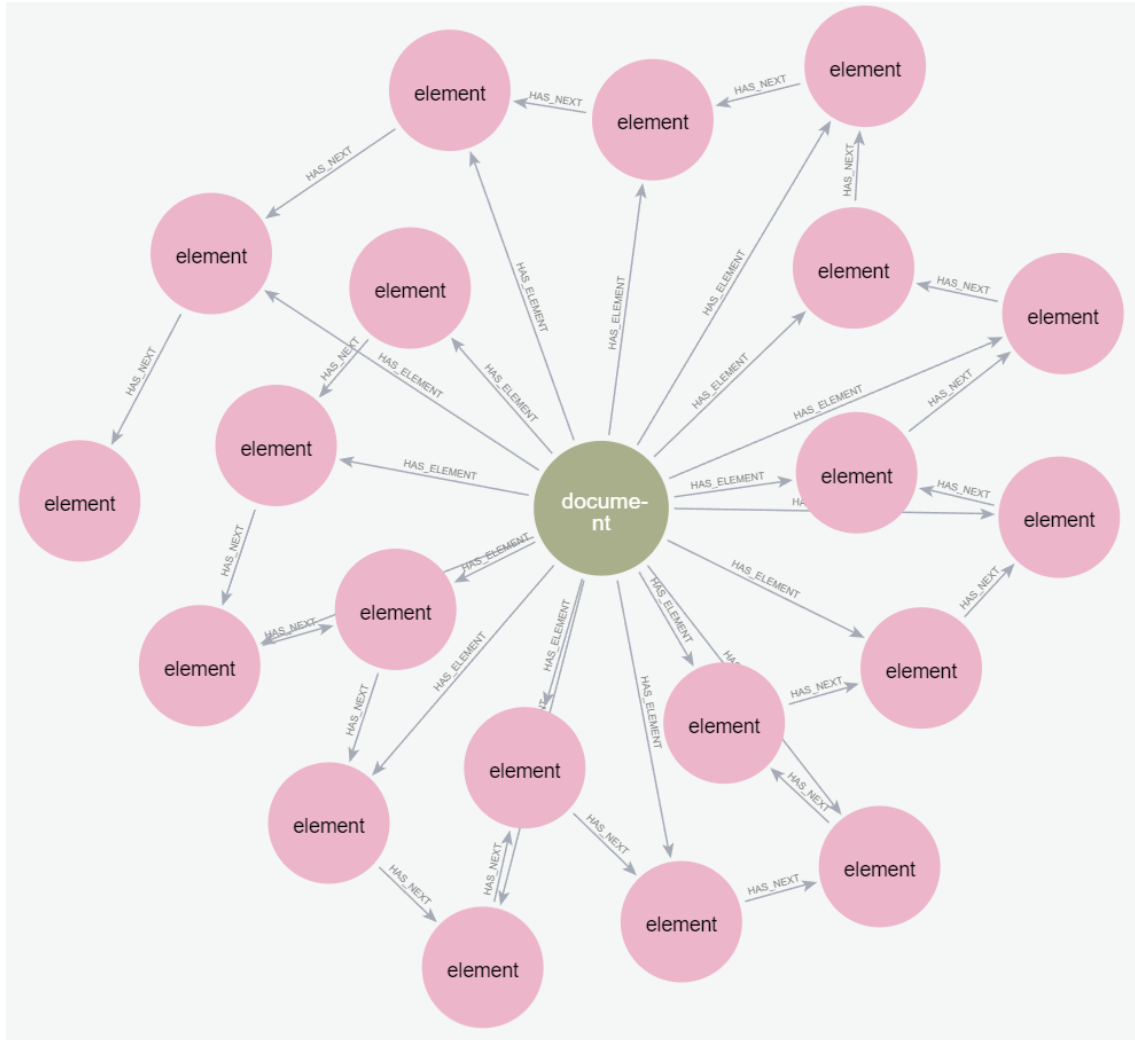
Legenda: Este caso demonstra o entendimento de que “Brasil” é um país, uma república federativa, um estado laico, um estado de direito e um estado soberano. Enquanto a Índia é uma república constitucional democrática e possui algumas relações com o Brasil.

Fonte: próprio autor.

- 5) Busca de relações entre entidades extraídas na *wikidata*: esse processo foi realizado de forma similar ao anterior, mas com foco nas relações. Dada duas entidades que possuam resultado na *wikidata*, foi executada a busca por uma relação qualquer entre elas. Caso a relação encontrada fosse próxima à relação extraída, esta foi então substituída pela extraída para obter normalização. Caso a relação encontrada fosse diferente da extraída, foi então criada uma nova relação entre essas duas entidades como forma de enriquecimento desse fato. Esse processo de identificação de proximidade da relação encontrada na *wikidata* com a extraída foi executado por meio de distância sintática, utilizando-se *fuzzy*.
- 6) Elaboração de perguntas para cada segmento: para cada segmento foi gerada uma pergunta em que a resposta pode ser encontrada no segmento. Esse processo foi orientado por trabalhos que realizam a busca de informação por similaridade da pergunta do usuário às perguntas previamente geradas.

- 7) Cálculo de *embedding*: Para cada segmento e perguntas gerados foi calculado o *embedding* e criado um nó conectado ao segmento correspondente. Nos experimentos deste estudo, todos os *embeddings* foram calculados por meio do text-embedding-ada-02 via API da OpenAI (disponível em: <platform.openai.com/docs/guides/embeddings>). A existência de um *vector-index* conectado ao grafo permitiu que: 1) fosse possível simular um RAG tradicional baseado em base vetorial sem a necessidade de duplicação de informações ou manutenção de bancos separados; 2) fossem possíveis buscas híbridas em que uma busca vetorial é realizada para indicar pontos de entrada no grafo e, em seguida, uma busca em profundidade é realizada a partir do nó encontrado pela busca vetorial (semântica).
- 8) Construção do grafo de conhecimento: com as triplas formadas, foi realizado um processamento para agrupar as entidades iguais não só sintaticamente, mas semanticamente. Essas triplas foram organizadas em uma estrutura de dados que representa o grafo em *python*. O documento e seus elementos foram relacionados e cada elemento demonstra uma relação direcional com o próximo. Dessa forma, é possível expressar a hierarquia do documento e como cada elemento se relaciona, como pode ser observado na figura 25. Esse processo permitiu a representação de diferentes tipos de estrutura e hierarquias a depender do processo de extração do documento.

Figura 25. Representação de exemplo de subgrafo da estrutura hierárquica do documento em que este tem relação com todos os seus elementos e relação sequencial entre si.

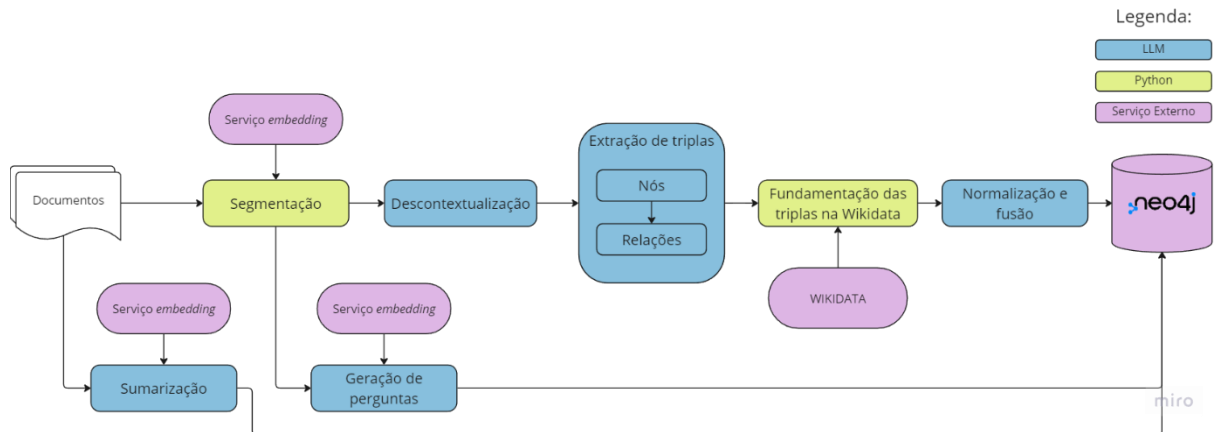


Fonte: próprio autor.

- 9) Adição do grafo construído ao grafo existente no banco de dados de grafo: a adição do grafo construído ao existente foi executada por meio do Neo4J (WEBBER, 2012). Essa ferramenta permite a visualização gráfica do grafo de conhecimento e busca por meio da linguagem *Cypher* e busca semântica por meio da distância de cosseno.

A figura 26 demonstra esse processo de forma visual e a figura 27 apresenta uma representação do grafo após a incorporação de um documento sobre história do Brasil processado por esse *pipeline*.

Figura 26. Fluxo de extração de informação de documentos e inserção na base de conhecimento.



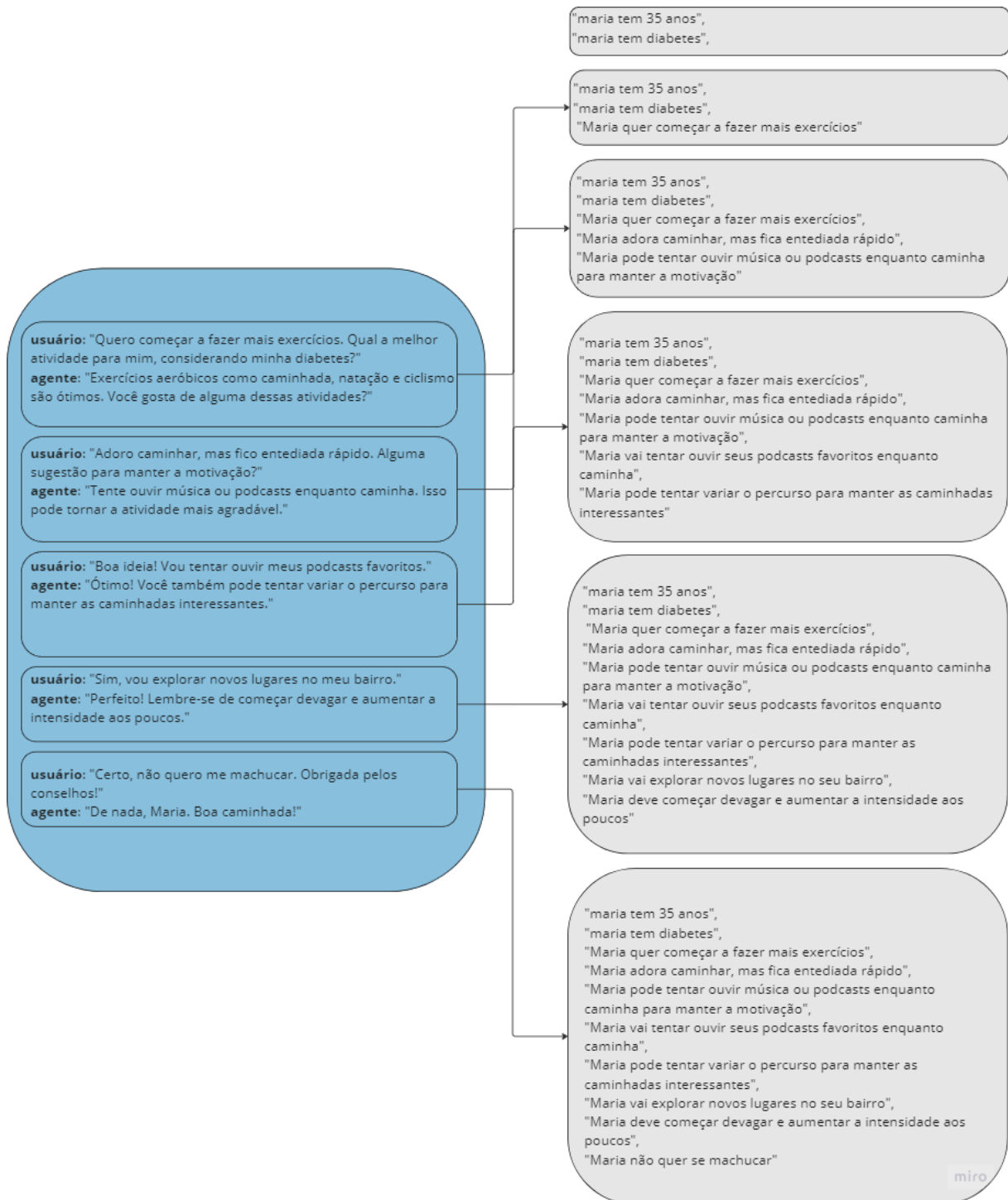
Fonte: próprio autor.

5.1.2.2 Memória de curto prazo

A memória de curto prazo é a informação dinâmica extraída a cada interação que é utilizada como contexto da conversa atual pelo gerador. A abordagem mais comum é a adição de mensagens anteriores no *prompt*, porém essa abordagem leva ao aumento nos *tokens* durante uma conversa. Em decorrência disso, ocorre o aumento do custo de conversas longas e, dependendo do modelo LLM que é utilizado, podem existir limitações no tamanho da memória disponível. Uma opção para mitigar esse obstáculo e ainda manter as informações relevantes da conversa é a sua sumarização. Porém, esse processo ainda pode levar à perda de informações em contextos muito extensos. Outro cenário que pode decorrer da sumarização é a junção de informações entre a informação gerada e a recebida do usuário. Por isso, foi proposta uma opção à sumarização: adição à memória de curto prazo de fatos relevantes do usuário. Essas informações são dinâmicas e são atualizadas a cada interação com o usuário, sendo que ao final da conversa ela é integrada à memória de longo prazo sobre o usuário.

As informações obtidas da conversa podem ser divididas em três tipos: 1) conhecimento geral; 2) informação sobre a conversa (armazenamento da experiência que o agente teve nessa interação) que pode ser utilizada para conversas futuras e possibilitar várias atividades temporais como, por exemplo, lembrar tarefas, confirmar consequências de sugestões, ajustar comportamento e outros; e 3) preferências do usuário que formarão o *user_memory* (conhecimento condensado do usuário com base em experiências passadas). Essa divisão permite acessos diferentes em momentos diferentes, pois o *user_memory* é carregado a cada início de conversa e está sempre disponível na memória de curto prazo do agente, enquanto as experiências ficam salvas como conhecimento do agente e podem ser acessadas quando necessário via *pipeline* de busca de informação. A figura 28 apresenta o processo de atualização da memória de curto prazo durante a conversa.

Figura 28. Representação de exemplo de atualização da memória de curto prazo durante a conversa com o agente.



Fonte: próprio autor.

5.1.2.3 Incorporação da experiência na memória de longo prazo

Ao final de conversa toda, a informação e experiência adquirida deve ser concretizada na memória de longo prazo. Esse processo foi fortemente inspirado pelo aprendizado humano por meio da concretização das informações contidas na memória de curto à memória de longo prazo e esse processo objetiva o armazenamento do conhecimento adquirido da experiência entre a interação humano-máquina. Os três tipos de informações foram processados e extraídos no GC por meio das seguintes etapas:

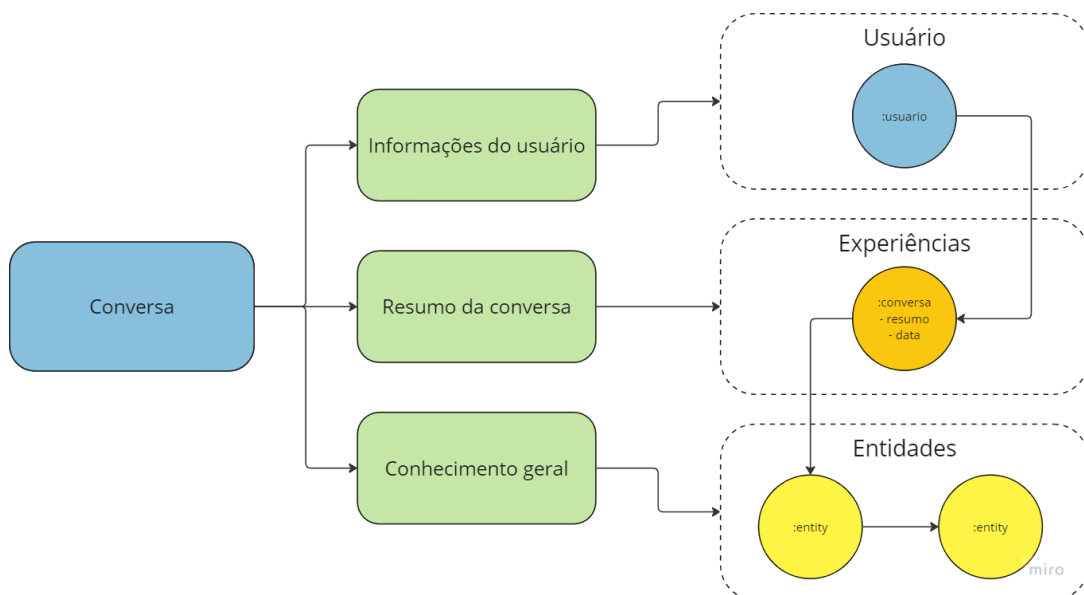
- 1) Extração dos fatos e conhecimento geral: este é o conhecimento novo que o agente adquiriu durante a conversa. Esse processo pode apresentar conflitos de informações entre o que o agente sabe previamente e o que o usuário informou. Porém, partindo do princípio de que o agente é capaz de questionar a veracidade, foi considerado aqui que, se a informação se manteve como verdadeira na memória de curto prazo do agente, significa que ela é correta e deve ser adicionada a sua base. Porém, é importante ressaltar que este é um comportamento que pode variar dependendo da aplicação e ambientes. Nesta primeira etapa do processo foram extraídas apenas triplas, ou seja, não foram extraídas entidades fora de uma relação, pois entende-se que um fato deve conter ao menos a relação entre duas entidades. Além disso, essa extração foi realizada sem propriedades e sem o processo de relacionamento com *wikidata*. Ao final desse processamento foi realizada a inserção dessa informação no GC (camada de entidades) para permitir que informações de diferentes fontes possuam interação.
- 2) Extração de preferências e características do usuário. As preferências formam um conjunto de informações que são representadas por uma lista de *strings* e que estão relacionadas à interesses e características pessoais do usuário como, por exemplo, “eu prefiro história do Brasil à europeia”, “eu prefiro respostas mais curtas” ou “eu sou diabético”. Essas informações foram utilizadas como *user_memory* em futuras interações do usuário para possibilitar uma maior personalização das interações. Dessa forma, o agente pode se adaptar às preferências e peculiaridades de cada usuário sem a necessidade de ajuste de modelo. Essa informação foi inserida como propriedade do nó que representa o usuário em forma de nós e relações. A segunda forma de inserção

é considerada mais adequada quando se tem múltiplos usuários por instância do GC por permitir a geração de relações entre usuários baseado em suas preferências. Além disso, isso permite uma melhor interação entre outras informações da base, apesar de maior dificuldade de manutenção. Dessa forma, considerou-se que a última informação extraída do usuário sempre prevalecerá sobre a mais antiga, visto que a visão do agente sobre o usuário é volátil e o usuário pode alterar suas preferências no decorrer do uso do agente.

3) Concretização da interação. Esse processo consistiu na criação de um novo nó que representa a interação do agente com um usuário. Neste novo nó está contido o resumo da conversa e pode conter, também, informações adicionais sobre a conversa como, por exemplo, sentimentos, data que ocorreu a interação, o tipo de interação (voz ou texto), qual plataforma de mensagem ou vídeo foi utilizada, quais documentos foram recebidos e outras informações importantes para a aplicação do agente. Nesse estudo, para fins de simplificação e foco, foram inseridos a data e o resumo da conversa.

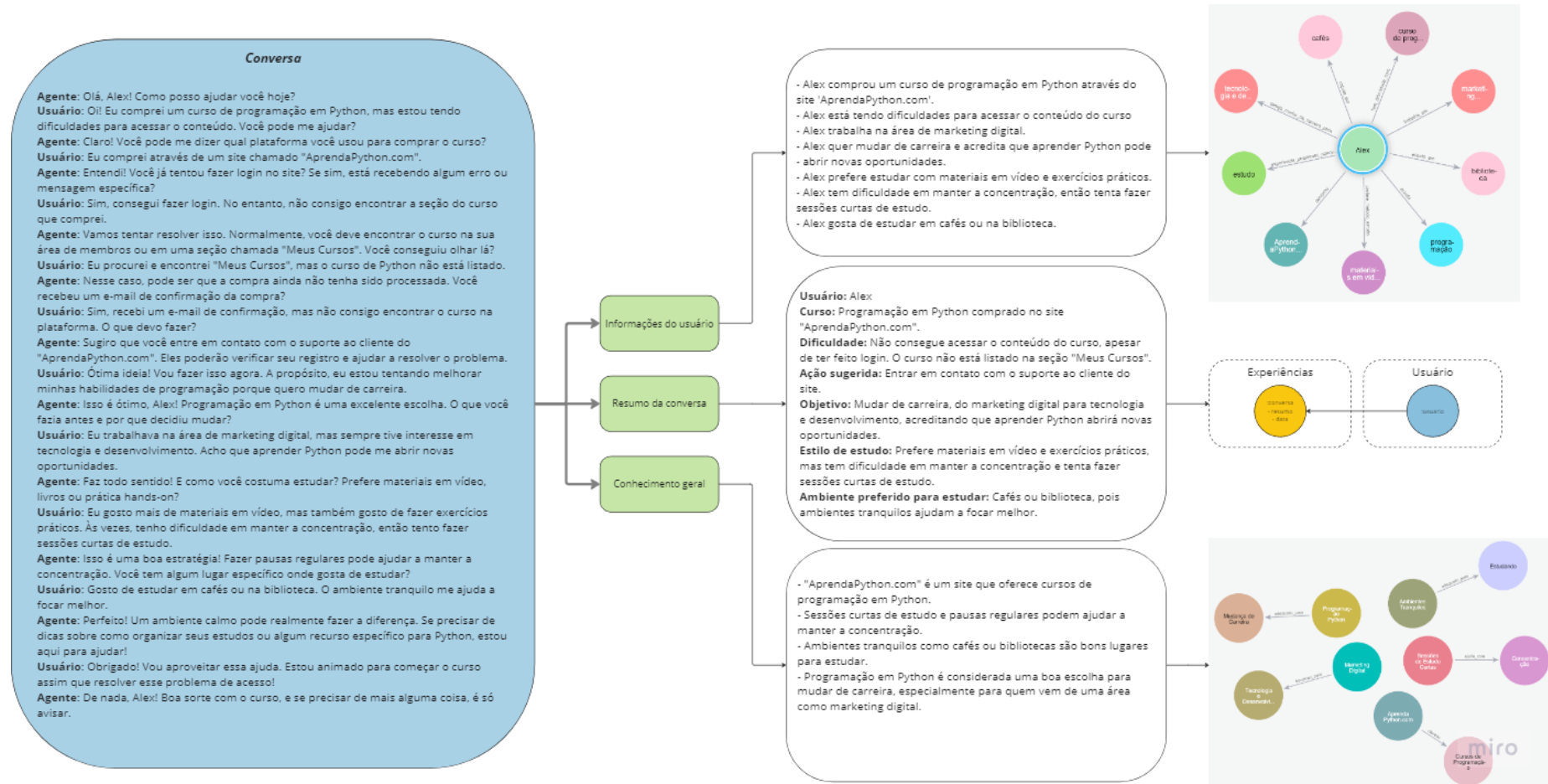
Esse processo de três etapas está representado na figura 29. A figura 30 exemplifica esse processo em uma conversa simulada.

Figura 29. Representação da incorporação da experiência na memória de longo prazo.



Fonte: próprio autor.

Figura 30. Exemplo de uma conversa com os três tipos de informações extraídas: informação do usuário, resumo da conversa e conhecimento geral.



Fonte: próprio autor.

Dessa forma, tem-se um sistema com o conhecimento codificado de forma híbrida: conhecimento simbólico por meio do CG e o conhecimento subsimbólico e o conhecimento em um espaço semântico (por meio dos *embeddings*). Além disso, tem-se o conhecimento intuitivo do agente que é responsável pelo comportamento instintivo e raciocínio (representado pelos pesos internos do modelo LLM).

5.2 Recuperação de informação

A proposição deste *pipeline* de construção do grafo e alteração da forma de armazenamento e codificação da informação foi decorrente das novas oportunidades de busca que essa abordagem pode proporcionar no momento que a informação é requisitada. Porém, novas codificações podem alterar a forma como a informação é retornada da base de conhecimento, ao passo que em um banco vetorial, o resultado da busca é uma lista de textos ranqueados pela distância semântica com a *query*. Dessa forma, o resultado do GC pode variar consideravelmente dependendo da *query* executada na busca, podendo este ser um resultado tabular, com múltiplos campos por elemento da lista de resultados. Por exemplo, ao aplicar a *cypher query* contida no quadro 7 (que busca todas as pessoas que atuaram em algo e que nunca dirigiram algo), pode-se obter como retorno uma lista de resultados. Nessa lista de resultados, cada linha é composta por três campos, sendo dois deles “nós” (p e m) e uma relação (r). Além disso, cada um deles possui nome e *label* e pode conter uma lista de propriedades.

Quadro 7. Resultado da busca no Neo4J para a *Cypher query* apresentada.

Cypher query	MATCH (p:Pessoa) - [r:ATUOU_EM] -> (m) WHERE NOT (p) - [:DIRIGIU] -> () RETURN p, r, m		
Resultado da Cypher query	p	r	m
	(:Person {name: 'Tom Hanks', born: 1956})	[:ACTED_IN {roles: ['Zachry']}]	(:Movie {title: 'Cloud Atlas', released: 2012})
	(:Person {name: 'Tom Hanks', born: 1956})	[:ACTED_IN {roles: ['Forrest Gump']}]	(:Movie {title: 'Forrest Gump', released: 1994})

Fonte: próprio autor.

Dessa forma, para possibilitar que o modelo LLM receba essa informação, esta deve ser convertida em um texto, processo conhecido como transcrição. A transcrição do subgrafo resultante de uma *query* é um campo de estudo em expansão e cuja relevância cresceu com a disseminação dos GCs. Para a realização dessa tarefa, pode-se utilizar um modelo LLM a fim de que essa transcrição resulte em um texto mais natural. Porém, nesse trabalho, foi utilizada uma formatação em json desse resultado em razão de: 1) um agente deve responder o mais rápido possível para a manutenção da naturalidade da conversa, engajamento do usuário engajado e resolução do problema apresentado o mais rápido possível. Dessa forma, a adição de uma inferência de um modelo LLM para formatação de informação pode preencher um tempo relevante no *pipeline* de resposta; 2) se o modelo LLM é capaz de compreender e formatar em um parágrafo, isso indica que, no contexto de uma resposta para uma pergunta, este também vai compreender essa formatação original do Neo4J; e 3) para manter a proposta a mais genérica e escalável possível. Dessa forma, o agente suportará todo tipo de resultado de busca e não dependerá de um modelo externo para codificação da informação extraída do GC. Um exemplo de como ocorre a formatação da saída do GC para utilização de contexto do agente pode ser observada na figura 31.

Figura 31. Exemplo de como é formatado a saída do grafo de conhecimento para utilização de contexto do agente.

```
{
  "results": [
    {
      "person": {
        "name": "Tom Hanks",
        "born": 1956
      },
      "relationship": {
        "ACTED_IN": {
          "role": "Zachry"
        }
      },
      "movie": {
        "title": "Cloud Atlas",
        "released": 2012
      }
    },
    {
      "person": {
        "name": "Tom Hanks",
        "born": 1956
      },
      "relationship": {
        "ACTED_IN": {
          "role": "Forrest Gump"
        }
      },
      "movie": {
        "title": "Forrest Gump",
        "released": 1994
      }
    }
  ]
}
```

Fonte: próprio autor.

Como o foco é em uma proposta de memória para gente conversacional, foi utilizado como base um agente simplificado contendo apenas os componentes essenciais para sua caracterização, sendo eles: capacidade conversacional através de múltiplas interações; memória de curta e longo prazo; capacidade de acesso a base de conhecimento, caracterizada pelo *pipeline* de RAG ou *GraphRAG* quando a base utilizada é um GC. Algumas propostas de agente apresentam a capacidade de planejamento, reflexão, autocrítica e decomposição de problema. No entanto, no presente trabalho foi considerado que estes não são módulos essenciais para o agente conversacional e grande parte dessas capacidades foram simplificados em diretrizes no *prompt* principal do agente.

A ferramenta de busca é a forma de interação do agente com sua memória de longo prazo e é definida como o *pipeline* do *GraphRAG*. Esse módulo de acesso à informação foi construído de forma que seja possível realizar diferentes tipos de buscas, dependendo da necessidade do agente.

É importante destacar que, sempre que uma busca semântica ou vetorial foi realizada, a *query* foi submetida ao processo de cálculo de *embedding* utilizando o serviço da OpenAI no modelo *text-embedding-ada-02* (disponível em: <<https://platform.openai.com/docs/guides/embeddings>>).

O processo de busca de informação de contexto para o modelo de LLM foi realizado através de dois fluxos: 1) extração da representação do usuário que será utilizado para personalização da conversa; 2) ferramenta de busca de informação na base de conhecimento, constituído de várias opções de busca e utilizando *queries* pré-estabelecidas para alcançar um tipo de informação ou criação da *query* em tempo de execução para prover maior liberdade ao agente. Nesta sessão também será apresentado a estratégia de tradução das informações extraídas do grafo de forma a otimizar a representação da informação para o modelo LLM.

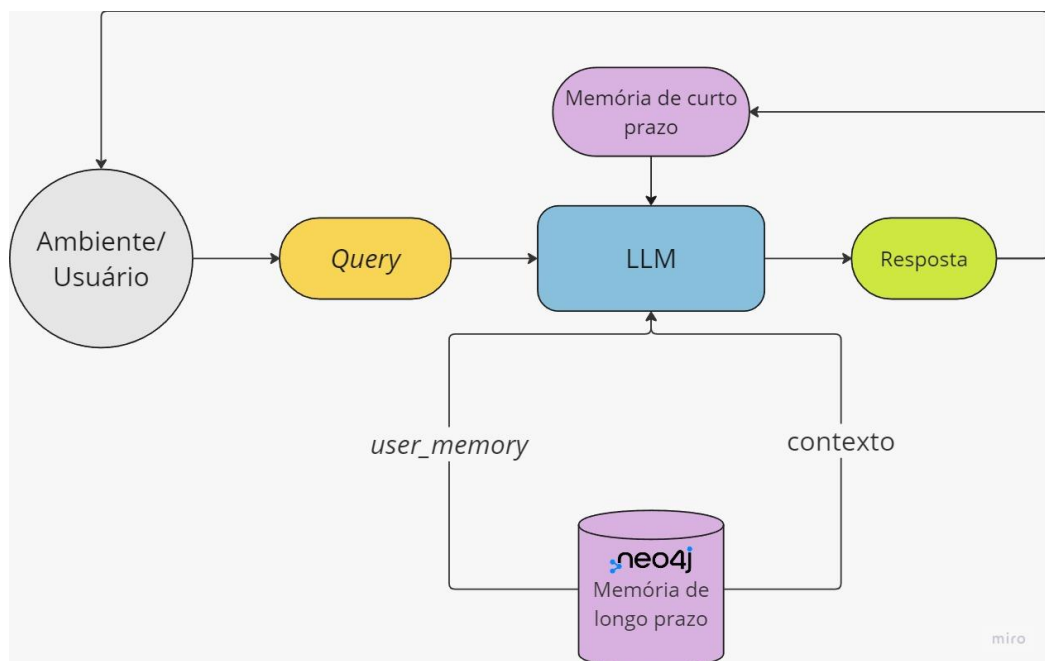
De forma complementar à fase de codificação e armazenamento do conhecimento, em que o modelo LLM suporta o processo de construção do GC, neste estudo é o GC que tem como objetivo suportar a geração do modelo LLM, finalizando o ciclo de suporte entre as duas ferramentas.

A figura 32 representa o fluxo do agente inteligente proposto quando o usuário envia uma mensagem, podendo ser descrito nos seguintes passos:

- 1) Se for a primeira mensagem do usuário na conversa, então deve ser carregado da memória de longo prazo do usuário a representação desse usuário contendo seus dados, experiências com o agente e preferências. Essa informação é formatada de maneira estruturada como um dicionário de informações e é chamado aqui de *user_memory*.
- 2) Recebe mensagem do usuário via interface de texto (ou voz, que é transcrito em texto) e essa entrada do usuário é chamada de *query*.
- 3) Um *prompt* com as diretrizes de funcionamento e comportamento esperado é carregado e complementado com: *query* do usuário; *user_memory*; histórico de mensagens trocadas entre usuário e o agente nessa conversa.

- 4) Esse *prompt* é enviado a um LLM e este pode decidir entre responder com essas informações ou retornar um pedido de busca na base de conhecimento.
- 5) Caso uma resposta seja gerada, ela é então enviada para o usuário.
- 6) Caso a resposta seja o pedido de busca da base de conhecimento, é então iniciada a ferramenta de busca que será definida na sessão seguinte.
- 7) O resultado da busca é formatado e inserido no *prompt* como contexto.
- 8) Esse *prompt* é reenviado para o LLM gerar a resposta.
- 9) Essa resposta é salva na memória de curto prazo que é representada pelo histórico da conversa. Caso o limite de mensagens seja alcançado, é iniciado, de forma paralela, um processo de sumarização das mensagens antigas, mantendo apenas as informações relevantes.
- 10) Essa resposta é então enviada ao usuário.

Figura 32. Fluxo de execução do agente.



Legenda: em vermelho estão destacados os componentes que se alteram em relação a arquitetura de referência, sendo elas: 1) a memória de longo prazo; 2) *user_memory* criado ao fim da conversa e carregado no início como contexto para o agente; e 3) conhecimento adquirido durante a interação com o usuário.

Fonte: próprio autor.

A fim de explorar as opções de busca que o GC possibilita, foram propostas diversas *cypher queries* para a busca de informações específicas. Nesse contexto, é possível definir algumas buscas fundamentais e, por meio da combinação entre elas, gerar buscas mais elaboradas, denominadas global e local. As buscas fundamentais podem ser definidas como:

- Busca semântica: utiliza um índice conectado aos *embeddings* dos nós (elementos, sumário, e perguntas geradas).
- Busca textual: utiliza o mecanismo do Neo4J baseado no *apache lucene* (disponível em: <<https://lucene.apache.org/core/>>) através de um índice que está conectado aos campos de texto dos nós.
- *Cypher query*: executa a busca por nomes ou *labels* de entidades específicas relacionados a *query* do usuário.
- Busca em profundidade: a partir de um nó inicial é realizado a travessia através das relações e explorando as vizinhanças do nó, e repete o processo até alcançar algum critério de parada. Então, todos os nós e relações nesse caminho são retornados.

As buscas globais utilizam todas as informações disponíveis em diversas camadas de abstração da base de conhecimento. Os elementos descritos nas buscas podem se referir aos segmentos textuais e também aos nós que representam as interações do agente, permitindo não somente a busca de informação de documentos, mas também de conversas passadas, a seguir são apresentadas algumas buscas globais:

- Primeiramente é realizada busca semântica nos nós de elementos e em seguida é executada a busca de todos os fatos relacionados aos elementos para cada de entidades. Isso permite que fatos formados por entidades em diferentes elementos sejam retornados, garantindo que uma das entidades sempre estará relacionada aos elementos encontrados na busca inicial. Isso traz, assim, informações que não seriam possíveis apenas com os segmentos.
- Inicialmente é realizada a busca por elementos relacionados a *query* do usuário e, a partir dos elementos encontrados, ocorre uma busca em profundidade por meio das entidades até o encontro de outros elementos. Esses novos nós são mantidos apenas se possuírem um alto grau de relação com os elementos

originais e isso é medido pelo número de caminhos que os ligam. Esse processo auxilia no cenário de *multihop* e mitiga a limitação de similaridade do *embedding* na busca de maior variedade de contexto.

- O mesmo processo de busca semântica é realizado, porém sobre as perguntas geradas. Esse método parte do princípio de que perguntas semelhantes possuem contextos semelhantes. Dessa forma, ao encontrar a *query* mais semelhante é possível utilizá-la como entrada para encontrar segmentos de textos e fatos relacionados.

Com o objetivo de avaliar a capacidade do agente ao utilizar apenas as informações extraídas e levantar o questionamento sobre a necessidade da busca vetorial nos documentos como um todo, as buscas locais não utilizaram os segmentos de texto (camada de documento). Assim, são apresentadas as seguintes buscas locais:

- Busca por fatos através das entidades da *query*. Essa busca tem como ponto de entrada as entidades extraídas da *query* do usuário. Cada entidade extraída da *query* do usuário é utilizada para gerar uma busca por entidades. No entanto, é utilizado um modelo LLM para gerar variações possíveis dessas entidades considerando plural, singular, siglas e idiomas para reduzir a probabilidade de uma entidade que existe na base não ser encontrada por estar redigida de maneira diferente. Em seguida, todas as triplas que esse nó participa são extraídas em forma de fatos.
- Informação de comunidade: é obtida por meio da busca por entidades relevantes para a *query* do usuário, assim como no método anterior. Em seguida, é realizada uma busca em nível de taxonomia partindo desses nós que resulta em um subgrafo que contém toda a informação que gira em torno dessa entidade. Esse subgrafo é, então, utilizado para gerar um resumo textual que representa a informação contida nessa comunidade. Essa busca tem como objetivo representar um conhecimento mais amplo relacionado aos principais conceitos da *query* e é considerada útil para problemas que necessitam de um contexto mais amplo sobre um assunto.

As principais vantagens dessa abordagem são: o controle do resultado retornado, garantia de funcionamento das *queries* e, ao conhecer a base, é possível reduzir o aumento de complexidade das *queries* com o crescimento do GC. Porém, esse método limita quais informações podem ser buscadas. Por isso, é proposto um segundo método, onde o LLM gera a *query* de busca de informação, dado o *schema* do GC e a pergunta que se deseja responder. Métodos similares para tradução de pergunta em SQL existem, e podem ser aplicados com sucesso (ZHOU et al., 2023; GUO et al., 2022; XU; LIU; SONG, 2017), mas são pouco traduzíveis para grafos, visto a natureza dinâmica do CG e do próprio *Cypher*. Esse método oferece uma liberdade de busca maior, permitindo busca de informações em situações não planejadas, utilização de mais recursos estruturais do GC, como busca em profundidades maiores e uso da camada de taxonomia para classificação de informações (FRANCIS et al., 2018). Por outro lado, essa abordagem se torna cada vez mais complexa com o crescimento do *schema*, além de ser dependente da qualidade da LLM na geração de *cypher*.

Uma opção apresentada para o problema da complexidade do *schema* na criação da *cypher query* é a possibilidade de não o apresenta-la à geração da mesma, ao invés disso, gerar uma *query* que considere as muitas variações possíveis da *label* e nome sempre baseado nas entidades principais da pergunta. Como exemplo, pode-se citar: dada a pergunta “qual o melhor medicamento para diabetes?”, as principais entidades nessa sentença são, “medicamento” e “diabetes”. Dessa forma, uma *query* que busca por nós que representam essas entidades deve considerar as variações dessas, como plural, contrações, idioma, sinônimos como, por exemplo, “medicamentos”, “remédio”, “medicine”, “diabete”, “dm”, “diabetes mellitus”. Essa solução representa uma opção para redução considerável do impacto do GC na criação da *query*, mas ao mesmo tempo essa falta de informação estrutural impede a geração de *queries* mais complexas, sendo ideal para busca a nível de entidades e vizinhanças.

Outra opção é a geração do *schema* do subgrafo relevante para a pergunta, baseado em entidades selecionadas e profundidade que se desejada buscar, as ferramentas de banco de dados de grafo podem retornar esse *schema*. Porém, esse processo envolve uma inferência no modelo LLM ou em algum método específico para

extração de palavras chaves, em seguida deve ser realizado a busca no GC e finalmente a geração do *schema*, para só então, ser possível gerar um *Cypher query* que será aplicado apenas nesse subgrafo. Em um ambiente que é necessário respostas rápidas, esse processo adiciona muitos passos, levando a um aumento importante no tempo de resposta. Dessa forma, alternativas que não necessitem acessar múltiplas vezes a banco de dados e realizar múltiplas inferências em LLM é preferível, logo, a primeira abordagem de gerar queries expandidas ou fornecer um *schema* enxuto dos níveis de informação do grafo é mais viável, e é a abordagem utilizada nessa proposta.

6 EXPERIMENTOS, RESULTADOS E DISCUSSÃO

Os experimentos foram definidos de forma a testar os diferentes aspectos do MemoryGraph, que é: a construção do GC baseado em informações externas; avaliação das capacidades e possibilidades de busca e contextualização do agente utilizando novo paradigma baseado em GC; e avaliação do impacto que o conhecimento extraído da experiência do agente, através do *user_memory*, tem na conversa do agente com o usuário.

Todos os experimentos foram executados em um computador com as seguintes configurações técnicas: 32 Gb RAM, CPU 8-Core Processor 4.20 GHz, SSD 1 terabyte, sistema operacional Windows 11 com WSL2 (Linux). O banco de dados de grafo utilizado foi o Neo4J por meio de uma instância gratuita no Aura DB. Também foi utilizado o serviço de cálculos de *embedding* da OpenAI por meio do modelo de *embedding* `text-embedding-ada-02` (disponível em: <<https://platform.openai.com/docs/guides/embeddings>>) e, sempre que não for especificado de outra forma, o modelo LLM utilizado foi o GPT-4o via API da OpenAI. A seguir serão apresentados os experimentos realizados, seus resultados e discussões sobre eles a partir dos tópicos: construção do GC, recuperação da informação geral e impacto do *user_memory* na experiência conversacional.

6.1 Construção do grafo de conhecimento e incorporação do conhecimento na memória de longo prazo.

A fim de validar a proposta de criação e o impacto de diferentes processos na criação do GC, elaborou-se um *dataset* formado por 40 manuais sobre a patologia crônica diabetes mellitus em português (PT-BR), com uma média de cinco páginas por documento (levantado por um grupo de profissionais da área de saúde), nomeado neste trabalho de *diretrizes-diabete* (este foi utilizado para validar o cenário de documentos pequenos, com muitos termos técnicos e que produz uma alta taxa de extração e inserção). Esses documentos foram levantados por um time de três profissionais da área de saúde, utilizando apenas fontes públicas e de órgãos oficiais.

Além disso, um processamento foi executado para tratamento de todos os documentos como apenas um grande documento, garantindo a mesma quantidade

de texto em ambos os casos. Este segundo foi elaborado para validar cenários de documentos extensos e como o tamanho deste pode influenciar no GC, visto que podem alterar importantes características do processo de adição de conhecimento. Para cada documento do *dataset* inserido no GC, foram calculados a quantidade de nós, *labels*, relações e tempo de criação e inserção. A tabela 1 contém exemplos de entradas do dataset contendo pergunta, resposta esperada e fontes utilizadas para a resposta apresentada.

Os experimentos receberam nomenclatura aditiva (quando algumas características são adicionadas ao nome), sendo elas:

- *size_x*: em que “x” é o tamanho do segmento do texto utilizado.
- *triple_x*: em que “x” pode ser “1” ou “2” (“1” quando os nós e relações foram extraídas juntas e “2” para extração separada).
- *wikidata*: representando o que foi extraído da taxonomia do *wikidata*.
- *unique*: representando o *dataset* unificado em um só documento.

Com o objetivo de avaliar o impacto da extração de triplas, a forma que elas são extraídas, a utilização de dados do *wikidata* e o tamanho dos segmentos de texto no resultado final do GC, foram propostos os seguintes cenários:

- 1) *size_512*: segmentos de 512 sem triplas pra ter um baseline do custo de extração entidades.
- 2) *size_512-triple_1*: segmentos de 512 com triplas extraídas em conjunto.
- 3) *size_512-triple_2*: segmentos de 512 com triplas extraídas separadamente.
- 4) *size_512-triple_2-wikidata*: segmentos de 512 com triplas extraídas separadamente e *wikidata*.
- 5) *size_256-triple_2*: segmentos de 256 com triplas extraídas separadamente.
- 6) *size_1024-triple_2*: segmentos de 256 com triplas extraídas separadamente.
- 7) *size_256-triple_2-unique*.
- 8) *size_512-triple_2-unique*.
- 9) *size_1024-triple_2-unique*.

Tabela 1. Exemplos de entradas do *dataset* contendo pergunta, resposta esperada e fontes utilizadas para a resposta apresentada.

Exemplo de entrada do <i>dataset</i>	Exemplo 1	Exemplo 2
Pergunta	"Qual é a relação entre hepatites virais crônicas e o desenvolvimento de diabetes mellitus tipo 2?"	"Como a infecção pelo HBV ou HCV pode contribuir para a resistência à insulina?"
Resposta esperada	"Diversos estudos têm demonstrado um aumento no risco de diabetes mellitus tipo 2 (DM2) em pacientes com hepatites virais crônicas, especialmente aqueles que evoluem para doença hepática crônica e cirrose. A infecção pelo vírus C (HCV) está associada a uma prevalência maior de DM2 em comparação com outras hepatopatias, incluindo a hepatite B (HBV)."	"A infecção pelo HCV pode promover resistência à insulina (RI) por múltiplas vias patogênicas, incluindo a inibição direta da sinalização intracelular da insulina, estresse oxidativo, ativação de vias inflamatórias e modulação de incretinas. Para HBV, a relação com resistência à insulina não é conclusiva, mas alguns estudos sugerem que pode estar presente especialmente em pacientes com cirrose."
Fontes extraídas do <i>dataset</i>	<p>1) "Diversos estudos têm demonstrado aumento do risco de DM2 e de resistência à insulina (RI) em pacientes com hepatites virais crônicas."</p> <p>2) "A infecção pelo HCV está associada a maior prevalência de DM2, sendo superior à encontrada em outras formas de hepatopatias, incluindo a hepatite crônica pelo HBV."</p> <p>3) "A hepatite pelo vírus C é uma das principais causas de doença hepática crônica, incluindo cirrose e câncer."</p>	<p>1) "A presença de DM2 em pacientes com infecção crônica pelo HCV aumenta o risco de CHC, sendo observada maior prevalência desse tipo de câncer em pacientes com hepatite C crônica com DM2 em comparação a pacientes com hepatite C crônica sem DM2."</p> <p>2) "Em pacientes com HCV, têm sido demonstradas diversas alterações da sinalização pós-receptor de insulina. Estudos in vivo demonstraram que a proteína do core do HCV genótipo 1 é capaz de inibir a fosforilação do substrato do receptor de insulina-1 (IRS-1), resultando em RI e diabetes."</p> <p>3) "A relação entre infecção crônica pelo HBV e DM2 é controversa. Alguns autores defendem que a maioria dos estudos que demonstraram associação entre DM2 e hepatite B não consideraram em que fase da infecção os pacientes se apresentavam e que esse é um potencial fator de confusão nessas análises."</p>

Os resultados desses experimentos podem ser observados nas figuras 33 a 47, que demonstram como os parâmetros do grafo crescem com a inserção dos documentos, além disso, pode ser observado no quadro 2 o resultado final do grafo para cada um dos cenários de inserção propostos e apresentados anteriormente. Vale ressaltar que no caso do dataset processado como um único documento (definido pela palavra-chave *unique*), só existe o estado antes e depois da inserção, visto que é só ocorre uma inserção, por isso não aparece nos gráficos de crescimento da figura 43 a 37, que foi construído com o estado do banco a cada documento inserido.

Figura 33. Gráfico de crescimento do número de nós a partir da variação do número de documentos inseridos.

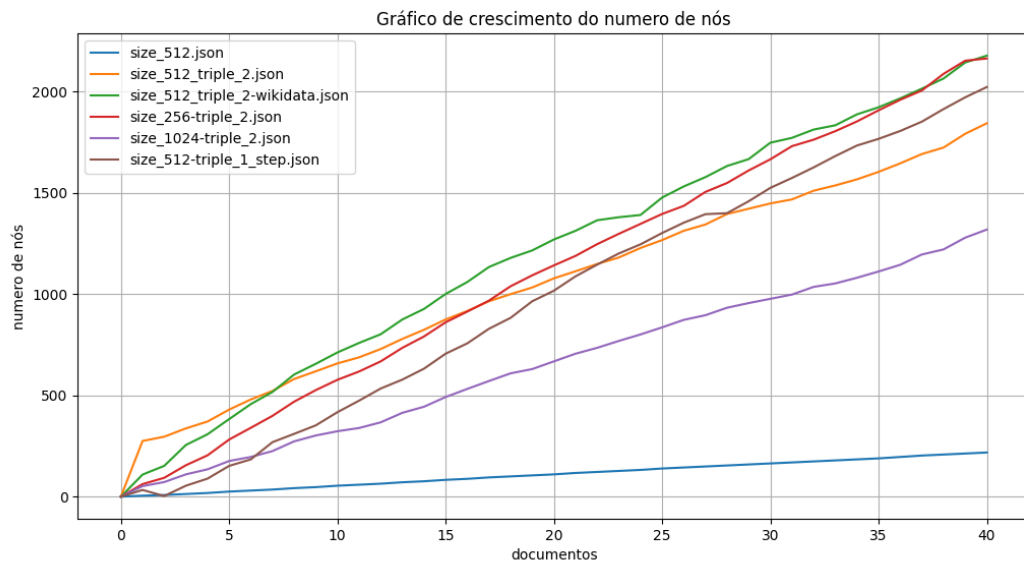


Figura 34. Gráfico de crescimento do número de relações a partir da variação do número de documentos inseridos.

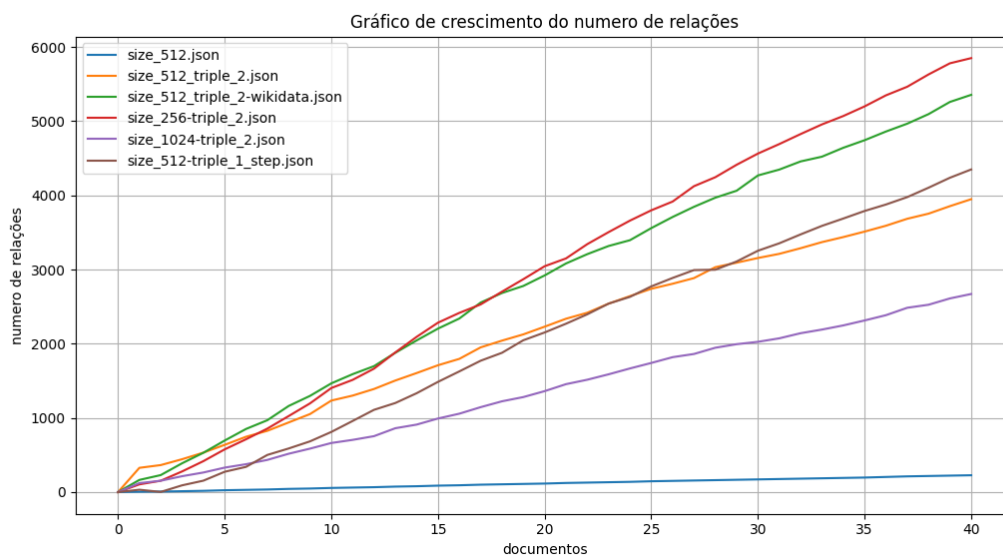


Figura 35. Gráfico de crescimento do número de *labels* dos nós a partir da variação do número de documentos inseridos.

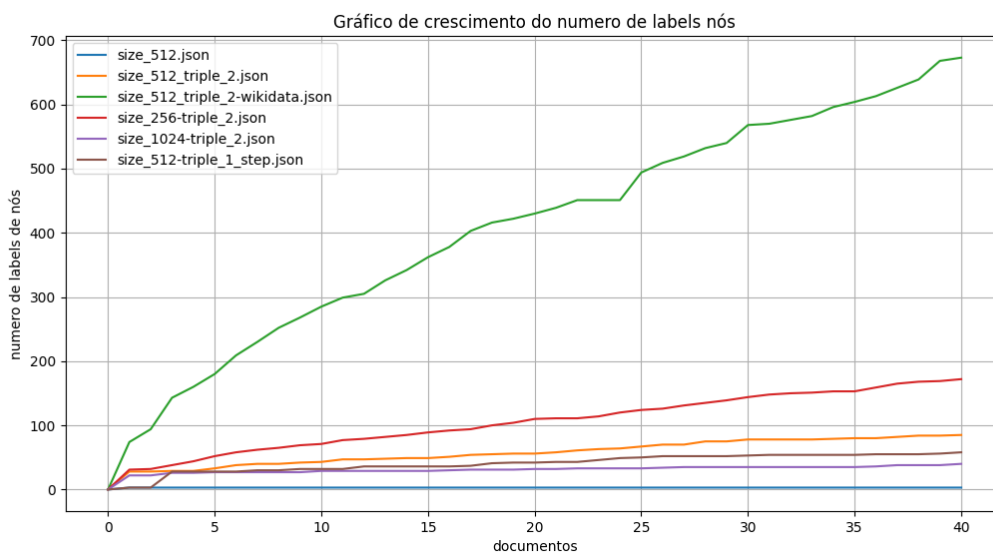


Figura 36. Gráfico de crescimento do número de tipos de relações a partir da variação do número de documentos inseridos.

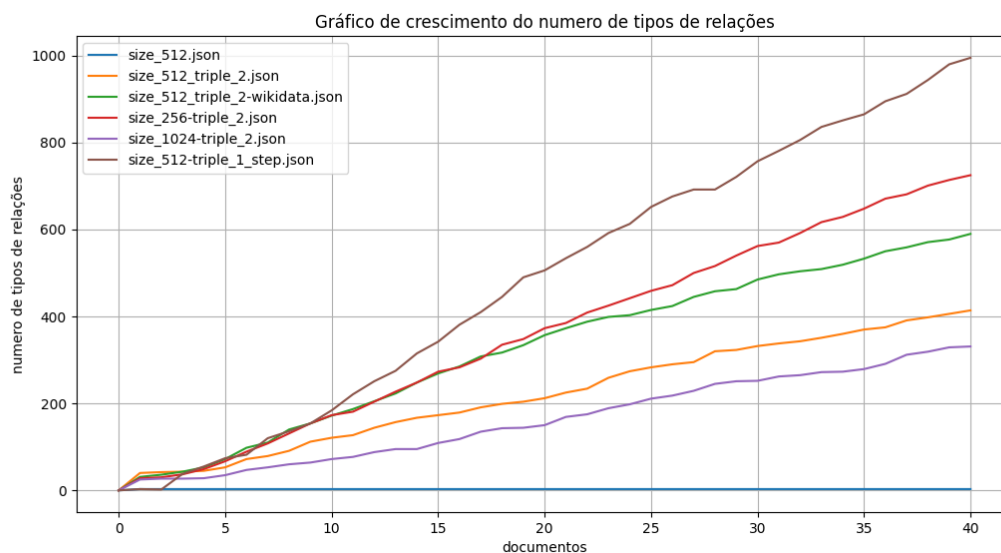


Figura 37. Gráfico demonstrando o tempo necessário para processamento e construção do grafo de conhecimento.

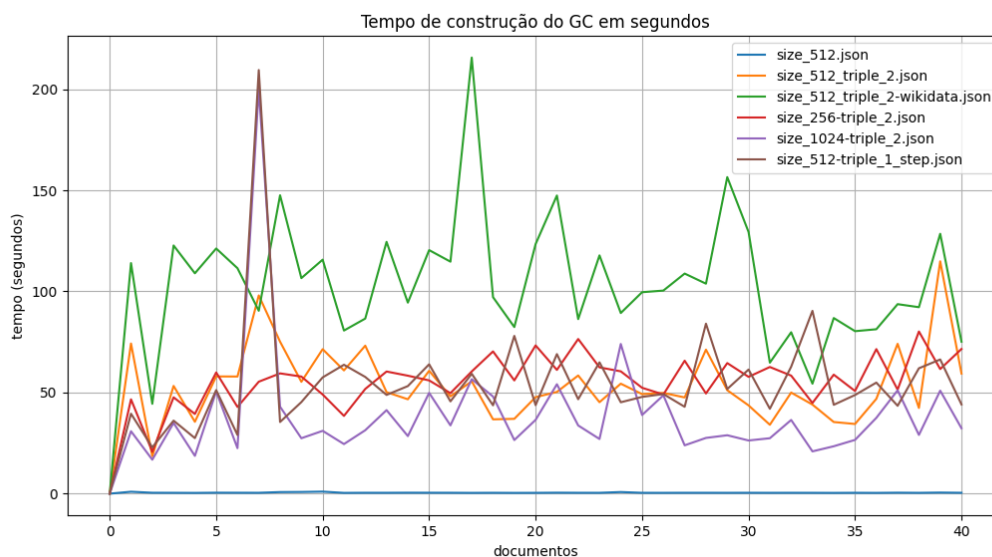


Tabela 2. Estado final do grafo de conhecimento em cada cenário de inserção.

id	Tipo de inserção	Entidades	Relações	Labels de nós	Tipos de relações
1	size_512	218	227	3	3
2	size_512-triple_1	2023	4350	58	995
3	size_512-triple_2	1844	3949	85	414
4	size_512-triple_2-wikidata	2178	5356	673	590
5	size_256-triple_2	2164	5851	172	725
6	size_1024-triple_2	1319	2671	40	331
7	size_256-triple_2-unique	1648	3847	413	695
8	size_512-triple_2-unique	1224	2798	313	411
9	size_1024-triple_2-unique	945	1850	204	214

Legenda: "id": identificador do tipo de inserção.

Valle ressaltar que esse experimento não teve como objetivo avaliar a melhor abordagem, mas como cada variável e processo influência no resultado final. Dessa forma, pode ser observado na tabela 2 em relação aos cenários 1 e 2, a quantidade de nós e relações aumentam de forma linear com o aumento de documentos. Quando se avalia a diferença entre extração e inserção de vários documentos separadamente e em conjunto, percebe-se que a comparação entre os cenários 3 e 8 e entre os cenários 5 e 7, ocorre um aumento de aproximadamente 50% e 30% no número de nós, respectivamente. O mesmo pode ser observado para a relação do número de entidades. Isso indica que a extração de textos em conjunto tende a gerar um grafo mais conciso, o que pode demonstrar uma falha no método de extração de triplas por perder informações importantes quando é apresentada muita informação durante a extração. Por outro lado, isso pode demonstrar a deficiência no processo de adição de informações novas as já existentes no GC. Porém, ao se considerar o número de *labels* dos nós nessas mesmas situações acima, observa-se um crescimento maior nas versões *unique* (mais de 200%). Isto acontece, provavelmente, pois ao se extrair múltiplos documentos de forma isolada é possível informar à próxima extração sobre o que foi extraído anteriormente, melhorando a normalização e escolha de *labels*.

Ao avaliar a influência do tamanho dos segmentos no grupo de documentos inseridos individualmente (3, 5 e 6) com os *unique* (7, 8 e 9), observa-se que o comportamento se altera: no primeiro caso a redução do tamanho dos segmentos

resultou em um aumento dos nós e relações, enquanto no segundo grupo o aumento dos segmentos reduziu o número de nós e relações. Esse comportamento pode indicar, novamente, que o processo de inserção de novos conhecimentos pode estar negligenciando entidades iguais com *labels* diferentes, gerando um nó distinto para o mesmo conceito. Esse experimento também demonstrou que o tamanho do segmento de texto influencia no tamanho do grafo. Este é considerado um comportamento não desejável, visto que a informação é a mesma nas duas situações. Isso demonstra, também, uma deficiência no processo de normalização do conhecimento. Isto é observado de forma mais explícita quando os documentos são inseridos separadamente, visto as múltiplas inserções. Logo, quanto menor o tamanho dos segmentos, mais inserções acontecem por documento, aumentando o número de nós. Por outro lado, a extração e inserção do *dataset* como um conjunto gera um resultado mais próximo do esperado em que segmentos maiores levam à perda de informação e, conseqüentemente, menos triplas.

Quando os cenários 2 e 3 são comparados, é possível avaliar a influência da extração de entidades e relações (formando triplas) unidas ou em processos separados a partir do crescimento do número de entidades e relações no cenário 2 (9,7%) em que a extração acontece em junção. Isso indica que a separação das tarefas permite um maior controle do crescimento do GC. A extração da triplas em um passo apresentou uma redução na variação de nós (*labels* de nós), indicando uma menor variabilidade nas *labels* nesse caso, o que pode aumentar a consistência na definição das mesmas. Entretanto, observa-se um aumento considerável nos tipos de relações, visto que quando a extração da relação é realizada junto à entidade não é possível realizar o processo de limpeza e desambiguação dos nós antes da extração da relação e, por isso, ele tende a gerar mais triplas e mais relações.

Ao se avaliar o impacto da adição da taxonomia nos cenários 2 e 3, observa-se que a taxonomia extraída da *wikidata* gerou um aumento no número de nós em aproximadamente 20% ou uma média de cinco taxonomias para cada entidade extraída. Isso demonstra que as taxonomias se repetem em diversos nós, principalmente dentro de um mesmo domínio. Esse resultado provavelmente seria maior em uma base com informações de múltiplos domínios. Quanto ao número de *labels*, observou-se um aumento de 30% na versão com taxonomia, demonstrando

que a *wikidata* não auxilia na redução do número de *labels*. Isso ocorre, provavelmente, porque as taxonomias trazem mais de um novo termo por entidade (especificamente três no presente trabalho). Porém, observou-se que o número de relações aumentou em, aproximadamente, 800% devido à taxonomia, demonstrando que muitos nós de taxonomia estão relacionados à mais de uma entidade, resultando em novas relações e em um GC com informações mais conectadas.

A figura 37 demonstra que o tempo de extração e construção do grafo apresenta uma variação linear com o número de páginas nos documentos, ou seja, quanto maior é o número de páginas, mais lento se torna o processo (média de 32 segundos para o processamento por página). Além disso, verifica-se que grande parte do tempo por documento é ocupado pela extração das triplas, visto que a versão de extração de triplas apresentou um tempo de construção próximo a zero. Também é possível observar que a adição de taxonomias na *wikidata* ao processo de busca mais que dobra o tempo de extração médio. Isso ocorre pelo acréscimo de novos processos como, por exemplo, buscas na *wikidata*, processamento do resultado, desambiguação de entidades e atualização de nós e relações com as novas informações.

Durante a execução experimentos foi possível observar uma característica importante: a existência de mais de um nó com o mesmo nome, porém, com *labels* diferentes. A princípio, esse comportamento é considerado não é desejável, uma vez que indica que a mesma entidade existe em diversos pontos e uma insuficiência no processo de desambiguação e adição de conhecimento. Como consequência, a qualidade da informação para aquela entidade pode ser reduzida e o processo de busca de informação pode ser prejudicado. Entretanto, pode ser observado também que esses cenários normalmente aparecem em diferentes contextos, permitindo que um mesmo conceito exista em contextos diferentes, mesmo quando possuem o mesmo nome. Um exemplo dessa situação é a entidade “glicemia de jejum” encontrada no *dataset* diretrizes-diabetes e que aparece diversas vezes em contextos diferentes como, por exemplo, em procedimentos médicos, medidas de glicemia, medidas anatômicas e outros. Este processo permite a diferenciação entre palavras iguais que possuem significados diferentes. Um outro exemplo seria a palavra “banco” nas seguintes frases: 1) “Hoje eu vou ao banco pegar um empréstimo”; e 2) “Hoje eu passei a tarde sentado no banco da praça”. Na primeira frase a palavra “banco” se

refere a instituição financeira e na segunda a um móvel. Vale ressaltar que apesar das possíveis vantagens, esse comportamento deve ser controlado, uma vez que ele prejudica o crescimento organizado do GC ao utilizar essas *labels* como referência a cada nova extração. A fim de controlar o crescimento do GC, foi realizado um procedimento de combinação de nós que contenham *labels* e nome muito similares. Também foram combinados quaisquer dois nós que possuam a *url* de referência na *wikidata* idênticos.

Uma limitação do uso de GC que foi explicitada por este experimento é o tempo de processamento até que a informação esteja disponível para o agente executar buscas. Se comparado ao uso do RAG, o GraphRAG apresenta um aumento de 50 a 150 vezes neste tempo, dependendo do *pipeline* de construção utilizado. Este trabalho proe como solução a paralelização dos processos da construção do GC, em que, um processo é criado para cada segmento de texto e o processo de extração de triplas é realizado individualmente e ao final todas essas triplas são juntadas e inseridas no grafo. A eficiência desse método de extração é dependente da quantidade de processos que é possível rodar em paralelo, por exemplo, o processamento de mil páginas em um ambiente com 16 *threads* é possível reduzir em 93% o tempo de extração.

No entanto, esse método apresenta uma importante limitação, que é a dificuldade de compartilhamento de informação entre as extrações. Por exemplo, na extração sequencial cada nova extração reconhece as *labels* que foram extraídas na anterior e pode utilizá-las para padronizar as *labels* extraídas, o que não é possível quando as extrações acontecem paralelamente.

Este trabalho propõe mitigar este obstáculo ao adicionando um novo processo ao *pipeline*, em que ocorre a normalização de *labels* antes da inserção. Neste passo, todas as triplas extraídas passam por normalização das *labels* e tipos de relações. Essa normalização é realizada por um *match* de palavras utilizando distancia sintática e o resultado é submetido como uma lista de extrações a uma LLM para que ocorra esse merge de extrações iguais. Para avaliar esse resultado, foi extraído um documento de 30 páginas de forma sequencial e depois utilizando o método paralelo sem e com normalização em um ambiente com 16 *threads*. Os resultados demonstram que houve uma redução de 68% no tempo do processo paralelo para o sequencial,

que é menor que o teórica devido aos processos pré e post extração de triplas, que ainda são sequenciais. Além disso, houve um aumento de 34% na quantidade de entidades na versão paralela sem normalização, e 11% na versão com normalização, mostrando que o processo de normalização de fato reduz extrações repetidas, mas ainda não elimina o problema.

Outra importante consideração ao se utilizar um GC é o custo do processo de construção do mesmo. Para exemplificar, é apresentada a seguinte simulação que está contida na tabela 3: considerando-se uma média de 1000 *tokens* por página (média de *tokens* em uma página com texto contida em um formato de documento científico como o observado no dataset diretrizes-diabetes), 700 *tokens* do *prompt* (média dos prompts utilizado neste trabalho), uma média de 400 *tokens* de saída e o cenário hipotético em que é necessário realizar a inserção de 1000 documentos por dia, contendo 20 páginas cada (total de 20000 páginas processadas por dia), tem-se que o custo diário seria de 210 dólares (utilizando o GPT-4o. Disponível em: <<https://openai.com/api/pricing/>>, acesso em julho de 2024).

Tabela 3. Custo necessário utilização do GPT-4o por dia via API da OpenAI para o cenário de 20.000 páginas por dia.

Tipo de token	Valor (em dólares)	Tokens (valor base em dólares por milhão de tokens)	Total (em dólares por dia)
Entrada	34.000.000,00	5,00	170,00
Saída	8.000.000,00	15,00	40,00
Total	-----	-----	210,00

Este é um valor aproximado, pois não considera o custo de manutenção do banco de dados de grafo, custo de geração de *embedding* e nem inferências extras para geração de perguntas, traduções ou outros tipos de pré e pós processamentos adicionais utilizando modelos LLM.

Esse exemplo demonstra o alto custo dessa solução e este pode ser impeditivo para empresas menores ou que processam centenas ou milhares de arquivos por dia. Se comparado ao RAG, em que o único custo é do cálculo de *embedding*, o custo benefício da solução pode ser desfavorável a depender da necessidade da aplicação.

O principal componente do custo acima para a sua utilização é o valor por *token* das provedoras de API para inferência em modelos LLMs. Uma solução para esta limitação é a substituição da API de terceiros por um modelo LLM proprietário ou *open source*. Se considerarmos as máquinas em nuvem para a hospedagem desses modelos, esse custo pode cair substancialmente. Porém, isso traz diversos desafios, pois os modelos que podem ser executados não possuem a mesma capacidade e qualidade de geração dos modelos das provedoras, exatamente por serem menores. Para avaliar a viabilidade desta abordagem, um exemplo é apresentado: considerando o modelo Phi-3-mini-4k (ABDIN et al., 2024) que é centena de vezes menor que o GPT-4o e apresenta resultados de *benchmark* competitivos em relação a modelos de até 20 bilhões de parâmetros em tarefas específicas, é necessário realizar o *fine-tuning* do mesmo para que ele apresente resultados competitivos em relação aos modelos de escala maior (GPT-4o Gemini-1.5). Para realizar o *fine-tuning*, é necessário a disponibilidade de uma máquina com uma gpu A100 por 24 horas na *runpod.io* (disponível em: <https://www.runpod.io/pricing>. Acesso em julho de 2024), o que representa um custo aproximado de 15 dólares. Seguindo nesse exemplo, para que ocorra a execução do modelo em um tempo semelhante aos comparados, uma máquina com duas gpu A100 pode ser utilizada na *runpod.io* ao custo aproximado de 3,88 dólares por hora, resultando em 93,12 dólares ao dia. Ao compararmos ao custo inicial aproximado de 15 dólares, somado o custo recorrente de 93,12 dólares, o resultado é de uma redução de 225% no custo final quando comparado ao uso do GPT-4o e o único limitante seria na quantidade de processamento por dia e o tempo de processamento.

Dessa forma, pode-se observar que o emprego esta tecnologia ainda depende de um entendimento mais aprofundado dos processos e custos envolvidos, visto que os resultados, tanto em termos de custo, quanto de tempo e qualidade, são altamente afetados pelo número de *tokens* por segmento, número de chamadas por segmento, modelo utilizado, abordagem utilizada na extração e outros parâmetros que podem variar nesse processo. Assim, faz-se necessário a análise da necessidade e cenário de cada aplicação para decisão de qual é o cenário com maior custo benefício. Além disso, essa simulação demonstra os ganhos que o *fine-tuning* de um modelo LLM *open source* podem trazer, além dos benefícios de controle e confiabilidade do

processo, viabilizando aplicações complexas e que necessitam o uso intensivo de modelos LLM.

6.2 Contextualização do agente utilizando grafo de conhecimento – acesso a memória

Essa sessão apresentará os resultados dos experimentos realizados para avaliar a capacidade que a memória de longo prazo usando GC proporciona ao agente conversacional e como as respostas se comparam à abordagem de RAG. Para isso, os experimentos foram divididos entre avaliação humana e avaliação automática utilizando modelos LLM como juízes. Além disso, foi utilizado como método de construção dos grafos a abordagem 4 (size_512-triple_2-wikidata) do experimento anterior, visto que ela apresentou o melhor o resultado mais equilibrado e teoricamente mais completo.

6.2.1 Avaliação Humana

Para observar o impacto do GraphRAG nos problemas existentes no RAG, foram definidos cinco cenários relevantes em sistemas conversacionais, sendo eles:

- Cenário 1 – Fatos: teste com perguntas diretas que têm respostas factuais simples como, por exemplo, nas perguntas "Quem é o presidente dos EUA?" ou "Qual é a capital da França?". Esse cenário foi utilizado para verificar a precisão do sistema ao responder perguntas que requerem apenas a recuperação de informações factuais.
- Cenário 2 - Multihop: teste com perguntas que exigem a integração de várias informações de diferentes partes do texto ou de diferentes fontes para se chegar à resposta. Exemplo de pergunta nesse cenário: "Qual era o emprego do pai de Barack Obama?" (essa pergunta pode exigir a combinação de informações sobre Barack Obama e seu pai). Este é um cenário importante para avaliar a capacidade do sistema em conectar informações dispersas e em raciocinar através de múltiplas etapas.

- Cenário 3 – Inferência: teste com perguntas que não têm uma resposta explícita no texto, mas que podem ser inferidas a partir de outras informações como, por exemplo, na pergunta "Se o clima estava ensolarado, por que João levou um guarda-chuva?". Este cenário é utilizado para avaliar a habilidade do sistema em realizar inferências lógicas.
- Cenário 4 – Negativas: teste com perguntas que envolvem negações ou contradições como, por exemplo, na pergunta "Quem não é o presidente dos EUA?". Este é um teste necessário para verificar a capacidade do sistema em processar e buscar negações.
- Cenário 5 – Temporal: teste com perguntas que requerem a compreensão de eventos em uma linha do tempo como, por exemplo, "Quem era o presidente dos EUA antes de Barack Obama?". Esse teste é relevante para avaliar a compreensão temporal e a capacidade em ordenar eventos ou ações.

Para cada cenário apresentado acima foram geradas dez perguntas e suas respostas esperadas utilizando a base diretrizes-diabetes. Foi utilizado o Neo4J como banco de dados de grafo e dois índices de busca conectados diretamente ao GC através do Neo4J, sendo eles: Vector index (disponível em: <<https://neo4j.com/docs/cypher-manual/current/indexes/semantic-indexes/vector-indexes/>>) e FullText index [disponível em: <<https://neo4j.com/docs/cypher-manual/current/indexes/semantic-indexes/full-text-indexes/>>]. Esses índices permitem realizar uma busca semântica a partir da distância de *embeddings* e busca por *match* de texto. Cada pergunta foi respondida utilizando cinco diferentes estratégias de busca, apresentadas a seguir:

1. *Naive RAG* – realiza a busca vetorial utilizando distância cosseno entre o *embedding* da pergunta e os segmentos da base. Essa busca representa o RAG tradicional, sem uso de nenhuma informação relacional ou entidades. Esse método é a referência de comparação com os outros métodos de busca, visto que representa a forma mais tradicional e difundida de acessar a base de conhecimento.
2. *GraphRAG_global_segmento_entidades* – neste é realizado, primeiramente, a busca global utilizando vetor (Neo4J *vector index*) e texto (Neo4J *FullText*

index) e, em seguida, para cada segmento encontrado é realizado a busca em profundidade por todas as triplas relacionadas eles, ou seja, qualquer entidade que pertença ao segmento e que possua relação com outra entidade será retornada como um fato (entidade, relação, entidade). Tanto os segmentos, quanto a lista de triplas são retornados como contexto para o agente. O objetivo dessa busca é avaliar o impacto de se explicitar as entidades junto ao texto do qual elas foram retiradas, visto que isso pode auxiliar a ressaltar as informações relevantes e reduzir a influência de ruído. Além disso, esses fatos podem conter informações adicionais, visto que apenas uma das entidades da triplas deve pertencer ao segmento.

3. GraphRAG_global_facts - essa busca é idêntica à anterior, porém será retornado apenas a lista de triplas como contexto. O objetivo dela é avaliar se os fatos extraídos dos segmentos são suficientes para representá-los.
4. GraphRag_global_segment_discovery - busca global utilizando vetor (Neo4J *vector index*) e texto (Neo4J FullText *index*). Em seguida é realizada a descoberta de segmentos relacionados via entidades, ou seja, se existe um segmento não descoberto pela busca global, mas que possui um caminho pela camada de entidades até um dos segmentos descobertos, ele é então retornado. Esses novos segmentos são ranqueados pelo tamanho do caminho, e pela quantidade de caminhos. Dessa forma, quanto menor o caminho e maior a quantidade de caminhos, maior é a chance de os segmentos possuírem uma alta relação semântica. Essa busca tem como objetivo avaliar o quanto as relações podem contribuir para descoberta de segmentos importantes para a *query*, mas que não possuem alta similaridade com a pergunta. Este é um problema do RAG vetorial e que é essencial para responder perguntas *multihop*.
5. GraphRag_local_facts - busca local utilizando entidades como ponto de entrada seguida de busca em profundidade (profundidade 3 para alcançar a camada de taxonomia). A fim de contemplar o maior número de variações da entidade buscada, foi realizada a extração dos termos chave da *query* e, então, gerada uma lista de variações das desses termos como, por exemplo, plural, variação de gênero e diferentes inflexões linguísticas para reduzir a variação

de escrita existente no CG ao se comparar ao termo na *query*. Esse processo foi realizado em uma única inferência no modelo LLM. Essa busca foi realizada para avaliar a real necessidade de armazenamento de todo o documento na base e se o método de extração proposto é capaz de extrair toda a informação relevante do texto em forma de entidades e relações.

6. *Query2Cypher* – este método foi utilizado para garantir liberdade ao agente na busca que for necessária para responder a pergunta (*query do usuário*) e, para isso, foi gerada uma *cypher query* dado a *query* do usuário (NEO4J CYPHER, 2020). Para melhor informar ao agente sobre a estrutura do GC, é informado um *schema* simplificado do GC, em que apenas as *labels* que definem cada nível de abstração são informadas (por exemplo: *documents*, *elements*, *entity*, *taxonomy*, *user* e outros). Isso se fez necessário devido ao tamanho do *schema* ser superior ao limite de entrada do modelo LLM. Além disso, foi realizado o mesmo processo da busca anterior em que é requisitado que a *query* contenha variações dos termos para cobrir as variações de termos. Esse processo adiciona um processo na busca (uma chamada ao modelo LLM para criação da *Cypher query*) e, por isso, aumenta o tempo total da busca, que por sua vez pode variar drasticamente dependendo da *query* gerada.

As respostas foram avaliadas manualmente por três pessoas de diferentes áreas (saúde, engenharia e direito) e para cada avaliador foram dadas todas as perguntas geradas, respostas esperadas e a resposta gerada em cada tipo de busca apresentado anteriormente. Foi então requisitado aos avaliadores que dessem uma nota de zero a 10 para cada resposta, sendo 10 uma resposta completa para a pergunta que estivesse de acordo com a resposta esperada e zero para uma resposta que não atende à pergunta realizada e nem condiz com a resposta esperada. A tabela 4 apresenta o resultado da nota média (multiplicada por 100).

Outro relevante resultado para o uso dessa proposta em aplicações reais é o tempo de resposta, também apresentado na tabela 4. Vale ressaltar que o Neo4J possui um sistema de cache em que a primeira busca é mais lenta que as subsequentes, podendo ter uma redução de mais de 200% da primeira para as

subsequentes. Dessa forma, o tempo considerado nesse experimento foi o do pior cenário em que cada *query* está sendo realizada pela primeira vez.

Tabela 4. Avaliação humana do resultado das buscas em diferentes cenários utilizando os dados do dataset diretrizes-diabete.

Tipo de busca	Nota Média Cenário 1 Fatos	Nota Média Cenário 2 <i>Multihop</i>	Nota Média Cenário 3 Inferência	Nota Média Cenário 4 Negação	Nota Média Cenário 5 Temporal	Nota Média Geral	Tempo médio (m)
1*	77	36	49	45	35	48,4	200
2	89	53	58	58	45	60,6	403
3	75	48	46	52	48	53,8	438
4	89	68	57	48	44	60,2	456
5	76	52	50	57	51	57,2	347
6	81	56	55	54	49	58,8	1152

Legenda: ms: milissegundos.

*O cenário 1 é a referência, representando a metodologia de busca de informação baseada em busca vetorial.

O principal resultado desse experimento é que todas as buscas realizadas por meio de GC (GraphRAG) performaram melhor do que utilizando apenas busca vetorial (RAG), demonstrando a capacidade da proposta deste trabalho em gerar um contexto mais rico para o agente. Destaca-se a melhoria em perguntas mais complexas que exigem a relação entre informações como multihop, inferência e temporal. Comparando-se a busca 2 com a 3, observa-se que os resultados da busca 2 foram superiores (média) ao da busca 3 e que esta foi inferior no cenário de busca temporal. Isso ocorreu, provavelmente, pela alta capacidade de extração de fatos temporais do GraphRAG, o que demonstra que os fatos não representam todo o conhecimento contido nos segmentos e que ainda há espaço para melhorias no processo de extração. Por outro lado, quando se observa a busca 5 em que somente são utilizadas as triplas como contexto, o resultado foi superior a busca 3. Isso se deve, provavelmente, pela profundidade da busca 5 e pela capacidade de encontrar mais informações relacionadas.

A camada de taxonomia apresentou um papel importante na conexão de informações, principalmente em cenários que necessitam de conhecimentos mais

gerais como o cenário de inferência (3) e negação (4). Sendo esse método de conectar conceitos, a taxonomia foi essencial para a qualidade da busca 4, onde a descoberta de novos segmentos necessita de um caminho que os conecte. O resultado da busca 4 no cenário *multihop* fortalece esse entendimento, apresentando resultado superior as outras técnicas.

As buscas 2 e 4 apresentaram os melhores resultados, sendo estas, buscas que apresentam uma abordagem de busca híbrida, onde é utilizado tanto segmentos, quanto triplas no processo. Esse resultado superior se estende para vários cenários, demonstrando que a principal vantagem do GC nesse contexto é a capacidade de relacionar informações.

A busca 5 apresentou resultados próximos às melhores buscas em todos os cenários, sendo melhor que a busca 1 e a busca 3. Isso indica a possibilidade do uso de uma base contendo apenas o conhecimento extraído dos segmentos (triplas), permitindo uma redução no tamanho da base, assim como a redução do tempo de busca e custo de construção, visto que não foram utilizados *embeddings* e índices auxiliares nessa busca. Porém, no cenário de inferência (3), que se faz necessário a busca de mais informações para que o agente pudesse inferir algo que não está explícito, as buscas 3 e 5 apresentaram resultados inferiores as buscas que retornam segmentos. Isso demonstra que ainda existe uma dificuldade em representar todo o conhecimento dos segmentos através de entidades e relações.

Os resultados das buscas utilizando as *queries* geradas pelo agente (busca 6), apresentaram resultados superiores, na média, às buscas 1, 3 e 5. Essa abordagem demonstrou melhorias mais expressivas nos cenários mais complexos em que as outras buscas propostas podem não ser suficientes para obter toda informação necessária. Por outro lado, no cenário 1, que apresenta perguntas mais simples e diretas, o resultado dessas buscas foi inferior, demonstrando uma tendência do agente em gerar *queries* complexas mesmo quando o cenário não exige e, consequentemente, gerar contextos com informação desnecessária.

Ao se avaliar o tempo na busca 6, foi observado uma grande variação que depende da *query* gerada devido à complexidade e otimização do gerador. Porém, grande parte do acréscimo de tempo apresentado na tabela 4 é decorrente da inferência extra no modelo LLM para geração da *query*. Isso é corroborado pelo fato

da busca 5, em que também é realizada uma inferência a mais no modelo LLM, ocorre um considerável acréscimo de tempo, dado que o tempo médio da aplicação da *query* na busca 5 é de 100 segundos.

Vale ressaltar, também, que as *cypher queries* geradas nem sempre são válidas para transpor este problema. Para essa situação foi utilizado um processo de validação e, caso a *query* seja inválida, seja realizada uma correção utilizando um modelo LLM. Se a tentativa de correção ainda retornar uma *query* incorreta, esta é então marcada como incorreta (nota 0). Esse problema pode ser mitigado através da utilização de *fine-tuning* de um modelo especialista em *cypher* e de melhores estratégias para informar a estrutura do GC ao modelo.

6.2.2 Avaliação baseada em LLM

Inspirado no RAGAS (ES et al., 2023), foram propostos seis métricas de avaliação baseadas em LLM para avaliar MemoryGraph. Estas foram divididas em métricas de *retriever* e *renerator* e cada uma respondeu às seguintes perguntas:

- 1) *Context relevancy*: "Os documentos são relevantes para a pergunta?"
- 2) *Context precision*: "O quanto os documentos/contexto são alinhados com a resposta?"
- 3) *Context recall*: "Todos os fatos relevantes, referentes à resposta, estão classificados em posição mais alta ou não?"
- 4) *Answer correctness*: "A resposta é consistente com uma resposta de referência?"
- 5) *Answer faithfulness*: "A resposta está fundamentada nos documentos ou houve alucinação?"
- 6) *Answer helpfulness*: "A resposta ajuda a responder à pergunta?"

Essas métricas foram aplicadas em três datasets: diretrizes-diabetes, *Natural Questions* (KWIATKOWSKI et al., 2019) e *RAGBench* (FRIEL; BELYI; SANYAL, 2024). Os dois últimos datasets são populares na comunidade de RAG e, juntos, cobrem uma grande gama de cenários. Porém, devido a limitação de tempo e custo

para execução de mais de 300 mil *samples* do *Natural Questions* e 100 mil do *RAGBench*, foram utilizados 10% de cada *dataset* amostrado de forma aleatória. Em relação ao *dataset* diretrizes-diabetes, foram criadas 100 perguntas com contexto variando entre 3 a 8 sentenças extraídas dos 40 documentos que o compõem, além de uma resposta esperada para cada pergunta. O objetivo foi criar um *dataset* com diferentes cenários como, por exemplo, perguntas que necessitam de relação entre diferentes partes do texto, necessidade de sumarização de informações dispersas pelo *dataset*, checagem de fatos e extração de entidades e informações temporais.

Para a execução os experimentos, cada *dataset* foi inserido em um GC isolado, de forma que cada amostra fosse processada como um elemento de texto e inserido no grafo. No caso do *Natural Questions*, a página *wiki* referente a cada amostra foi processada e inserida no GC e a *long answer* foi utilizada como contexto para as métricas de *retriever*. Para cada GC foi submetido, de forma isolada, as perguntas do *dataset* para o agente em dois cenários distintos: utilizando RAG e utilizando MemoryGraph. Enquanto no RAG foi utilizado uma busca vetorial com K de 10, no MemoryGraph foi utilizado uma estratégia híbrida das melhores buscas apresentadas anteriormente, focando na melhoria através de métodos híbridos de busca. Essa busca consiste em: realizar uma busca vetorial pelos cinco melhores segmentos e então realizar a descoberta de mais cinco segmentos através de uma busca em profundidade. Dessa forma, são retornados os próprios segmentos e todas as triplas contidas no caminho entre eles. A tabela 5 demonstra os resultados desses experimentos a partir das seis métricas utilizadas.

Tabela 5. RAG vs MemoryGraph nos datasets diretrizes-diabetes, RAGBench e *Natural Questions*.

RAG Módulo	Métricas	diretrizes-diabetes		RAGBench		Natural Questions	
		RAG	Memory Graph	RAG	Graph RAG	RAG	Memory Graph
Retriever	<i>Context relevancy</i>	82	76	60	65	75	72
	<i>Context precision</i>	69	75	54	64	62	71
	<i>Context recall</i>	80	82	54	56	68	73
Generator	<i>Answer correctness</i>	81	91	68	72	76	86
	<i>Answer faithfulness</i>	64	86	65	76	52	72
	<i>Answer helpfulness</i>	68	78	62	69	84	86

Observa-se na tabela 5 que o MemoryGraph demonstrou ser superior ao RAG convencional em quase todas as métricas, apresentando maiores diferenças entre as duas abordagens no *dataset* diretrizes-diabetes e menor diferença no RAGBench. Esse resultado pode ser explicado pelo fato do primeiro *dataset* ser de domínio específico, resultando em um GC mais conciso (menor variações nas *labels* de nós e tipos de relações) e, conseqüentemente, reduzindo a dificuldade das buscas. Por outro lado, os resultados foram melhores no *dataset* *Natural Questions*, mesmo sendo o maior *dataset* e de domínio geral. Isso pode ser decorrente do fato que os modelos LLMs possuem maior conhecimento sobre informações de domínio geral, equilibrando as falhas da busca. Observa-se, também, que os resultados nas métricas da classe generator no *dataset* *Natural Questions* apresentaram uma diferença mais expressiva entre as abordagens do que nos outros *datasets*, corroborando os resultados do experimento anterior em que o MemoryGraph demonstrou melhor capacidade em buscar informações espalhadas pela base de conhecimento.

A única métrica em que o RAG foi superior foi a *context relevancy* em dois *datasets*. Esse resultado ocorre, provavelmente, porque o MemoryGraph busca mais informações relacionadas aos segmentos iniciais da busca semântica que, apesar de relacionados, nem sempre são relevantes para a pergunta. Por outro lado, no *dataset* mais complexo (*RAGBench*), o MemoryGraph apresentou resultados melhores nesta métrica, indicando uma deficiência da busca vetorial em encontrar informações mais complexas. Vale ressaltar que, mesmo quando o *context relevancy* do MemoryGraph foi inferior, o *answer correctness* foi superior, indicando que esse conteúdo extra, julgado como irrelevante, pode ajudar a gerar um contexto mais rico e melhorar o entendimento do agente sobre as informações que são realmente importantes para a geração da resposta.

O MemoryGraph apresentou resultado superior na métrica *answer faithfulness* (38% no melhor caso), resultado que pode ser justificado pela limitação do RAG em perguntas *multihop*, resultando em respostas incompletas ou contendo alucinações. A diferença na métrica *answer helpfulness* pode ser relacionada a mesma característica, demonstrando que as respostas geradas a partir do contexto retornado pelo MemoryGraph são mais completas. Esse comportamento pode ser mais evidenciado em *datasets* de domínio abertos (por exemplo, o *Natural Questions*), pois o modelo LLM tende a conter informações desse tipo em seu conhecimento paramétrico e pode utilizá-lo erroneamente para complementar a resposta.

O MemoryGraph apresentou uma melhora mais expressiva nas métricas de *retriever*, indicando o aumento na qualidade da informação extraída e, conseqüentemente, a entrega de um contexto mais rico para o agente gerar uma resposta.

Uma importante característica do uso de LLM como juiz é, além do valor quantitativo através da nota, a produção de uma justificativa para tal, permitindo, assim, uma maior transparência nos critérios de cada avaliação e um melhor entendimento sobre a qualidade do contexto retornado da base de conhecimento e das respostas geradas pelo agente, o que fornece importantes percepções para futuras melhorias dos mesmos. A tabela 6 apresenta um exemplo de pergunta e resposta gerada utilizando RAG e MemoryGraph. Observa-se que o MemoryGraph traz mais informações, como constatado nos experimentos anteriores. Essas

informações adicionais são, principalmente, fatos e dados relacionados ao assunto contido na pergunta como no exemplo “diabetes em países em desenvolvimento” da tabela 6. Isso demonstra, assim, a capacidade do MemoryGraph em buscar informações relacionadas entre si que não necessariamente possuem uma alta similaridade com a *query*. Essa capacidade de avaliação pode ser utilizada como um componente de auto crítica do agente, em que ele consegue avaliar a qualidade de cada processo interno e tomar decisões diferentes, como refazer a busca com uma query alterada, ou utilizar uma busca mais completa para buscar mais informações, ou regerar a resposta de forma mais completa.

Vale ressaltar por este exemplo que, por mais que a resposta gerada pelo RAG não possua uma informação importante para a pergunta (“5% e 20% do gasto total com saúde dos países desenvolvidos”), o modelo LLM descontou apenas 0,05 da nota final, o que pode indicar uma deficiência no critério de penalização por falta de informações e, conseqüentemente, pode indicar que os resultados poderiam apresentar uma discrepância ainda maior se estes critérios estivessem mais ajustados. Este experimento demonstra como as características do MemoryGraph influenciam os principais componentes da tarefa de responder uma pergunta utilizando como contexto uma base de conhecimento. Além disso, demonstra a capacidade que o GC traz para o agente, não somente na representação das informações advindas de documentos, mas também como o conhecimento extraído destes pode contribuir para a melhora no desempenho do agente.

Tabela 6. Exemplo de pergunta com a resposta esperada e a avaliação da métrica Answer Correctness para a execução com RAG e MemoryGraph

Pergunta	"Quais são os impactos econômicos do diabetes em países em desenvolvimento?"
Resposta esperada	"O diabetes representa uma carga econômica significativa tanto nos custos diretos para os sistemas de saúde e sociedade, quanto nos custos indiretos atribuíveis à mortalidade prematura e incapacitações temporárias e permanentes, consumindo entre 5% e 20% do gasto total com saúde dos países desenvolvidos."
Resposta gerada utilizando Memory Graph	<p>"Os impactos econômicos do diabetes em países em desenvolvimento são substanciais e multifacetados. Em termos diretos, o diabetes impõe uma carga financeira significativa tanto nos sistemas de saúde quanto nos indivíduos afetados e suas famílias, devido aos custos com insulina, antidiabéticos orais, e outros medicamentos essenciais. Indiretamente, o diabetes também contribui para a perda de produtividade, incapacitações temporárias e permanentes, e mortalidade prematura, aumentando ainda mais os custos econômicos."</p> <p>Além disso, o elevado número de casos não diagnosticados, que representa cerca de 84,3% em países em desenvolvimento, agrava a situação, pois a falta de detecção precoce permite o avanço de complicações graves como insuficiência renal, cegueira, problemas cardíacos, e o pé diabético. Essas complicações exigem cuidados prolongados e intensivos, resultando em maior utilização dos serviços de saúde e aumentando os gastos dos sistemas públicos de saúde.</p> <p>Em muitos países em desenvolvimento, o diabetes consome entre 5% e 20% do gasto total com saúde, o que evidencia a considerável pressão que a doença coloca sobre os recursos limitados desses sistemas."</p>
Nota MemoryGraph	1,0
Justificativa	"A resposta do estudante está correta em relação à resposta padrão. O estudante mencionou tanto os custos diretos (insulina, antidiabéticos orais, medicamentos essenciais) quanto os custos indiretos (perda de produtividade, incapacitações temporárias e permanentes, mortalidade prematura), os quais são abordados na resposta padrão. Além disso, o estudante forneceu informações adicionais sobre a alta porcentagem de casos não diagnosticados e as complicações associadas, juntamente com dados sobre o impacto percentual nos gastos com saúde, que são informações factuais precisas que não estão em contradição com a resposta padrão. Portanto, a resposta está completa e não contém declarações conflitantes, justificando a pontuação máxima."
Resposta gerada utilizando RAG	"Os impactos econômicos do diabetes em países em desenvolvimento são significativos e multifacetados. Eles incluem tanto os custos diretos, como despesas com tratamentos, medicamentos (como insulina e antidiabéticos orais), e utilização intensiva dos serviços de saúde, quanto os custos indiretos, como perda de produtividade devido à mortalidade prematura e incapacitações temporárias ou permanentes causadas pelas complicações da doença. Além disso, o diabetes impõe uma carga financeira considerável sobre os indivíduos e suas famílias, especialmente em contextos onde o acesso a cuidados de saúde pode ser limitado ou oneroso. Esses impactos são exacerbados pela falta de conscientização adequada sobre a gravidade da doença, tanto por parte dos governos quanto dos profissionais de saúde, o que impede a implementação de políticas e práticas eficazes de prevenção e manejo do diabetes e suas complicações crônicas, como insuficiência renal, cegueira, problemas cardíacos e pé diabético."
Nota RAG	0,95
Justificativa	"A resposta do aluno abrange de forma ampla e detalhada os impactos econômicos do diabetes em países em desenvolvimento, mencionando tanto custos diretos quanto indiretos, conforme exigido. Além disso, inclui pontos adicionais corretos e relevantes, como a carga financeira sobre os indivíduos e famílias e desafios na conscientização e políticas públicas. No entanto, a resposta do estudante não menciona explicitamente a faixa percentual de 5% a 20% do gasto total com saúde como na resposta padrão. Essa omissão é um pequeno detalhe, mas reduz ligeiramente a completude em relação à resposta esperada."

6.3 *user_memory* no contexto do agente

Para avaliar o conceito do *user_memory* foi necessária a elaboração de um conjunto de interações entre um usuário e o agente. Para isso, foi gerado um *dataset* simulando um usuário fictício chamado Maria que possui 35 anos e a patologia crônica diabetes como condição de saúde (exemplos dessa interação estão contidos na figura 47 e 48). Nessa situação, o agente é um assistente capaz de auxiliar em tarefas diárias, tirar dúvidas, auxiliar nos estudos, auxiliar com lembretes, alimentação e com sua condição em saúde. Foram criados alguns cenários de diálogo em que o mesmo assunto é abordado novamente, mas com o usuário em um estado diferente em que ele fala sobre consequências de sugestões e conversas anteriores. Isso permitiu validar a capacidade do sistema em atualizar informações. Esse *dataset*, chamado de Assistente-da-Maria, possui 50 conversas em que cada uma há de três a 12 interações, em português brasileiro (PT-BR) e considera a interação apenas com um usuário, chamada aqui de Maria.

O processo de extração do *user_memory* foi executado e salvo no GC a cada interação no *dataset*. Depois das 50 interações foram realizadas 20 perguntas para o agente para avaliar a retenção do conhecimento sobre o usuário. Estas perguntas foram divididas entre 10 perguntas simples (cujas respostas poderiam ser encontradas diretamente nas conversas) e 10 perguntas complexas (cujas respostas dependeriam do relacionamento de informações contidas em diferentes conversas). Todas as respostas esperadas foram geradas utilizando o conhecimento de todas as conversas do *dataset*, enquanto o agente utilizou apenas a memória extraída das conversas (*user_memory*) para responder as perguntas apresentadas.

A avaliação de cada resposta poderia ter como resultado: correto (valor 1 e ocorre quando 100% da informação da resposta esperada está contida na resposta gerada; parcial (valor 0,5 e ocorre quando parte substancial da informação - aproximadamente 50% - contida na resposta gerada está presente na resposta esperada); e incorreta (valor 0 e ocorre quando a totalidade ou a maioria da resposta esperada não está contida na resposta gerada). Dessa forma, foi possível calcular a retenção do conhecimento do agente, definida pela divisão da soma do valor atribuído a cada resposta, por um humano, dividido pelo número de respostas respondidas pelo agente.

Observa-se pelos resultados apresentados nas tabelas 7 e 8 que o agente é capaz de substituir informações antigas ou eliminá-las quando não são mais necessárias ou pertinente naquele momento como, por exemplo, em uma dada conversa, a Maria diz que tem dificuldades com frações matemáticas, o agente então adiciona este fato à memória. Porém, em conversas futuras, Maria diz ter resolvido seu problema com frações. O agente, então, entende que essa não é mais uma informação relevante para a caracterização do usuário e retira essa informação da sua representação em memória. Vale ressaltar, que a informação não foi perdida, ela ainda pode ser acessada através do resumo da conversa onde essa informação foi obtida, não sendo uma característica relevante para a caracterização daquele usuário.

Tabela 7. Resultado das perguntas simples realizadas utilizando *user_memory*.

Pergunta	Resposta esperada	Resposta gerada	Resultado
Qual exercício Maria começou a praticar e que a ajudou a se sentir mais relaxada?	Yoga.	Yoga	Correto
Quais são os legumes que Maria mais gosta de incluir em suas refeições?	Cenoura e brócolis.	Cenoura e brócolis.	Correto
Que tipo de podcasts Maria gosta de ouvir durante suas caminhadas?	Podcasts sobre história e entrevistas inspiradoras.	Podcasts sobre história e entrevistas inspiradoras.	Correto
Qual é a preferência de Maria em relação ao horário para fazer exercícios?	No final da tarde.	Maria prefere fazer exercícios no final da tarde.	Correto
Qual comida Maria realmente não gosta e prefere evitar?	Beterraba	Beterraba	Correto
Qual foi a combinação de refeição favorita mencionada por Maria?	Salmão grelhado com cenouras e brócolis.	Salmão grelhado com cenouras e brócolis.	Correto
Qual ferramenta de gestão de tarefas o agente sugeriu para ajudar Maria a organizar suas tarefas?	Todoist ou Trello.	Aplicativos de gestão de tarefas como Todoist ou Trello.	Correto
Quem é a mãe de Maria?	Ana	Ana	Correto
Qual tipo de música Maria gosta de ouvir para relaxar?	Piano clássico, como Chopin e Beethoven.	Música clássica, especialmente piano de Chopin e Beethoven	Correto
Qual foi a receita de frango que Maria experimentou e gostou?	Frango ao curry.	Frango ao curry.	Correto

Tabela 8. Resultado das perguntas complexas realizadas utilizando *user_memory* (continua).

Pergunta	Resposta esperada	Resposta gerada	Resultado
Como a prática de yoga, que Maria começou a fazer regularmente, influenciou não apenas seus níveis de açúcar no sangue, mas também sua produtividade no trabalho?	A prática de yoga ajudou a estabilizar os níveis de açúcar no sangue de Maria e a deixá-la mais calma e focada, aumentando sua produtividade no trabalho.	A prática de yoga ajudou Maria a estabilizar seus níveis de açúcar no sangue e, ao mesmo tempo, contribuiu para uma maior produtividade no trabalho. Ela se sentiu mais relaxada e focada, o que a ajudou a lidar melhor com suas tarefas diárias e a reduzir o estresse.	Parcial
Qual foi a reação de Maria à sugestão de experimentar novas combinações de temperos na receita de espinafre, e como isso se relaciona com sua experiência anterior de não ter gostado da receita?	Maria ficou animada com a sugestão, pois reconheceu que a receita anterior estava sem graça. Ela está aberta a novas combinações e espera que a próxima tentativa seja melhor.	Maria está aberta a experimentar novas combinações de temperos e especiarias para melhorar suas receitas. Isso está relacionado à sua experiência anterior com a receita de espinafre, que ela não gostou porque estava sem graça. A disposição dela em experimentar novos temperos mostra seu desejo de aprimorar suas refeições.	Correto
Quais dificuldades Maria mencionou ao usar o Trello, e como a sugestão do agente de definir um horário fixo para revisar tarefas poderia ajudá-la a superar essas dificuldades??	Maria mencionou que se distraía e não atualizava as tarefas. A sugestão de definir um horário fixo ajudaria a criar uma rotina que a manteria mais focada e organizada.	Maria mencionou que às vezes se distrai e não atualiza as tarefas no Trello. Definir um horário fixo para revisar suas tarefas pode ajudá-la a criar uma rotina mais estruturada, o que pode minimizar as distrações e garantir que as tarefas sejam sempre atualizadas e monitoradas.	Correto
Como a aceitação da ajuda de sua mãe, Ana, impactou a sensação de sobrecarga que Maria tinha em sua rotina, e que responsabilidades ela decidiu compartilhar?	Aceitar a ajuda de sua mãe, Ana, aliviou a sobrecarga de Maria, permitindo que ela compartilhasse algumas responsabilidades de organização e tarefas domésticas.	A aceitação da ajuda de sua mãe, Ana, teve um impacto positivo na sensação de sobrecarga que Maria estava enfrentando. Ela decidiu compartilhar algumas responsabilidades relacionadas à organização em casa, o que a fez sentir-se menos sobrecarregada e mais apoiada.	Correto
Quais ajustes Maria fez nos horários dos lembretes de medicação, e como isso afetou sua experiência geral em gerenciar sua saúde?	Maria ajustou os lembretes para um horário mais tarde, quando está mais relaxada. Isso facilitou a gestão dos medicamentos e a fez sentir-se menos apressada.	Maria ajustou os lembretes de medicação para horários mais tarde, quando está mais relaxada. Isso afetou positivamente sua experiência em gerenciar sua saúde, tornando mais fácil para ela lembrar de tomar a insulina e medir os níveis de glicose.	Correto
De que forma o novo podcast sobre história que Maria começou a ouvir melhorou suas caminhadas e como isso se relaciona com sua luta contra o tédio?	O novo podcast tornou as caminhadas mais interessantes e envolventes, ajudando a combater o tédio que ela sentia antes ao ouvir os mesmos podcasts repetidamente.	O novo podcast sobre história que Maria começou a ouvir tornou suas caminhadas mais interessantes e divertidas. Isso ajudou a combater o tédio que ela sentia durante as caminhadas, proporcionando uma distração agradável e enriquecedora.	Correto

Tabela 8. pergunta / resposta / resultado / comentário sobre resposta (conclusão).

Pergunta	Resposta esperada	Resposta gerada	Resultado
Como a experiência de Maria com as pausas regulares nos estudos influenciou sua capacidade de concentração, e que nova estratégia ela decidiu implementar para melhorar ainda mais sua rotina de estudos?	As pausas ajudaram Maria a manter a concentração. Ela decidiu experimentar estudar em sessões de 25 minutos para otimizar seu foco e reduzir a fadiga.	A experiência de Maria com pausas regulares durante os estudos melhorou sua capacidade de concentração. Ela decidiu implementar sessões de estudo mais curtas, com intervalos, para aumentar ainda mais sua eficácia e reduzir a exaustão.	Parcial
Qual foi a diferença que Maria notou em sua receita de frango ao curry quando ela decidiu ajustá-la e como isso se relaciona com sua intenção de aperfeiçoar suas habilidades culinárias?	Maria notou que a receita ficou melhor, mas ainda precisa ajustar a quantidade de sal. Isso a motivou a continuar aprimorando suas habilidades culinárias.	Maria notou que ao ajustar a receita de frango ao curry, o prato ficou mais saboroso. Essa experiência está alinhada com sua intenção de aperfeiçoar suas habilidades culinárias, buscando melhorar suas receitas e aumentar a variedade em sua dieta.	Parcial
Como a organização do espaço de trabalho de Maria contribuiu para seu aumento de produtividade e que mudanças ela notou em sua rotina diária após essa reorganização?	A organização do espaço de trabalho fez com que Maria encontrasse tudo mais rapidamente e se distraísse menos, resultando em uma rotina mais produtiva.	A organização do espaço de trabalho de Maria contribuiu para um aumento em sua produtividade. Ela notou que consegue encontrar tudo mais rápido e não se distrai tanto, o que melhorou sua rotina diária e a ajudou a lidar melhor com suas tarefas.	Correto
Quais foram os benefícios que Maria experimentou em sua saúde e bem-estar após ajustar sua rotina de exercícios para o final da tarde, e como isso se relaciona com os lembretes que o agente ajudou a programar?	Ajustar os exercícios para o final da tarde fez Maria se sentir mais ativa e melhorou a estabilização dos níveis de açúcar no sangue, tornando os lembretes mais eficazes na sua rotina.	Após ajustar sua rotina de exercícios para o final da tarde, Maria experimentou benefícios em sua saúde e bem-estar, como níveis de açúcar mais estáveis e uma sensação de maior energia. Isso se relaciona com os lembretes que foram programados para ajudá-la a manter essa prática regular de exercícios.	Correto

Os resultados do experimento mostram uma retenção de conhecimento de 92% para essas 20 perguntas realizadas em cima das 50 conversas, um resultado promissor. Além disso, nas perguntas mais complexas, alguns casos as respostas com memória foram consideradas mais completa que a resposta com acesso a todas as conversas. Isso se dá, provavelmente, pela redução de ruídos que a memória trás, apresentando apenas as informações relevantes sobre o usuário.

Uma consideração importante a ser realizada sobre a memória, é em relação ao seu tamanho. Da forma como a extração foi realizada, houve um grande consumo de memória pelos *tokens* do prompt. Por exemplo, ao final das 50 conversas do agente com a Maria, sua memória possuía 2050 *tokens* e 9217 caracteres. Na tentativa de reduzir o tamanho dessa memória, inicialmente o processo foi ajustado para manutenção de apenas informações chave sobre o usuário e para descartar fatos temporais, além de tentar agrupar informações similares. Além disso, ao carregar essas informações para o agente, um novo processo de simplificação e resumo dos fatos foi adicionado através de LLM, permitindo a redução desse mesmo *user_memory* para 493 *tokens* 1817 caracteres. Para avaliar o impacto da compactação da informação, o experimento anterior foi repetido, resultando em uma retenção de informação de 70% no total e de 100% no conjunto de perguntas consideradas simples. No conjunto perguntas complexas, que necessitavam de relação entre informações temporais e mais granulares, a retenção foi 40%. Isso demonstra que há perdas em decorrência da compactação da informação. Porém, isto pode ser uma solução para sistemas que possuem alta limitação no tamanho do *prompt* do modelo LLM do agente.

A figura 38 apresenta um exemplo de conversa entre o paciente fictício e o agente sem a utilização de *user_memory* enquanto a figura 39 apresenta o resultado da mesma conversa utilizando *user_memory* extraída das primeiras 30 primeiras conversas do agente com o paciente fictício.

Figura 38. Exemplo de conversa entre o paciente fictício e o agente sem a utilização de *user_memory*.

Usuário: Oi, eu estou me sentindo um pouco tonta, sera o que pode ser?

Agent: Sinto muito que você esteja se sentindo assim. Tontura pode ter várias causas, como desidratação, queda de pressão, falta de sono ou até algo mais sério. Você tem algum outro sintoma? Já tentou beber água ou descansar um pouco?

Usuário: Ja, fiz o de sempre

Agent: Entendi. Se a tontura persistir, é importante procurar um médico para uma avaliação mais detalhada. Pode ser algo simples, mas é sempre bom ter certeza. Se precisar de mais alguma coisa, estou aqui para ajudar.

Usuário: vou tentar medir minha glicose

Agent: Boa ideia! Verificar a glicose pode ajudar a identificar se há alguma alteração que possa estar causando a tontura. Se os níveis estiverem fora do normal, procure orientação médica. Fique bem!

Usuário: Falando em se sentir melhor, como era o nome do aplicativo para gerenciamento de tarefas que voce sugeriu mesmo?

Agent: Eu não sugeri nenhum aplicativo específico antes, mas alguns populares para gerenciamento de tarefas são Todoist, Trello e Microsoft To Do. Algum desses te interessa?

Usuário: so o trello

Agent: Ótimo! O Trello é uma ferramenta excelente para organizar tarefas e projetos. Ele usa um sistema de quadros e cartões que é bem intuitivo. Se precisar de ajuda para começar a usar, me avise!

Usuário: Pensando em criar numa ferramenta uma lista daquelas receitas q ja fiz e que gostei. aceito mais sugestões inclusive.

Agent: Que ideia bacana! Além do Trello, você pode considerar o Evernote ou o Notion. Ambos são ótimos para organizar informações e permitem adicionar notas, imagens e links. Você pode criar uma seção específica para suas receitas favoritas e até categorizá-las por tipo de prato ou ingredientes. Boa sorte com sua organização!

Usuário: Vou tentar criar amanha de manha entao com aquelas comidas que ja fiz

Agent: Perfeito! Boa sorte com a criação da sua lista. Se precisar de ajuda ou tiver dúvidas durante o processo, estou aqui para ajudar. Aproveite e divirta-se organizando suas receitas!

miro

Fonte: próprio autor.

Figura 39. Exemplo da conversa descrita na figura 38 utilizando *user_memory* extraída das primeiras 30 primeiras conversas do agente com o paciente fictício.

Usuário: Oi, eu estou me sentindo um pouco tonta, sera o que pode ser?

Agente: Oi, Maria. A tontura pode ser causada por várias coisas, como níveis baixos de açúcar no san-gue, desidratação ou até mesmo efeitos colaterais de medicamentos. Como você tem diabetes, é importante verificar seus níveis de glicose para ver se estão dentro do normal. Também é uma boa ideia se hidratar bem. Se a tontura persistir, é melhor falar com seu médico para uma avaliação mais detalhada. Quer que eu te lembre de medir sua glicose agora?

Usuário: Por favor, me lembre daqui 30 minutos.

Agente: Claro, Maria! Vou te lembrar de medir sua glicose daqui a 30 minutos. Enquanto isso, tente descansar um pouco e beber água.

Usuário: Falando em se sentir melhor, como era o nome do aplicativo para gerenciamento de tarefas que voce sugeriu mesmo?

Agente: Eu sugeri o Todoist e o Trello. Ambos são ótimos para organizar suas tarefas e ajudar a manter tudo em ordem. Você já teve a chance de dar uma olhada neles?

Usuário: Só no Trello.

Agente: Legal! O Trello é bem visual e fácil de usar. Se você quiser algo mais simples e direto, o To-doist também pode ser uma boa opção. Se precisar de ajuda para configurar, é só falar!

Usuário: Pensando em criar numa ferramenta uma lista daquelas receitas q ja fiz e que gostei. aceito mais sugestões inclusive.

Agente: Ótima ideia, Maria! Aqui vão algumas sugestões de receitas que acho que você pode gostar:

1. Tilápia ao forno com legumes: Tempere a tilápia com limão, alho e ervas, e asse junto com cenouras, brócolis e batatas-doces.
2. Salada de quinoa com frango grelhado: Misture quinoa cozida com frango grelhado, tomates-cereja, pepino, abacate e um molho de limão e azeite.
3. Sopa de lentilhas: Uma sopa nutritiva com lentilhas, cenoura, aipo, alho e espinafre.
4. Omelete de espinafre e cogumelos: Uma opção rápida e saudável para o café da manhã ou jantar.

Usuário: Posso comer isso tudo ai tranquilamente?

Agente: Sim, Maria, todas essas receitas são saudáveis e podem ser incorporadas na sua dieta, especi-almente considerando que você está tentando manter seus níveis de açúcar no sangue estáveis. Apenas lembre-se de monitorar as porções e evitar ingredientes que possam afetar negativa-mente seus níveis de glicose. Se tiver alguma dúvida especifica sobre algum ingrediente, é sempre bom consultar seu médico ou nutricionista.

Usuário: Acho q vou tentar montar isso amanha de manhã.

Agente: Parece um ótimo plano, Maria! Boa ideia, já que você gosta de estudar e se sente mais produ-tiva de manhã. Se precisar de mais sugestões ou ajuda com algo, estarei por aqui. E não se es-queça de medir sua glicose daqui a pouco!

user_memory

- Maria tem 25 anos.
- Maria tem diabetes.
- sua mae se chama Ana.
- Maria gosta de ler.
- toma insulina e Metformina.
- prefere lembretes automáticos para tomar insulina.
- Prefere estudar de manhã.
- gosta de mapas mentais.
- Adora caminhar ouvindo podcasts.
- Adora vegetais, proteínas magras, iogurte com nozes.
- Gosta de salmão grelhado com cenouras e brócolis.
- Gosta de piano clássico, especialmente Chopin e Beethoven.
- Gosta de yoga e pratica regularmente.

...

miro

Fonte: próprio autor.

Dessa forma, pode-se perceber como a memória traz uma qualidade melhor ao agente para a conversa por demonstrar conhecimento de que o usuário tem diabetes. Em decorrência disso, o agente pode tomar ações mais rápidas, sugestões de alimentação baseadas na preferência do usuário (no exemplo acima, por legumes e peixes), rememorar ao usuário sobre sugestões passadas e reforçar comportamentos positivos anteriores. Como consequência, o agente pode gerar maior engajamento e satisfação do usuário, além de aumentar a velocidade da entrega da informação, visto que não é necessário compreender o contexto a cada nova conversa/ interação.

7 CONSIDERAÇÕES FINAIS

O paradigma apresentado pelo MemoryGraph permeia todo o processo de representação, armazenamento e busca do conhecimento, apresentando ser um método eficaz para representar diferentes tipos de conhecimento, sendo possível em grande parte pela capacidade dos LLMs. Esse processo contrasta com o paradigma mais tradicional, que é a representação baseada em vetores, que, apesar de ser o responsável por diversos avanços em PNL, apresentou diversas limitações quanto a representação do conhecimento.

Apesar disso, ainda há desafios a serem superados como, por exemplo, a incerteza nos resultados devido à sensibilidade dos *prompts*, reduzindo a consistência das extrações. Outra limitação apresentada pelo uso de LLM é a variabilidade de respostas entre diferentes modelos, tornando a substituição de modelos uma tarefa custosa.

O lento processo de construção do grafo de conhecimento pode inviabilizar certas aplicações, principalmente aquelas que necessitam de alteração constante e rápida do conhecimento. Uma opção para esses casos é um processo em duas etapas em que é realizada a indexação para buscas vetoriais de forma rápida enquanto a construção do grafo ocorre em paralelo. Outro aspecto importante no processo de construção são os custos. Como discutido, estes podem aumentar em centena de vezes o custo da aplicação e, por isso, é necessário avaliar cada cenário e a necessidade para definir os melhores métodos e parâmetros desse sistema.

Uma forma de reduzir esse custo pode ser através do uso de LLM menores e especialistas nas tarefas de construção do grafo e a realização de *fine-tuning* em modelos para geração de *cypher queries* que podem permitir a unificação e simplificação do processo de busca, além de aumentar a capacidade do agente em buscar informações no GC. Esse processo pode ser ainda mais otimizado com a exploração de diferentes ferramentas de grafo como, por exemplo, ferramentas de banco de grafos baseados em Memória RAM e não disco, que permitem um aumento na velocidade da busca.

Com o aumento da capacidade do agente em acessar a memória, trabalhos futuros podem explorar memórias mais complexas e que fornecem informações mais detalhadas ao agente. Por exemplo, a incorporação de resultados de processos

decisivos, como o planejamento, permitirá que futuras decisões sejam baseadas em experiências anteriores, mitigando erros. Além disso, a inclusão de resultados do uso de ferramentas em cenários específicos pode permitir que o agente tome decisões não apenas com base no presente, mas também em experiências positivas passadas, melhorando a consistência dos agentes no uso dessas ferramentas. Ademais, uma evolução natural desta proposta é a base de conhecimento em GC multimodais, o que pode ampliar ainda mais a capacidade dos agentes conversacionais, tornando-os ferramentas ainda mais robustas e versáteis.

Essa pesquisa explorou os estágios iniciais da aplicação de diferentes abordagens de extração e construção do GC, resultado em um GC mais próximo de um grafo de propriedade ou RDF (como foi utilizado). O grafo de propriedade pode permitir um GC mais conciso, menos ambíguo e de mais simples manutenção. Esse resultado pode ser otimizado com o uso de grafos de ontologias em grafos especialistas ou por meio de métodos automáticos de geração de ontologias.

Para uma aplicação prática, é crucial avaliar se é mais vantajoso ter um único grafo para todos os usuários, um grafo por usuário ou uma divisão diferente como, por exemplo, um grafo por empresa. Além disso, sistemas multi-*graph* que utilizam grafos especializados ao invés de um único grafo grande, podem oferecer melhor performance e organização do conhecimento.

Finalmente, essa pesquisa mostrou a necessidade de melhores métricas de avaliação e controle da qualidade do GC, assim como *benchmarks* para a avaliação de agentes e sistemas de busca baseado em GC, como o MemoryGraphv.

8 CONCLUSÃO

O MemoryGraph como metodologia de implementação de memória para agente conversacional utilizando grafos de conhecimento proporcionaram percepções valiosos sobre as vantagens e desafios dessa abordagem. As propostas de construção do GC se mostraram eficazes em comparação ao uso de bancos vetoriais, apesar das limitações de tempo e custo apresentadas. Pode ser observado que, diferente da maioria das abordagens apresentadas no mercado e na academia, a divisão da extração de entidades e relação em dois processos se mostrou mais eficaz, apesar de menos eficiente. O mecanismo de acesso à informação proposto se mostrou superior ao RAG vetorial em vários aspectos como, por exemplo, a melhoria na qualidade do contexto fornecido para o modelo LLM, resultando em respostas mais precisas e contextualizadas. Além disso, a introdução do conceito de *user_memory* permitiu que o agente obtivesse uma representação mais precisa e dinâmica dos usuários através da interação com ele, melhorando a personalização e relevância das interações por meio das experiências do agente.

O MemoryGraph se mostrou um paradigma que avanço o conceito de inteligência artificial rumo a um sistema mais completo, através da sua representação rica, avançando o conceito difundido e já tradicional da representação vetorial do conhecimento. Trazendo uma perspectiva diferente à aplicações inteligentes e permitindo que empresas possuam sistemas especializados em seus dados sem a necessidade de terem que treinar ou ajustar LLM caras e de difícil acesso a cada novo documento ou conversa realizada. Além disso, viabiliza aplicações que necessitam de uma maior transparência na base de conhecimento das LLMs, que possuem uma dinâmica de atualização mais complexa e frequente, permitindo representar e buscar informações em uma base de conhecimento mais complexa e relacionada.

A sinergia entre GC e LLM foi evidente ao longo do trabalho, se complementando em diversos processos, como a construção do grafo e fornecimento de contexto mais rico para o agente. Já a combinação de GC com banco vetorial demonstrou-se uma abordagem complementar e não excludentes, proporcionando uma visão mais completa do conhecimento, trazendo novas capacidades a partir da integração da informação relacional e estruturada do GC com a não estruturada e intuitiva dos modelos LLMs e representações vetoriais. Além disso, foram utilizados

seis métricas em que o LLM atuou como juiz, mostrando sua ampla aplicação em todas as fases do desenvolvimento do agente e demonstrando, mais uma vez, a capacidade superior do GC em armazenar e retornar informações relevantes. Além de trazer percepções sobre a capacidade de auto avaliação do agente.

Contudo, é importante reconhecer os desafios associados ao MemoryGraph, como o maior custo de manutenção e construção dos grafos de conhecimento e o aumento do tempo necessário para a construção da base de conhecimento e muitas vezes ao acesso à informação. Portanto, embora os benefícios do MemoryGraph sejam evidentes, é crucial equilibrar esses ganhos com os custos e esforços adicionais envolvidos, garantindo que a solução adotada seja viável e sustentável a longo prazo.

REFERÊNCIAS

- ABADI, M.; AGARWAL, A.; BARHAM, P.; BREVDO, E.; CHEN, Z.; CITRO, C.; CORRADO, G. S.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; GOODFELLOW, I.; HARP, A.; IRVING, G.; ISARD, M.; JOZEFOWICZ, R.; JIA, Y.; ABBASIANAEV, Z.; MOMTAZI, S. Text-based question answering from information retrieval and deep neural network perspectives: a survey. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, v.11, n.6, 2021.
- ABBASIANAEV, Z.; MOMTAZI, S. Text-based question answering from information retrieval and deep neural network perspectives: A survey. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, v.11, n.6, 2021.
- ABDIN, M. et al. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone. **ArXiv preprint**, arXiv: 2404.14219, 2024.
- AMAZON. Amazon Staff. **How Alexa keeps getting smarter**, 2018. Disponível em: <https://www.aboutamazon.com/devices/how-alexa-keeps-getting-smarter>. Acesso em: 20 out. 2021.
- ANGLES, R.; ARENAS, M.; BARCELÓ, P.; BONCZ, P. A.; FLETCHER, G. H. L.; GUTIERREZ, C.; LINDAAKER, T.; PARADIES, M.; PLANTIKOW, S.; SEQUEDA, J. F.; VAN REST, O.; VOIGT, H. **G-CORE: A Core for Future Graph Query Languages**, 2017. Disponível em: <https://arxiv.org/pdf/1712.01550.pdf>. Acesso em: 20 out. 2021.
- ANGLES, R.; ARENAS, M.; BARCELÓ, P.; HOGAN, A.; REUTTER, J. L.; VRGOC, D. Foundations of Modern Query Languages for Graph Databases. **ACM Computing Surveys**, v.50, n.5, p.1–68, 2016.
- ANNERVAZ, K. M.; CHOWDHURY, S. B. C.; DUKKIPATI, A. Learning beyond datasets: Knowledge graph augmented neural networks for natural language processing. **ArXiv preprint**, arXiv:1802.05930, 2018.
- APPENZELLER, G.; BORNSTEIN, M.; CASADO, M. **Navigating the High Cost of AI Compute**. Adreessen Horowitz, 2023. Disponível em: <https://a16z.com/navigating-the-high-cost-of-ai-compute/>. Acesso em: 20 jun. 2024.
- ARORA, D.; KINI, A.; CHOWDHURY, S. R.; NATARAJAN, N.; SINHA, G.; SHARMA, A. GAR-meets-RAG Paradigm for Zero-Shot Information Retrieval. **ArXiv preprint**, arXiv: 2310.20158, 2023.
- ASAI, A.; WU, Z.; WANG, Y.; SIL, A.; HAJISHIRZI, H. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. **ArXiv preprint**, arXiv: 2310.11511, 2023.
- BAJAJ, P. et al. MS MARCO: A Human Generated MACHine Reading COMprehension Dataset. **ArXiv preprint**, arXiv: 1611.09268, 2016.

- BAKKER, René Ronald. **Knowledge Graphs: Representation and Structuring of Scientific Knowledge**. [Thesis] University of Twente, Enschede, The Netherlands, 1987.
- BANERJEE, S.; LAVIE, A. 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**. Ann Arbor, Michigan, 2005. p.65–72.
- BARNETT, S.; KURNIAWAN, THUDUMU, S.; BRANNELLY, Z.; ABDELRAZEK, M. Seven Failure Points When Engineering a Retrieval Augmented Generation System. **ArXiv preprint**, arXiv: 2401.05856, 2024.
- BENGIO, Y.; DUCHARME, R.; VINCENT, P.; JAUVIN, C. A neural probabilistic language model. **Journal of machine learning research**, v.3, p.1137-1155, 2003.
- BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. **IEEE Transactions on in Neural Networks**, v.5, n.2, p.157-166, 1994.
- BENHAM, R.; CULPEPPER, J. S. Risk-reward trade-offs in rank fusion. **Proceedings of the 22nd Australasian Document Computing Symposium**, p.1–8, 2017.
- BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. **Scientific American**, v.284, n.5, p.28–37, 2001.
- BISHOP, C. M. **Neural networks for pattern Recognition**. Oxford University Press, 1996.
- BIZER, C.; LEHMANN, J.; KOBILAROV, G.; AUER, S.; BECKER, C.; CYGANIAK, R.; HELLMANN, S. Dbpedia-a crystallization point for the web of data. **Journal of Web Semantics**, v.7, n.3, p.154–165, 2009.
- BIZER, Christian; HEATH, Tom; BERNERS-LEE, Tim. Linked data: The story so far. In: SHET, Amit. **Semantic services, interoperability and web applications: emerging concepts**. IGI Global, 2011. p.205–227.
- BOJANOWSKI, P.; GRAVE, E.; JOULIN, A.; MIKOLOV, T. Enriching Word Vectors with Subword Information. **ArXiv preprint**, arXiv:1607.04606, 2017.
- BOLLACKER, K.; COOK, R.; TUFTS, P. FREEBASE: A shared database of structured general human knowledge. In: **AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence**, v.7, p.1962–1963, 2007.
- BOLLACKER, K.; TUFTS, P.; PIERCE, T.; COOK, R. A platform for scalable, collaborative, structured information integration. In: **IIWeb'07 Workshop on Information Integration on the Web**, 2007.

BOMMASANI, R. et al. On the opportunities and risks of foundation models. **ArXiv preprint**, arXiv:2108.07258, 2021.

BORGEAUD, S.; MENSCH, A.; HOFFMANN, J.; CAI, T.; RUTHERFORD, E.; MILLICAN, K.; VAN DEN DRIESSCHE, G.; LESPIAU, J.; DAMOC, B.; CLARK, A.; DE LAS CASAS, D.; GUY, A.; MENICK, J.; RING, R.; HENNIGAN, T.; HUANG, S.; MAGGIORE, L.; JONES, C.; CASSIRER, A.; BROCK, A.; PAGANINI, M.; IRVING, G.; VINYALS, O.; OSINDERO, S.; SIMONYAN, K.; RAE, J. W.; ELSEEN, E.; SIFRE, L. Improving language models by retrieving from trillions of tokens. **ArXiv preprint**, arXiv:2112.04426, 2021.

BRAN, A. M.; COX, S.; SCHILTER, O.; BALDASSARI, C.; WHITE, A. D.; SCHWALLER, P. ChemCrow: Augmenting large-language models with chemistry tools. **ArXiv preprint**, arXiv: 2308.04026, 2023.

BRATANIC, T. **Using a Knowledge Graph to Implement a RAG Application**. Neo4j, 2024. Disponível em: <https://neo4j.com/developer-blog/knowledge-graph-rag-application/>. Acesso em: 26 jun. 2024.

BRICKLEY, D.; GUHA, R. V. **RDF Schema 1.1, W3C Recommendation 25 February 2014**. W3C Recommendation. World Wide Web Consortium, 2014. Disponível em: <https://www.w3.org/TR/2014/REC-rdf-schema-20140225/>. Acesso em: 16 out. 2021.

BRIN, S.; PAGE, L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. **Computer Networks**, v.30, p.107-117, 1998.

BROWN, T. B.; MANN, B.; RYDER, N.; SUBBIAH, M.; KAPLAN, J.; DHARIWAL, P.; NEELAKANTAN, A.; SHYAM, P.; SASTRY, G.; ASKELL, A.; AGARWAL, S.; HEBERT-VOSS, A.; KRUEGER, G.; HENIGHAN, T.; CHILD, R.; RAMESH, A.; ZIEGLER, D. M.; WU, J.; WINTER, C.; HESSE, C.; CHEN, M.; SIGLER, E.; LITWIN, M.; GRAY, S.; CHESSE, B.; CLARK, J.; BERNER, C.; MCCANDLISH, S.; RADFORD, A.; SUTSKEVER, I.; AMODEI, D. Language models are few-shot learners. **ArXiv preprint**, arXiv:2005.14165, 2020.

BUIBELAAR, P.; CIMIANO, P.; MAGNINI, B. **Ontology learning from text: methods, Evaluation and applications**. IOS Press, 2005.

CALLAHAN, A.; CRUZ-TOLEDO, J.; ANSELL, P.; DUMONTIER, M. **Bio2RDF Release 2: Improved Coverage, Interoperability and Provenance of Life Science Linked Data**. In: CIMIANO, P.; CORCHO, O.; PRESUTTI, V.; HOLLINK, L.; RUDOLPH, S. (eds) *The Semantic Web: Semantics and Big Data*. ESWC 2013. Lecture Notes in Computer Science, v.7882, 2013. Springer, Berlin, Heidelberg.

CARRERAS, X.; MÀRQUEZ, L.; PADRÓ, L. A simple named entity extractor using AdaBoost. In: **Proceedings of the seventh conference on Natural language learning at HLT-NAACL**, v.4, p.152-155, 2003.

CHAI, Z.; ZHANG, T.; WU, L.; HAN, K.; HU, X.; HUANG, X.; YANG, Y. GraphLLM: boosting graph reasoning ability of large language model. **ArXiv preprint**, arXiv: 2310.05845, 2023.

CHAN, A.; EZELL, C.; KAUFMANN, M.; WEI, K.; HAMMOND, L.; BRADLEY, H.; BLUEMKE, E.; RAJKUMAR, N.; KRUEGER, D.; KOLT, N.; HEIM, L.; ANDERLJUNG, M. Visibility into AI Agents. **ArXiv preprint**, arXiv: 2401.13138, 2024.

CHATTERJEE, C. C. **Basics of the Classic CNN. How a classic CNN (Convolutional Neural Network) work?**, 2019. Disponível em: <https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add>. Acesso: 2 mar. 2024.

CHAVAN, A. **Complete Tutorial on Named Entity Recognition (NER) using Python and Keras**. AI TIME JOURNAL, 2020. Disponível em: <https://www.aitimejournal.com/complete-tutorial-on-named-entity-recognition-ner-using-python-and-keras/7728/>. Acesso em: 15 mai. 2024.

CHEIN, Michel; MUGNIER, Marie-Laure. **Graph-based knowledge representation: computational foundations of conceptual graphs**. Springer Science & Business Media, 2009.

CHEN, D.; FISCH, A.; WESTON, J.; BORDES, A. Reading Wikipedia to Answer Open-Domain Questions. **ArXiv preprint**, arXiv: 1704.00051, 2017.

CHOI, E.; PALOMAKI, J.; LAMM, M.; KWIATKOWSKI, T.; DAS, D.; COLLINS, M. Decontextualization: Making Sentences Stand-Alone. **Transactions of the Association for Computational Linguistics**, v.9, p.447-461, 2021.

CHRISTIANO, P. F.; LEIKE, J.; BROWN, T.; MARTIC, M.; LEGG, S.; AMODEI, D. Deep reinforcement learning from human preferences. **NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems**, p.4302-4310, 2017.

CHURCH, K. W. A stochastic parts program and noun phrase parser for unrestricted text. In: **Proceedings of the Second Conference on Applied Natural Language Processing**. Austin, Texas, 1988, p.136—143.

CLAVERA, W. V. **Aprendizado de Máquina (Machine Learning)**, 2019. Disponível em: <https://www.redesdesaude.com.br/aprendizado-de-maquina-machine-learning/>. Acesso em: 2 mar. 2024.

COURSEY, K.; MIHALCEA, R. Topic Identification Using Wikipedia Graph Centrality. In: **NAACL-Short '09: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics**, 2009. p.117-120.

COST ESTIMATES FOR GPT-4. **Cost estimates for GPT-4**, 2023. Disponível em: https://colab.research.google.com/drive/1O99z9b1I5O66bT78r9ScsIE_nOj5irN9?usp=sharing. Acesso em 23 jun. 2024.

DAN, Y.; LEI, Z.; GU, Y.; LI, Y.; YIN, J.; LIN, J.; YE, L.; TIE, Z.; ZHOU, Y.; WANG, Y.; ZHOU, A.; ZHOU, Z.; CHEN, Q.; ZHOU, J.; HE, L.; QIU, X. EduChat: A Large-Scale Language Model-based Chatbot System for Intelligent Education. **ArXiv preprint**, arXiv: 2308.02773, 2023.

DANG, N. C.; MORENO-GARCÍA, M. N.; DE LA PRIETA, F. Sentiment Analysis Based on Deep Learning: A Comparative Study. **ArXiv preprint**, arXiv:2006.03541, 2020.

DAVIS, E.; MARCUS, G. Commonsense reasoning and commonsense knowledge in artificial intelligence. **Communications of the ACM**, v.58, n.9, p.92–103, 2015.

DEEPGRAM. The Top 10 arXiv Papers About AI Agents. **Deepgram**, 2024. Disponível em: <https://deepgram.com/learn/top-arxiv-papers-about-ai-agents#1-a-framework-for-intelligent-multiagent-system-based-neural-networks>. Acesso em: 21 jul. 2024.

DEFFERRARD, M.; BRESSON, X.; VANDERGHEYNST, P. Convolutional neural networks on graphs with fast localized spectral filtering. **Advances in neural information processing systems**, v.29, p.3844-3852, 2016.

DENG, J.; DONG, W.; SOCHER, R.; LI, L.-J.; LI, K.; FEI-FEI, L. Imagenet: A large-scale hierarchical image database. In: **2009 IEEE conference on computer vision and pattern recognition**, 2009. p.248–55.

DENG, L.; LIU, Y. **Deep Learning in natural language processing**. Springer, 2018.

DEVLIN, J.; CHANG, M.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. **ArXiv preprint**, arXiv:1312.3005, 2019.

DIENG, R.; GIBOIN, A.; TOURTIER, P.; CORBY, O. Knowledge Acquisition for Explainable, Multi-Expert, Knowledge-Based Design Systems. In: WETTER, T.; ALTHOFF, K.; BOOSE, J. H.; GAINES, B. R.; LINSTER, M.; SCHMALHOFER, F. **Current Developments in Knowledge Acquisition - EKAW'92, 6th European Knowledge Acquisition Workshop**. Springer, Berlin, 1992.

DODDINGTON, G. R.; MITCHELL, A.; PRZYBOCKI, M. A.; RAMSHAW, L. A.; STRASSEL, S.; WEISCHEDEL, R. M. The automatic content extraction (ace) program-tasks, data, and evaluation. In: **Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)**. Lisbon, Portugal, 2004.

DONAHUE, J.; DIELEMAN, S.; BINKOWSKI, M.; ELSEN, E.; SIMONYAN, K. End-to-End Adversarial Text-to-Speech, **ArXiv preprint**, arXiv:2006.03575, 2020.

DONGMIN, P.; HONG, S.; HAN, B.; LEE, K. M. Continual learning by asymmetric loss approximation with single-side overestimation. In: **Proceedings of the IEEE/CVF International Conference on Computer Vision**, 2019. p.3335-3344.

DROST-FROMM, I. **Elasticsearch Testing & QA: Testing Levels of Elasticsearch**, 2014. Disponível em: <https://www.elastic.co/blog/elasticsearch-testing-qa-testing-levels-elasticsearch>. Acesso em: 13 out. 2021.

DURANTE, Z.; HUANG, Q.; WAKE, N.; GONG, R.; PARK, J. S.; SARKAR, B.; TAORI, R.; NODA, Y.; TERZOPOULOS, D.; CHOI, Y.; IKEUCHI, K.; VO, H.; FEI-FEI, L.; GAO, J. AGENT AI: surveying the horizons of multimodal interaction. **ArXiv preprint**, arXiv: 2401.03568, 2024.

EHRLINGER, L.; WÖß, W. Towards a definition of knowledge graphs. **SEMANTICS**, v.48, p.1–4, 2016.

ELBASSUONI, S.; RAMANATH, M.; SCHENKEL, R.; SYDOW, M.; WEIKUM, G. Language-model based ranking for queries on RDF-graphs. In: **CIKM 2009 Proceedings of the 18th ACM Conference on Information and Knowledge Management**, 2009. p.977–986.

ES. S.; JAMES, J.; ESPINOSA-ANKE, L.; SCHOCKAERT, S. RAGAS: Automated Evaluation of Retrieval Augmented Generation. **ArXiv preprint**, arXiv: 2309.15217, 2024.

FATEMI, B.; HALCROW, J.; PEROZZI. Talk like a graph: encoding graphs for large language models. **ArXiv preprint**, arXiv: 2310.04560, 2023.

FENG, F.; YANG, Y.; CER, D.; ARIVAZHAGAN, N.; WANG, W. Language-agnostic bert sentence embedding. **ArXiv preprint**, arXiv:2007.01852, 2020.

FENG, Y.; CHEN, X.; LIN, B. Y.; WANG, P.; YAN, J.; REN, X. Scalable multi-hop relational reasoning for knowledge-aware question answering. **ArXiv preprint**, arXiv:2005.00646, 2020.

FRANCIS, N.; GREEN, A.; GUAGLIARDO, P.; LIBKIN, L.; LINDAAKER, T.; MARSAULT, V.; PLANTIKOW, S.; RYDBERG, M.; SELMER, P.; TAYLOR, A. Cypher: An Evolving Query Language for Property Graphs. In: **SIGMOD '18: Proceedings of the 2018 International Conference on Management of Data**, 2018. p.1433–1445.

FRIEL, R.; BELYI, M.; SANYAL, A. RAGBench: Explainable Benchmark for Retrieval-Augmented Generation Systems. **ArXiv preprint**, arXiv: 2407.11005, 2024.

FU, B.; QIU, Y.; TANG, C.; LI, Y.; YU, H.; SUN, J. A survey on complex question answering over knowledge base: Recent advances and challenges. **ArXiv preprint**, arXiv:2007.13069, 2020.

GANTZ, J.; REINSEL, D. **The digital universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East.** IDC IVIEW, 2012. Disponível em: <https://www.cs.princeton.edu/courses/archive/spring13/cos598C/idc-the-digital-universe-in-2020.pdf>. Acesso em: 10 jun. 2024.

GAO, Y.; XIONG, Y.; GAO, X.; JIA, K.; PAN, J.; BI, Y.; DAI, Y.; SUN, J.; WANG, M.; WANG, H. Retrieval-Augmented Generation for Large Language Models: A Survey. **ArXiv preprint**, arXiv: **2312.10997**, 2024.

GECER, B.; PLOUMPIS, S.; KOTSIA, I.; ZAFEIRIOU, S. GANFIT: Generative Adversarial Network Fitting for High Fidelity 3D Face Reconstruction. **ArXiv preprint**, arXiv: **1902.05978**, 2020.

GIRSHICK, R.; RADOSAVOVIC, I.; GKIOXARI, G.; DOLL, P.; HE, K. **DETECTRON**, 2018. Disponível em: <https://github.com/facebookresearch/detectron>. Acesso em: 20 set. 2021.

GOLDBERG, Yoav. **Neural Network Methods in Natural Language Processing.** Morgan & Claypool, 2017.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Feedforward Networks. Back-propagation and other differentiation algorithms**, 2016. Disponível em: <https://www.deeplearningbook.org/contents/mlp.html#pf25>. Acesso em 15 out. 2021.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning.** MIT Press, 2016.

GOOGLE DEEP MIND. **Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context.** Google Deep Mind, 2024. Disponível em: https://storage.googleapis.com/deepmind-media/gemini/gemini_v1_5_report.pdf. Acesso em: 10 ago. 2024.

GROOTENDORST, M. **KeyBERT: Minimal keyword extraction with BERT**, 2020. Disponível em: <https://github.com/MaartenGr/KeyBERT>. Acesso em: 05 nov. 2021.

GROVER, A.; LESKOVEC, J. Node2vec: Scalable feature learning for networks. In: **Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining**, 2016. p.855-864.

GUARINO, N.; OBERLE, D.; STAAB, S. What is an ontology?. In: **Handbook on ontologies.** Springer, Berlin, 2009. p. 1-17.

GUO, T.; CHEN, X.; WANG, Y.; CHANG, R.; PEI, S.; CHAWLA, N. V.; WIEST, O.; ZHANG, X. Large Language Model based Multi-Agents: A Survey of Progress and Challenges. **ArXiv preprint**, arXiv: 2402.01680, 2024.

GUO, A.; LI, X.; XIAO, G.; TAN, Z.; ZHAO, X. SpCQL: A Semantic Parsing Dataset for Converting Natural Language into Cypher. **CIKM '22: Proceedings of the 31st ACM**

International Conference on Information & Knowledge Management, p.3973-3977, 2022.

GUPTA, V.; CHINNAKOTLA, M.; SHRIVASTAVA, M. Retrieve and re-rank: A simple and effective IR approach to simple question answering over knowledge graphs. In: **Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)**. Brussels, Belgium, 2018, p.22-27.

GUU, K.; LEE, K.; TUNG, Z.; PASUPAT, P.; CHANG, M. W. Realm: Retrieval-augmented language model pre-training. **ArXiv preprint**, arXiv:2002.08909, 2020.

HAASE, P.; NIKOLOV, A.; TRAME, J.; KOZLOV, A.; HERZIG, D. M. Alexa, Ask Wikidata! Voice Interaction with Knowledge Graphs using Amazon Alexa. In: **The 16th International Semantic Web Conference**. Vienna, Austria, 2017.

HAMILTON, S. Blind Judgement: Agent-Based Supreme Court Modelling With GPT. **ArXiv preprint**, arXiv: 2301.05327, 2023.

HAMILTON, W. L.; YING, R.; LESKOVEC, J. Inductive representation learning on large graphs. In: **Proceedings of the 31st International Conference on Neural Information Processing Systems**, 2017. p.1025-1035.

HAN, X.; GAO, T.; YAO, Y.; YE, D.; LIU, Z.; SUN, M. OpenNRE: An open and extensible toolkit for neural relation extraction. **ArXiv preprint**, arXiv:1909.13078, 2019.

HANEY, D.; GIBSON, D. Stack Overflow Blog. Ask like a human: Implementing semantic search on Stack Overflow. **Stack Overflow blog**, 2023. Disponível em: <https://stackoverflow.blog/2023/07/31/ask-like-a-human-implementing-semantic-search-on-stack-overflow/>. Acesso em: 24 nov. 2023.

HARRIS, S.; SEABORNE, A.; PRUD'HOMMEAUX, E. **SPARQL 1.1 Query Language, W3C Recommendation 21 March 2013**. W3C Recommendation. World Wide Web Consortium. 2013. Disponível em: <https://www.w3.org/TR/2013/REC-sparql11-query-20130321/>. Acesso em: 17 out. 2021.

HARRIS, Z. S. Distributional structure. **Word**, v.10, n.2-3, p.146-162, 1954.

HAWKINS, Jeff. **A thousand brains: A new theory of intelligence**. Basic Books: New York, 2021.

HE, Q. **Machine Learning in LinkedIn Knowledge Graph**, 2016. Disponível em: https://www.linkedin.com/pulse/machine-learning-linkedin-knowledge-graph-qi-he/?trk=pulse_spock-articles. Acesso em 20 out. 2021.

HEARST, M. A.; DUMAIS, S. T.; OSUNA, E.; PLATT, J.; SCHOLKOPF, B. Support vector machines. **IEEE Intelligent Systems and their applications**, v.13, n.4, p.18-28, 1998.

HENDLER, J. A.; HOLM, J.; MUSIALEK, C.; THOMAS, G. US Government Linked Open Data: Semantic.data.gov. **IEEE Intelligent Systems**, v.27, n.3, p.25–31, 2012.

HIMMELHUBER, A.; GRIMM, S.; ZILLNER, S.; JOBLIN, M.; RINGSQUANDL, M.; RUNKLER, T. Combining Sub-Symbolic and Symbolic Methods for Explainability. **ArXiv preprint**, arXiv: 2112.01844, 2021.

HINTON, G. E.; OSINDERO, S.; TEH, Y. W. A fast-learning algorithm for deep belief nets. **Neural computation**, v.18, n.7, p.1527-1554, 2006.

HITZLER, P.; KRÖTZSCH, M.; PARSIA, B.; PATEL-SCHNEIDER, P. F.; RUDOLPH, S. **OWL 2 Web Ontology Language Primer (Second Edition). W3C Recommendation 11 December 2012**. W3C Recommendation. World Wide Web Consortium. Disponível em: <https://www.w3.org/TR/2012/REC-owl2-primer-20121211/>. Acesso em: 16 out. 2021.

HITZLER, P.; KRÖTZSCH, M.; RUDOLPH, S. **Foundations of Semantic Web Technologies**. Chapman & Hall/CRC, 2009.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. **Neural Computation**, v.9, n.8, p.1735-1780, 1997.

HOFFART, J.; SUCHANEK, F. M.; BERBERICH, K.; LEWIS-KELHAM, E.; MELO, G.; WEIKUM, G. YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. In: **Proceedings of the 20th International Conference on World Wide Web**. Hyderabad, India, 2011. p.229–232.

HOGAN, A. et al. Knowledge Graphs. **ACM Computing Surveys (CSUR)**, v.54, p.1-37, 2021.

HONG, S. et al. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. **ArXiv preprint**, arXiv: 2308.00352, 2023.

HORROCKS, I.; PATEL-SCHNEIDER, P. F.; BOLEY, H.; TABET, S.; GROSOFF, B.; DEAN, M. **SWRL: A Semantic Web Rule Language Combining OWL and RuleML**. W3C Member Submission, 2004. Disponível em: <https://www.w3.org/Submission/2004/SUBM-SWRL-20040521/>. Acesso em: 16 out. 2021.

HU, E. J.; SHEN, Y.; WALLIS, P.; ALLEN-ZHU, Z.; LI, Y.; WANG, S.; WANG, L.; CHEN, W. LoRA: Low-Rank Adaptation of Large Language Models. **ArXiv preprint**, arXiv:2106.09685, 2021.

HUANG, L.; YU, W.; ZHONG, W.; FENG, Z.; WANG, H.; CHEN, Q.; PENG, W.; FENG, X.; QIN, B.; LIU, T. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. **ArXiv preprint**, arXiv: 2311.05232, 2023.

HUDSON, D. A.; MANNING, C. D. Compositional attention networks for machine reasoning. **ArXiv preprint**, arXiv:1803.03067, 2018.

IZACARD, G.; GRAVE, E. Leveraging passage retrieval with generative models for open domain question answering. **ArXiv preprint**, arXiv:2007.01282, 2020.

JAIN, S. Question answering over knowledge base using factual memory networks. In: **Proceedings of the NAACL student research workshop**. San Diego, California, 2016, p.109-115.

JANEZ, B.; LEBAN, G.; GROBELNIK, M. Annotating Documents with Relevant Wikipedia Concepts. In: **Proceedings of the Slovenian Conference on Data Mining and Data Warehouses (SiKDD 2017)**. Ljubljana, Slovenia, 2017.

JAVOR, A. **Memgraph or Neo4j: Analyzing Write Speed Performance**. Memgraph, 2024. Disponível em: <https://memgraph.com/blog/memgraph-or-neo4j-analyzing-write-speed-performance>. Acesso em: 22 jun. 2024.

JIANG, Y. F.; MA, N. A Plan Recognition Algorithm Based on Plan Knowledge Graph. **Journal of Software**, v.13, 2002.

JIANG, Z.; XU, F. F.; GAO, L.; SUN, Z.; LIU, Q.; DWIVEDI, J.; YANG, Y.; CALLAN, J.; NEUBIG, G. Active Retrieval Augmented Generation. **ArXiv preprint**, arXiv:2305.06983, 2023.

JOHNSON, M. How the statistical revolution changes (computational) linguistics. In: **ILCL '09: Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?**, 2009, p.3-11.

JOHNSON-LAIRD, P. N. Mental models in cognitive science. **Cognitive science**, v.4, n.1, p.71–115, 1980.

JORDAN, Michael. **Serial order: a parallel distributed processing approach**. Institute for Cognitive Science, University of California, San Diego. Disponível em: <https://cseweb.ucsd.edu/~gary/258/jordan-tr.pdf>. Acesso em: 10 out. 2021.

JOSHI, M.; CHOI, E.; WELD, D.; ZETTLEMOYER, L. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. **ArXiv preprint**, arXiv:1705.03551, 2017.

JURAFSKY, Dan; MARTIN, James. **Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition**. Prentice Hall, 2008.

KAISER, L.; KUDLUR, M.; LEVENBERG, J.; MANÉ, D.; SCHUSTER, M.; MONGA, R.; MOORE, S.; MURRAY, D.; OLAH, C.; SHLENS, J.; STEINER, B.; SUTSKEVER, I.;

TALWAR, K.; TUCKER, P.; VANHOUCHE, V.; VASUDEVAN, V.; VIÉGAS, F.; VINYALS, O.; WARDEN, P.; WATTENBERG, M.; WICKE, M.; YU, Y.; ZHENG, X. **TensorFlow: Large-scale machine learning on heterogeneous systems**, 2015.

KAMAR, E.; LEE, P.; LEE, Y, T.; LI, Y.; LUNDBERG, S.; NORI, H.; PALANGI, H.; RIBEIRO, M. T.; ZHANG, Y. Sparks of Artificial General Intelligence: Early experiments with GPT-4. **ArXiv preprint**, arXiv: 2303.12712, 2023.

KARPUKHIN, V.; OGUZ, B.; MIN, S.; LEWIS, P.; WU, L.; EDUNOV, S.; CHEN, D.; YIH, W. Dense Passage Retrieval for Open-Domain Question Answering. In: **Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)**, 2020. p.6769–6781.

KASNECI, G.; SUCHANEK, F. M.; IFRIM, G.; RAMANATH, M.; WEIKUM, G. NAGA: Searching and Ranking Knowledge. **2008 IEEE 24th International Conference on Data Engineering**, 2008, p.953-962.

KBPEDIA. Disponível em: <https://kbpedia.org/>. Acesso em: 01 dez. 2021.

KEJRIWAL, M.; KNOBLOCK; C. A.; SZEKELY, P. **Knowledge graphs: fundamentals, techniques and applications**. The MIT Press, 2021.

KERTKEIDKACHORN, N. NARARATWONG, R.; XU, Z.; ICHISE, R. FinKG: A Core Financial Knowledge Graph for Financial Analysis. **IEEE 17th International Conference on Semantic Computing (ICSC)**, p.90-93, 2023.

KHARYA, P.; ALVI, A. **Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, the World's Largest and Most Powerful Generative Language Model**. Microsoft Research Blog, 2021. Disponível em: <https://www.microsoft.com/en-us/research/blog/using-deepspeed-and-megatron-to-train-megatron-turing-nlg-530b-the-worlds-largest-and-most-powerful-generative-language-model/>. Acesso em: 29 nov. 2021.

KHATTAB, O.; POTTS, C.; ZAHARIA, M. Baleen: Robust Multi-Hop Reasoning at Scale via Condensed Retrieval. **ArXiv preprint**, arXiv:2101.00436, 2021b.

KHATTAB, O.; POTTS, C.; ZAHARIA, M. **Building Scalable, Explainable, and Adaptive NLP Models with Retrieval**, 2021c. Disponível em: <http://ai.stanford.edu/blog/retrieval-based-NLP/>. Acesso em: 04 dez. 2021.

KHATTAB, O.; POTTS, C.; ZAHARIA, M. Relevance-guided supervision for openqa with colbert. **Transactions of the Association for Computational Linguistics**, v.9, p.929-944, 2021a.

KIFER, M.; BOLEY, H. **RIF Overview (Second Edition), W3C Working Group Note 5 February 2013**. W3C Working Group Note. World Wide Web Consortium, 2013. Disponível em: <https://www.w3.org/TR/2013/NOTE-rif-overview-20130205/>. Acesso em: 16 out. 2021.

KIM, G.; KIM, S.; JEON, B.; PARK, J.; KANG, J. Tree of Clarifications: Answering Ambiguous Questions with Retrieval-Augmented Large Language Models. **ArXiv preprint**, arXiv: 2310.14696, 2023.

KIPF, T. N.; WELLING, M. Semi-supervised classification with graph convolutional networks. **ArXiv preprint**, arXiv:1609.02907, 2016.

KNOWLEDGE MINING API. **Microsoft Satori**, 2016. Disponível em: <https://www.microsoft.com/en-us/research/project/knowledge-mining-api/>. Acesso em: 01 dez. 2021.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Advances in neural information processing systems**, v.25, p.1097-1105, 2012.

KWIATKOWSKI, T.; PALOMAKI, J.; REDFIELD, O.; COLLINS, M.; PARIKH, A.; ALBERTI, C.; EPSTEIN, D.; POLOSUKHIN, I.; DEVLIN, J.; LEE, K.; TOUTANOVA, K. Natural questions: a benchmark for question answering research. **Transactions of the Association for Computational Linguistics**, v.1, n.7, p.453-66, 2019.

LAFFERTY, J.; MCCALLUM, A.; PEREIRA, F. C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: **ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning**, 2001. P.282-289.

LAMPLE, G.; CONNEAU, A. Cross-lingual language model pretraining. **ArXiv preprint**, arXiv:1901.07291, 2019.

LeCUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v.521, p.436-444, 2015.

LEHMANN, J.; ISELE, R.; JAKOB, M.; JENTZSCH, A.; KONTOKOSTAS, D.; MENDES, P. N.; HELLMANN, S.; MORSEY, M.; VAN KLEEF, P.; AUER, S.; BIZER, C. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. **Semantic Web Journal**, v.6, n.2, p.167–195, 2015.

LEWIS, P., PEREZ, E., PIKTUS, A., PETRONI, F., KARPUKHIN, V., GOYAL, N., KUTTLER, H.; LEWIS, M.; YIH, W.; ROCKTASCHHEL, T.; RIEDEL, S.; KIELA, D. Retrieval-augmented generation for knowledge-intensive NLP tasks. NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems, artigo 793, p.9459 – 9474, 2020.

LI, J.; SUN, A.; HAN, J.; LI, C. A survey on deep learning for named entity recognition. **ArXiv preprint**, arXiv:1812.09449v3, 2020.

LI, Z.; WANG, C.; LIU, Z.; WANG, H.; CHEN, D.; WANG, S.; GAO, C. CCTest: Testing and repairing code completion systems. **ICSE '23: Proceedings of the 45th International Conference on Software Engineering**, p.1238-1250, 2023.

LIFFITON, M.; SHEESE, B.; SAVELKA, J.; DENNY, P. CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes. **ArXiv preprint**, arXiv: 2308.06921, 2023.

LIN, B. Y.; CHEN, X.; CHEN, J.; REN, X. Kagnet: Knowledge-aware graph networks for commonsense reasoning. **ArXiv preprint**, arXiv:1909.02151v1, 2019.

LIN, C.; HOVY, E. Automatic evaluation of summaries using n-gram cooccurrence statistics. In: **Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)**. Edmonton, Canada, 2003. p.150–157.

LIN, C.; OCH, F. J. 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, 2004. p.605–612.

LIN, J.; ZHAO, A.; WU, Y.; PING, H.; CHEN, Q. AgentSims: An Open-Source Sandbox for Large Language Model Evaluation. **ArXiv preprint**, arXiv: 2308.04026, 2023.

LINNAINMAA, S. Taylor expansion of the accumulated rounding error. **BIT Numerical Mathematics**, v.16, p.146-160, 1976.

LIU, J.; LIU, C.; ZHOU, P.; LV, R.; ZHOU, K.; ZHANG, Y. Is ChatGPT a good recommender? A preliminar study. **ArXiv preprint**, arXiv: 2304.10149v3, 2023.

LIU, P.; YUAN, W.; FU, J.; JIANG, Z.; HAYASHI, H.; NEUBIG, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. **ArXiv preprint**, arXiv:2107.13586, 2021.

LONG, J. Large Language Model Guided Tree-of-Thought. **ArXiv preprint**, arXiv: 2305.08291, 2023.

MALINKA, K.; PERESINI, M.; FIRCI, A.; HUJNÁK, O.; JANUS, F. On the educational impact of ChatGPT: Is artificial intelligence ready to obtain a university degree? **ITiCSE 2023: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education**, p.47-53, 2023.

MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. **Introduction to information retrieval**. Cambridge University Press, 2008.

MARCHI, E.; MIGUEL, O. On the structure of the teaching-learning interactive process. **International Journal of Game Theory**, v.3, n.2, p.83–99, 1974.

MARCUS, G. The Next Decade in AI: Four Steps Towards Robust Artificial Intelligence. **ArXiv preprint**, arXiv:2002.06177, 2020.

MARTSCHAT, S.; STRUBE, M. Latent structures for coreference resolution. **Transactions of the Association for Computational Linguistics**, v.3, p.405-418, 2015.

MATURANA, R. A.; ALVARADO-CORTES, E.; LÓPEZ-SOLA, S.; MARTÍNEZ-LOSA, M. O.; HERMOSO-GONZÁLEZ, P. La Rioja Turismo: The Construction and Exploitation of a Queryable Tourism Knowledge Graph. In: **Current Trends in Web Engineering - ICWE 2018 International Workshops, MATWEP, EnWot, KD-WEB, WEOD, TourismKG**. Cáceres, Spain, 2018.

MEDHAT, W.; HASSAN, A.; KORASHY, H. Sentiment analysis algorithms and applications: A survey. **Ain Shams Engineering Journal**, v.5, n.4, p.1093-1113, 2014.

MIHAYLOV, T.; CLARK, P.; KHOT, T.; SABHARWAL, A. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In: **Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing**. Brussels, Belgium, 2018, p. 2381 – 2391.

MIHAYLOV, T.; FRANK, A. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge. In: **Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics**. Melbourne, Australia, 2018, p.821-832.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **ArXiv preprint**, arXiv:1301.3781, 2013a.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. S.; DEAN, J. Distributed representations of words and phrases and their compositionality. In: **NIPS'13: Proceedings of the 26th International Conference on Neural Information Processing Systems**, 2013b, p.3111-3119.

MIN, S.; ZHONG, V.; ZETTLEMOYER, L.; HAJISHIRZI, H. Multi-hop Reading Comprehension through Question Decomposition and Rescoring. **ArXiv preprint**, arXiv: 1906.02916, 2019.

MINAEE, S.; MIKOLOV, T.; NIKZAD, N.; CHENAGHLU, M.; SOCHER, R.; AMATRIAIN, X.; GAO, J. Large language models: a survey. **ArXiv preprint**, arXiv: **2402.06196**, 2024.

MITCHELL, Tom. **Machine Learning**. New York: McGraw Hill, 1997.

MITKOV, Ruslan. **The Oxford Handbook of Computational Linguistics**. Oxford University Press, 2005.

MUNGALL, C.; RUTTENBERG, A.; HORROCKS, I.; OSUMI-SUTHERLAND, D. **OBO Flat File Format 1.4 Syntax and Semantics**. Editor's Draft, 2012. Disponível em: <http://owllcollab.github.io/oboformat/doc/obo-syntax.html>. Acesso em: 16 out. 2021.

NADEAU, D.; SEKINE, S. A survey of named entity recognition and classification. **Linguisticae Investigationes**, v.30, n.1, p.3-26, 2007.

NAKANO, R.; HILTON, J.; BALAJI, S.; WU, J.; OUYANG, L.; KIM, C.; HESSE, C.; JAIN, S.; KOSARAJU, V.; SAUNDERS, W.; JIANG, X.; COBBE, K.; ELOUNDOU, T.; KRUEGER, G.; BUTTON, K.; KNIGHT, M.; CHESS, B.; SCHULMAN, J. WebGPT: Browser-assisted question-answering with human feedback. **ArXiv preprint**, arXiv:2112.09332, 2021.

NARAYANAN, A.; CHANDRAMOHAN, M.; VENKATESAN, R.; CHEN, L.; LIU, Y.; JAISWAL, S. Graph2vec: Learning distributed representations of graphs. **ArXiv preprint**, arXiv:1707.05005, 2017.

NEBULAGRAPH. **NebulaGraph Launches Industry-First Graph RAG: Retrieval-Augmented Generation with LLM Based on Knowledge Graphs**. NebulaGraph, 2023. Disponível em: <https://www.nebula-graph.io/posts/graph-RAG>. Acesso em: 22 jun. 2024.

NEO4J CYPHER. **The Cypher Query Language**. The Neo4j Team, 2020. Disponível em: <https://neo4j.com/developer/cypher/>. Acesso em: 02 mai. 2024.

NGUYEN, T.; ROSENBERG, M.; SONG, X.; GAO, J.; TIWARY, S.; MAJUMDER, R.; DENG, L. MS MARCO: A human generated machine reading comprehension dataset. **ArXiv preprint**, arXiv:1611.09268v1, 2016.

NITHYANANDAN, S.; RASEEK, C. Deep learning models for word sense disambiguation: A comparative study. In: **Proceedings of the International Conference on Systems, Energy & Environment (ICSEE)**. Kerala, India, 2019.

NOGUEIRA, R.; CHO, K. Passage Re-ranking with BERT. **ArXiv preprint**, arXiv:1901.04085, 2019.

NOY, N.; GAO, Y.; JAIN, A.; NARAYANAN, A.; PATTERSON, A.; TAYLOR, Industry-scale Knowledge Graphs: Lessons and Challenges. **Communications of the ACM**, v.62, n.8, p.36-43, 2019.

OGUNDARE, O.; MADASU, S.; WIGGINS, N. Industrial Engineering with Large Language Models: A case study of ChatGPT's performance on Oil & Gas problems. **ArXiv preprint**, arXiv: 2304.14354, 2023.

OPENAI. **Fine-tuning**, 2024. Disponível em: <https://platform.openai.com/docs/guides/fine-tuning>. Acesso em: 17 mai. 2024.

OPENAI. GPT-4 Technical Report. **ArXiv preprint**, arXiv: 2303.08774, 2023.

OPENAI. **Hello GPT-4o**. OpenAI, 2024. Disponível em: <https://openai.com/index/hello-gpt-4o/>. Acesso em: 20 mai. 2024.

OPENAI. New and improved embedding model. **OpenAI**, 2022. Disponível em: <https://openai.com/index/new-and-improved-embedding-model/>. Acesso em: 20 jul. 2024.

OUYANG, L.; WU, J.; JIANG, X.; ALMEIDA, D.; WAINWRIGHT, C. L.; MISHKIN, P.; ZHANG, C.; AGARWAL., SLAMA, K.; RAY, A.; SCHULMAN, J.; HILTON, J.; KELTON, F.; MILLER, L.; SIMENS, M.; ASKELL, A.; WELINDER, P.; CHRISTIANO., P.; LEIKE, J.; LOWE, R. Training language models to follow instructions with human feedback. **NIPS'22: Proceedings of the 36th International Conference on Neural Information Processing Systems**, p.27730-27744, 2022.

PAN, S.; LUO, L.; WANG, Y.; CHEN, C.; WANG, J.; WU, X. Unifying Large Language Models and Knowledge Graphs: A Roadmap. **IEEE Transactions on Knowledge and Data Engineering**, v.36, n.7, p.3580-3599, 2024.

PAPINENI, K.; ROUKOS, S.; WARD, T.; ZHU, W. 2002. Bleu: a method for automatic evaluation of machine translation. In: **Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)**. Philadelphia, USA, 2002. p.311–318.

PARISI, G. I.; RONALD, K.; PART, J. L.; KANAN, C.; WERMTER, S. Continual lifelong learning with neural networks: a review. **Neural Networks**, v.113, p.54-71, 2019.

PARK, J. S.; O'BRIEN, J. C.; CAI, C. J.; MORRIS, M. R.; LIANG, P.; BERNSTEIN, M. S. Generative Agents: Interactive Simulacra of Human Behavior. **ArXiv preprint**, arXiv: 2304.03442, 2023.

PASZKE, A.; GROSS, S.; MASSA, F.; LERER, A.; BRADBURY, J.; CHANAN, G.; KILLEEN, T.; LIN, Z.; GIMELSHEIN, N.; ANTIGA, L.; DESMAISON, A.; KOPF, A.; YANG, E.; DeVITO, Z.; RAISON, M.; TEJANI, A.; CHILAMKURTHY, S.; STEINER, B.; FANG, L.; BAI, J.; CHINTALA, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: **Advances in Neural Information Processing Systems**. Vancouver, Canada, 2019, p. 8024–8035.

PAULHEIM, H. Knowledge graph refinement: A survey of approaches and evaluation methods. **Semantic web**, v.8, n.3, p.489–508, 2016.

PENNINGTON, J.; SOCHER, R.; MANNING, C. Glove: Global vectors for word representation. In: **Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)**. Doha, Qatar, 2014, p.1532-1543.

PEROZZI, B.; AL-RFOU, R.; SKIENA, S. Deepwalk: Online learning of social representations. In: **KDD '14: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining**, 2014. p.701-710.

PETERS, M. E.; NEUMANN, M.; IYYER, M.; GARDNER, M.; CLARK, C.; LEE, K.; ZETTLEMOYER, L. Deep contextualized word representations. **ArXiv preprint**, arXiv:1802.05365, 2018.

POWERS, D. M. W. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. **ArXiv preprint**, arXiv:2010.16061, 2020.

PRENGER, R.; VALLE, R.; CATANZARO, B. WaveGlow: A Flow-based Generative Network for Speech Synthesis, **ArXiv preprint**, arXiv:1811.00002, 2018.

PYTORCH FAIRSEQ. **Beyond English-Centric Multilingual Machine Translation**, 2020. Disponível em: https://github.com/pytorch/fairseq/tree/main/examples/m2m_100. Acesso em: 25 out. 2021.

QIAN, C. et al. ChatDev: Communicative Agents for Software Development. **ArXiv preprint**, arXiv: 2307.07924, 2023.

QIN, Y. et al. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. **ArXiv preprint**, arXiv: 2307.16789, 2023.

QUINLAN, J. R. Induction of decision trees. **Machine learning**, v.1, n.1, p.81-106, 1986.

RABINER, L.; JUANG, B. An introduction to hidden Markov models. **IEEE ASSP Magazine**, v.3, n.1, p.4-16, 1986.

RADA, R. Gradualness eases refinement of medical knowledge. **Medical Informatics**, v.11, n.1, p.59–73, 1986.

RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. **Improving language understanding by generative pre-training**, 2018. Disponível em: <https://www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf>. Acesso em: 10 nov. 2021.

RADFORD, A.; WU, J.; CHILD, R.; LUAN, D.; AMODEI, D.; SUTSKEVER, I. **Language Models are Unsupervised Multitask Learners**, 2019. Disponível em: https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf. Acesso em: 02 jul. 2024.

RAE, J. W.; BERGEAUD, S.; CAI, T.; MILLICAN, K.; HOFFMANN, J.; SONG, F.; ASLANIDES, J.; HENDERSON, S.; RING, R.; YOUNG, S.; RUTHERFORD, E.; HENNIGAN, T.; MENICK, J.; CASSIRER, A.; POWELL, R.; VAN DEN DRIESCHE, G.; HENDRICKS, L. A.; RAUH, M.; HUANG, P.; GLAESE, A.; WELBL, J.; DATHATHRI, S.; HUANG, S.; UESATO, J.; MELLOR, J.; HIGGINS, I.; CRESWELL, A.; MCALEESE, N.; WU, A.; ELSER, E.; JAYAKUMAR, S.; BUCHATSKAYA, E.; BUDDEN, D.; SUTHERLAND, E.; SIMONYAN, K.; PAGANINI, M.; SIFRE, L.; MARTENS, L.; LI, X. L.; KUNCORO, A.; NEMATZADEH, A.; GRIBOVSKAYA, E.; DONATO, D.; LAZARIDOU, A.; MENSCH, A.; LESPIAU, J.; TSIMPOUKELLI, M.; GRIGOREV, N.; FRITZ, D.;

SOTTIAUX, T.; PAJARSKAS, M.; POHLEN, T.; GONG, Z.; TOYAMA, D.; D'AUTUME, C. M.; LI, Y.; TERZI, T.; MIKULIK, V.; BABUSCHKIN, I.; CLARK, A.; DE LAS CASAS, D.; GUY, A.; JONES, C.; BRADBURY, J.; JOHNSON, M.; HECHTMAN, B.; WEIDINGER, L.; GABRIEL, I.; ISAAC, W.; LOCKHART, E.; OSINDERO, S.; RIMELL, L.; DYER, C.; VINYALS, O.; AYOUB, K.; STANWAY, J.; BENNETT, L.; HASSABIS, D.; KAVUKCUOGLU, K.; IRVING, G. Scaling Language Models: Methods, Analysis & Insights from Training Gopher. **ArXiv preprint**, arXiv:2112.11446, 2021.

RAFFEL, C.; SHAZEER, N.; ROBERTS, A.; LEE, K.; NARANG, S.; MATENA, M.; ZHOU, Y.; LI, W.; LIU, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. **ArXiv preprint**, arXiv:1910.10683, 2019.

RAIMOND, Y.; FERNE, T.; SMETHURST, M.; ADAMS, G. The BBC World Service Archive prototype. **Journal of Web Semantics**, v.2, n.9, 2014.

RAJPURKAR, P.; ZHANG, J.; LOPYREV, K.; LIANG, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. **ArXiv preprint**, arXiv: 1606.05250, 2016.

RANDALL, D.; SHROBE, H.; SZOLOVITS, P. What is a knowledge representation? **AI magazine**, v.14, n.1, p.17-17, 1993.

RATNAPARKHI, A. **A simple introduction to maximum entropy models for natural language processing**. IRCS Technical Reports Series, 1997. Disponível em: https://repository.upenn.edu/ircs_reports/81/. Acesso em: 28 nov. 2021.

REDDY, S.; LAPATA, M.; STEEDMAN, M. Large-scale semantic parsing without question answer pairs. **Transactions of the Association for Computational Linguistics**, v.2, p.377-392, 2014.

REIMERS, N.; GUREVYCH, I. Making monolingual sentence embeddings multilingual using knowledge distillation. **ArXiv preprint**, arXiv:2004.09813, 2020.

REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. **ArXiv preprint**, arXiv:1908.10084, 2019.

REN, Y.; HU, C.; TAN, X.; QIN, T.; ZHAO, Z.; LIU, T. FastSpeech 2: Fast and High-Quality End-to-End Text to Speech, **ArXiv preprint**, arXiv:2006.04558, 2020.

RISCH, J.; MÖLLER, T.; GUTSCH, J.; PIETSCH, M. Semantic Answer Similarity for Evaluating Question Answering Models. **ArXiv preprint**, arXiv:2108.06130, 2021.

ROBERTSON, S.; ZARAGOZA, H. The Probabilistic Relevance Framework: BM25 and Beyond. **Foundations and Trends in Information Retrieval**, v.3, n.4, p.333-389, 2009.

RODRIGUEZ, M. A. The Gremlin graph traversal machine and language. In: **Proceedings of the 15th Symposium on Database Programming Languages, Pittsburgh**. Pensilvânia, USA, 2015. p.1–10.

ROY, A. Recent Trends in Named Entity Recognition (NER). **ArXiv preprint**, arXiv:2101.11420, 2021.

RUDOLPH, S. **The Semantic Web: Semantics and Big Data. ESWC 2013. Lecture Notes in Computer Science**, v.7882. Springer, Berlin, 2013.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. 1986. Learning internal representations by error propagation. In: RUMELHART, D. E.; MCCLELLAND, J. L. **Parallel distributed processing: explorations in the microstructure of cognition**. MIT Press, 1986.

RUSSEL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. New Jersey: Prentice Hall, 2003.

SANG, E.; F.; T.; K.; DE MEULDER, F. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In: **Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003**, 2003. P.142-147.

SCHNEIDER, E. W. Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis. In: **Association for the Development of Instructional Systems (ADIS)**. Chicago, Illinois, 1972.

SENTENCE-TTRANSFORMERS. **Sentence-transformers/multi-qa-mpnet-base-dot-v1**. Disponível em: <https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1>. Acesso em: 10 set. 2021.

SEQUADA, J. F.; ALLEMANG, D.; JACOB, B. A benchmark to understand the role of knowledge graphs on large language model's accuracy for question answering on enterprise SQL databases. **ArXiv preprint**, arXiv: 2311.07509v1, 2024.

SEVGILI, O.; SHELMANOV, A.; ARKHIPOV, M.; PANCHENKO, A.; BIEMANN, C. Neural entity linking: A survey of models based on deep learning. **ArXiv preprint**, arXiv:2006.00575, 2020.

SHEN, W.; WANG, J.; HAN, J. Entity linking with a knowledge base: Issues, techniques, and solutions. **IEEE Transactions on Knowledge and Data Engineering**, v.27, n.2, p.443-460, 2015.

SHIMONY, S. E.; DOMSHLAK, C.; SANTOS JR, E. Cost-Sharing in Bayesian Knowledge Bases. In: **Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence**. Rhode Island, 1997; p. 421–428.

SINGHAL, A. **Introducing the Knowledge Graph: things, not strings**, 2012. Disponível em: <https://blog.google/products/search/introducing-knowledge-graph-things-not/>. Acesso em: 02 nov. 2021.

SOON, W. M.; NG, H. T.; LIM, D. C. Y. A Machine Learning Approach to Coreference Resolution of Noun Phrases. **Computacional Linguistics**, v.27, n.4, 2001.

SPEER, R.; CHIN, J.; HAVASI, C. "ConceptNet 5.5: An Open Multilingual Graph of General Knowledge." In: **AAAI'17: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence**, p.4444-4451, 2017.

SRIKANTH, R.; JARKE, M. The Design of Knowledge-Based Systems for Managing Ill-Structured Software Projects. **Decision Support Systems**, v.5, n.4 , p.425–447,1989.

SRIVASTAVA, A. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. **ArXiv preprint**, arXiv: 2206.04615, 2022.

STADLER, C.; LEHMANN, J.; HÖFFNER, K.; AUER, S. LinkedGeoData: A core for a web of spatial open data. **Semantic Web Journal**, v.3, n.4, p.333–354, 2012.

STOKMAN, F. N.; DE VRIES, P. H. Structuring knowledge in a graph. In: VAN DER VEER, C G.; MULDER, G. **Human-Computer Interaction**. Springer, 1988.

STRICKLAND, E. **The Turbulent Past and Uncertain Future of AI**. IEEE Spectrum, 2021. Disponível em: <https://spectrum.ieee.org/history-of-ai>. Acesso em: 3 dez. 2021.

STRUBELL, E.; GANESH, A.; MCCALLUM, A. Energy and policy considerations for deep learning in NLP. **ArXiv preprint**, arXiv:1906.02243, 2019.

SUCHANEK, F. M.; KASNECI, G.; WEIKUM, G. Yago: A large ontology from wikipedia and wordnet. **Journal of Web Semantics**, v.6, n.3, p.203–217, 2008.

SURAN, S.; PATTANAIK, V.; DRAHEIM, D. Frameworks for collective intelligence: A systematic literature review. **ACM Computing Surveys**, v.53, n.1, p.1-36, 2020.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. Sequence to sequence learning with neural networks. **ArXiv preprint**, arXiv:1409.3215, 2014.

SWAMY, V.; ROMANOU, A.; JAGGI, M. Interpreting Language Models Through Knowledge Graph Extraction. **ArXiv preprint**, arXiv:2111.08546, 2021.

TALMOR, A.; BERANT, J. The Web as a Knowledge-base for Answering Complex Questions. **ArXiv preprint**, arXiv: 1803.06643, 2018.

TALMOR, A.; HERZIG, J.; LOURIE, N.; BERANT, J. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. **ArXiv preprint**, arXiv:1811.00937, 2018.

TANG, S.; ANDRES, B.; ANDRILUKA, M.; SCHIELE, B. Multi-Person Tracking by Multicut and Deep Matching. **ArXiv preprint**, arXiv:1608.05404, 2016.

THE DEREK BOK CENTER FOR TEACHING AND LEARNING. How memory works? **Harvard University**, 2024. Disponível em: <https://bokcenter.harvard.edu/how-memory-works>. Acesso em: 23 jul. 2024.

TRISEDYA, B. D.; WEIKUM, G. QI, J.; ZHANG, R. Neural relation extraction for knowledge base enrichment. In: **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**. Florence, Italy, 2019. p.229–240.

TRIVEDI, H.; BALASUBRAMANIAN, N.; KHOT, T.; SABHARWAL, A. Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. **ArXiv preprint**, arXiv: 2212.10509, 2022.

USHIO, A.; CAMACHO-COLLADOS, J. T-NER: An All-Round Python Library for Transformer-based Named Entity Recognition. In: **Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations**, 2021, p.53-62.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, T.; POLOSUKHIN, I. Attention is all you need. In: **Advances in neural information processing systems 30 (NIPS 2017)**, Long Beach, California, 2017. p.5998-6008.

VELIČKOVIĆ, P.; CUCURULL, G.; CASANOVA, A.; ROMERO, A.; LIO, P.; BENGIO, Y. Graph attention networks. **ArXiv preprint**, arXiv:1710.10903, 2017.

VOORHEES, E. M.; TICE, D. M. The TREC-8 Question Answering Track. **Proceedings of the Second International Conference on Language Resources and Evaluation**, 2000. Disponível em: <http://www.lrec-conf.org/proceedings/lrec2000/pdf/26.pdf>. Acesso em: 23 mar. 2024.

VRANDEČIĆ, D.; KRÖTZSCH, M. Wikidata: A Free Collaborative Knowledgebase. **Communications of the ACM**, v.57, n.10, p.78–85, 2014.

WAIBEL, A.; HANAZAWA, T.; HINTON, G.; SHIKANO, K.; LANG, K. J. Phoneme recognition using time-delay neural networks. **IEEE Transactions on Acoustics, Speech and Signal Processing**, v.37, n.3, p.328-339, 1989.

WANG, A.; SINGH, A.; MICHAEL, J.; HILL, F.; LEVY, O.; BOWMAN, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. **ArXiv preprint**, arXiv:1804.07461, 2018.

WANG, D.; MA, Z.; NOURBAKHS, A.; GU, K.; SHAH, S. DocGraphLM: Documental Graph Language Model for Information Extraction. **ArXiv preprint**, arXiv: **2401.02823v1**, 2024.

WANG, H.; FENG, S.; HE, T.; TAN, Z.; HAN, X.; TSVETKOV, Y. Can Language Models Solve Graph Problems in Natural Language? **ArXiv preprint**, arXiv: 2305.10037, 2023.

WANG, H.; QIN, K.; ZAKARI, R. Y.; LU, G.; YIN, J. Deep Neural Network Based Relation Extraction: An Overview. **ArXiv preprint**, arXiv:2101.01907, 2021.

WANG, Q.; MAO, Z.; WANG, B.; GUO, L. Knowledge graph embedding: A survey of approaches and applications,” **IEEE Transactions on Knowledge and Data Engineering**, v.29, n.12, p.2724–2743, 2017.

WANG, Z. Modern Question Answering Datasets and Benchmarks: A Survey. **ArXiv preprint**, arXiv: 2206.15030, 2022.

WEBBER, J. A programmatic introduction to neo4j. In: **Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity**, 2012, p.217-218.

WEI, J.; WANG, X.; SCHUURMANS, D.; BOSMA, M.; ICHTER, B.; XIA, F.; CHI, E.; LE, Q.; ZHOU, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. **ArXiv preprint**, arXiv: 2201.11903, 2022.

WEISSENBORN, D.; KOCISKY, T.; DYER, C. Dynamic integration of background knowledge in neural nlu systems. **ArXiv preprint**, arXiv:1706.02596, 2018.

WISEMAN, S.; RUSH, A. M.; SHIEBER, S. M. Learning global features for coreference resolution. **ArXiv preprint**, arXiv:1604.03035, 2016.

WU, Z.; HAN, C.; DING, Z.; WENG, Z.; LIU, Z.; YAO, S.; YU, T.; KONG, L. OS-Copilot: Towards Generalist Computer Agents with Self-Improvement. **ArXiv preprint**, arXiv: 2402.07456, 2023.

XI, Z.; CHEN, W.; GUO, X.; HE, W.; DING, Y.; HONG, B.; ZHANG, M.; WANG, J.; JIN, E.; ZHOU, E.; ZHENG, R.; FAN, X.; WANG, X.; XIONG, L.; ZHOU, Y.; WANG, W.; JIANG, C.; ZOU, Y.; LIU, X.; YIN, Z.; DOU, S.; WENG, R.; CHENG, W.; ZHANG, Q.; QIN, W.; ZHENG, Y.; QIU, X.; HUANG, X.; GUI, T. The Rise and Potential of Large Language Model Based Agents: A Survey. **ArXiv preprint**, arXiv: 2309.07864, 2023.

XIONG, L.; XIONG, C.; LI, Y.; TANG, K.; LIU, J.; BENNETT, P.; AHMED, J.; OVERWIJK, A. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. **ArXiv preprint**, arXiv: 2007.00808, 2020.

XU, D.; RUAN, C.; KORPEOGLU, E.; KUMAR, S.; ACHAN, K. Product Knowledge Graph Embedding for E-commerce. **WSDM '20: Proceedings of the 13th International Conference on Web Search and Data Mining**, p.672-680, 2020.

XU, X.; LIU, C.; SONG, D. SQLNet: Generating Structured Queries From Natural Language Without Reinforcement Learning. **ArXiv preprint**, arXiv: 1711.04436, 2017.

YADAV, V.; BETHARD, S. A Survey on Recent Advances in Named Entity Recognition from Deep Learning models. **Proceedings of the 27th International Conference on Computational Linguistics**, p.2145-2158, 2018.

YANG, B.; MITCHELL, T. Leveraging knowledge bases in lstms for improving machine reading. **ArXiv preprint**, arXiv: 1902.09091, 2019.

YANG, C.; CUI, H.; LU, J.; WANG, S.; XU, R.; MA, W.; YU, Y.; YU, S.; KAN, X.; LING, C.; FU, T.; ZHAO, L.; HO, J.; WANG, F. A Review on Knowledge Graphs for Healthcare: Resources, Applications, and Promises. **ArXiv preprint**, arXiv: 2306.04802, 2023.

YANG, Z. et al. HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. **ArXiv preprint**, arXiv: 1809.09600, 2018.

YAO, S.; YU, D.; ZHAO, J.; SHAFRAN, I.; GRIFFITHS, T. L.; CAO, Y.; NARASIMHAN, K. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. **ArXiv preprint**, arXiv: 2305.10601, 2023.

YASUNAGA, M. **Reasoning with language models and knowledge graphs for question answering**, 2021. Disponível em: <http://ai.stanford.edu/blog/qagnn/>. Acesso em: 12 nov. 2021.

YASUNAGA, M.; REN, H.; BOSSELUT, A.; LIANG, P.; LESKOVEC, J. QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering. **ArXiv preprint**, arXiv:2104.06378, 2021.

YIH, W.; CHANG, M. W.; HE, X.; GAO, J. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In: **Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing**. Beijing, China, 2015. p.1321-1331.

YUAN, L.; ZHANG, Z.; LI, L.; GUAN, C.; YU, Y. A survey of progress on cooperative multi-agent reinforcement learning in open environment. **ArXiv preprint**, arXiv: 2312.01058, 2023.

ZELLERS, R.; BISK, Y.; FARHADI, A.; CHOI, Y. From recognition to cognition: Visual commonsense reasoning. **ArXiv preprint**, arXiv: 1811.10830, 2019.

ZELLERS, R.; BISK, Y.; SCHWARTZ, R.; CHOI, Y. Swag: A large-scale adversarial dataset for grounded commonsense inference. **ArXiv preprint**, arXiv: 1808.05326, 2018.

ZENG, D.; LIU, K.; LAI, S.; ZHOU, G.; ZHAO, J. Relation classification via convolutional deep neural network. In: **Proceedings of COLING 2014, the 25th**

International Conference on Computational Linguistics: Technical Papers. Dublin, Ireland, 2014. p.2335-2344.

ZHAO, G.; SUN, X.; XU, J.; ZHANG, Z.; LUO, L. Muse: Parallel multi-scale attention for sequence to sequence learning. **ArXiv preprint**, arXiv:1911.09483, 2019.

ZHAO, J.; ZHUO, L.; SHEN, Y.; QU, M.; LIU, K.; BRONSTEIN, M.; ZHU, Z.; TANG, J. GraphText: Graph Reasoning in Text Space. **ArXiv preprint**, arXiv: 2310.01089, 2023.

ZHAO, X.; XIAO, F.; ZHONG, H.; YAO, J.; CHEN, H. Condition aware and revise transformer for question answering. In: **WWW '20: Proceedings of The Web Conference 2020**, 2020. p.2377–2387.

ZHOU, X.; LI, G.; LIU, Z. LLM As DBA. **ArXiv preprint**, arXiv: 2308.05481, 2023.

ZHOU, Y.; HE, Y.; TIAN, S.; NI, Y.; YIN, Z.; LIU, X.; JI, C.; LIU, S.; QIU, X.; YE, G.; CHAI, H. R3-NL2GQL: A Model Coordination and Knowledge Graph Alignment Approach for NL2GQL. **ArXiv preprint**, arXiv: 2311.01862, 2023.

ZHU, Y.; LI, X.; LIU, C.; ZOLFAGHARI, M.; XIONG, Y.; WU, C.; ZHANG, Z.; TIGHE, J.; MANMATHA, R.; LI, M. A Comprehensive Study of Deep Video Action Recognition. **ArXiv preprint**, arXiv:2012.06567, 2020.