VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

REPORT
CAPSTONE PROJECT

MOTION PLANNING AROUND OBSTACLES

MAJOR: COMPUTER SCIENCE

THESIS COMMITTEE : COMPUTER ENGINEERING

SUPERVISOR(s) : LE HONG TRANG, Assoc. Prof

:

————o0o————

STUDENT : NGUYEN TRAN DANG KHOA - 2211635

: HO DANG KHOA - 2211

HO CHI MINH CITY, November 2025

Declaration Of Authenticity

I declare that I solely conducted this specialized project, under the supervision of Assoc. Prof. Le Hong Trang at the Faculty of Computer Science and Engineering, Vietnam National University - Ho Chi Minh City University Of Technology.

I have taken care to properly acknowledge and document all external sources and references used in the project.

If there is any instance of plagiarism, I am ready to accept the consequences. Ho Chi Minh City University of Technology - Vietnam National University HCMC will not be held responsible for any copyright violations that may have occurred during my research.

Ho Chi Minh City, November 2025

Author,

Nguyen Tran Dang Khoa

Acknowledgement

I would like to express my appreciation to Assoc. Prof. Le Hong Trang and Miss. for their support and invaluable guidance. My research has greatly benefited from their deep knowledge, perceptive criticism, and constant support. In addition, I would like to extend my gratitude to my family. Their unwavering faith in my abilities and constant encouragement have been my pillars of strength. Their belief in my potential has been a constant source of motivation and resilience. I am eternally grateful for their love and support.

Abstract

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In chapter 1, the overview, objectives and goals of the research's project are illustrated. The outline of the report is also presented.

1.1 Motivation

Trajectory optimization offers mature tools for motion planning in high-dimensional spaces under dynamic constraints. However, when facing complex configuration spaces, cluttered with obstacles, roboticists typically fall back to sampling-based planners that struggle in very high dimensions and with continuous differential constraints. Indeed, obstacles are the source of many textbook examples of problematic nonconvexities in the trajectory-optimization prob- lem. Here we show that convex optimization can, in fact, be used to reliably plan trajectories around obstacles. Specifically, we consider planning problems with collision-avoidance con- straints, as well as cost penalties and hard constraints on the shape, the duration, and the velocity of the trajectory. Combining the properties of Bézier curves with a recently-proposed framework for finding shortest paths in Graphs of Convex Sets (GCS), we formulate the planning problem as a compact mixed-integer optimization. In stark contrast with existing mixed-integer planners, the convex relaxation of our programs is very tight, and a cheap round- ing of its solution is typically sufficient to design globally-optimal trajectories. This reduces the mixed-integer program back to a simple convex optimization, and au-

tomatically provides optimality bounds for the planned trajectories. We name the proposed planner GCS, after its underlying optimization framework. We demonstrate GCS in simulation on a variety of robotic platforms, including a quadrotor flying through buildings and a dual-arm manipulator (with fourteen degrees of freedom) moving in a confined space. Using numerical experiments on a seven-degree-of-freedom manipulator, we show that GCS can outperform widely-used sampling-based planners by finding higher-quality trajectories in less time.

1.2 Goals

1.3 Scope

This project focuses on improving the factual accuracy of LLMs in English one-hop question answering tasks. The research will be conducted within the following constraints:

- One-Hop Question Answering: The project will solely focus on questions answerable through a single relationship traversal within the knowledge graph. Multi-hop reasoning or questions requiring complex inference beyond one hop are outside the scope
- English Language: The project will be limited to English text and knowledge graphs. Multilingual question answering or knowledge graphs with other languages will not be considered
- Static Knowledge Graph: The knowledge graph utilized in this project will be static. Dynamically updating the knowledge graph with new information or incorporating a knowledge graph completion component is beyond the scope
- Limited Computational Resources: Due to computational resource limitations, the size of the knowledge graph and the scale of the LLM used may be constrained

1.4 Thesis structure

There are five chapters in this capstone project:

- Chapter 1 briefly describes the project about problem's motivation, as well as the project's objectives and scope
- Chapter 2 is dedicated to presenting the foundational knowledge about Natural Language Processing (NLP) utilized in the project
- Chapter 3 arranges the discussion of related works on the same task to deepen the understanding of existing methods and their constraints.
- Chapter 4 depicts approach to reproduce RAG, KG-RAG, and illustrates the KEALLM and evaluate the performance
- Chapter 5 summarizes whole project and the plan for future development.

Chapter 2

Theoretical Background

In Chapter 2, it presents about preliminary knowledge in this project.

2.1 Cosine similarity

Cosine similarity is a metric that measures the similarity between two vectors of an inner product space. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The value of cosine of 0 degree is 1, and it is less than 1 for the angles between $(0, \pi)$ radians. In the field of natural language processing, each document is projected in a multi-dimensional space. When using cosine similarity to compare these documents with each other, both vectors are normalized to 1, and the similarity of documents is computed independent of the size of the documents[1].
The formula to find the cosine similarity between two vectors is

$$Similarity(\vec{a},\vec{b}) = cos(\theta) = \frac{\vec{a}\cdot\vec{b}}{||\vec{a}||\,||\vec{b}||} \tag{2.1}$$

2.2 Transformer

Transformer is a deep learning architecture showed in figure 2.1 that was introduced in 2017 by Vaswani et al [2]. It builds on the ideas presented in two influ-

ential researchs, sequence to sequence [3], [4] and attention mechanism [5], and has become the state-of-the-art architecture for various natural language processing tasks.

The Transformer Encoder is composed of multiple layers. The first layer is Multi-head Attention, and the second layer is Position-wise Feed-forward network. In the Encoder Self-Attention, Queries, Keys, and Values are all derived from the outputs of the previous Encoder Layer. Both sublayers use a residual connection, inspired by the ResNet architecture [6]. This additional residual connection is immediately followed by Layer Normalization [7]. As a result, the Transformer Encoder produces a vector representation for each position of the input sequence.

The Transformer Decoder is also a stack of multiple layers. In addition to the two sublayers described in the Encoder, the Decoder includes a third sublayer called the Encoder-Decoder Attention, or Cross-Attention, between these two. In the Encoder-Decoder Attention, Queries are derived from the outputs of the previous Decoder Layer, while the Keys and Values are derived from the Transformer Encoder outputs. In the Decoder Self-Attention, Queries, Keys, and Values are derived from the outputs of the previous Decoder Layer. However, each position in the Decoder can only attend to all positions in the Decoder up to that position, which is called Masked Self-Attention. This Masked Self-Attention preserves the autoregressive property, ensuring that the prediction only depends on those output tokens that have been generated.

In summary, the transformer model is a powerful and flexible architecture that owes its success to the combination of the sequence to sequence learning approach and the attention mechanism.

Scaled Dot-Product Attention: It computes how much each value vector con-

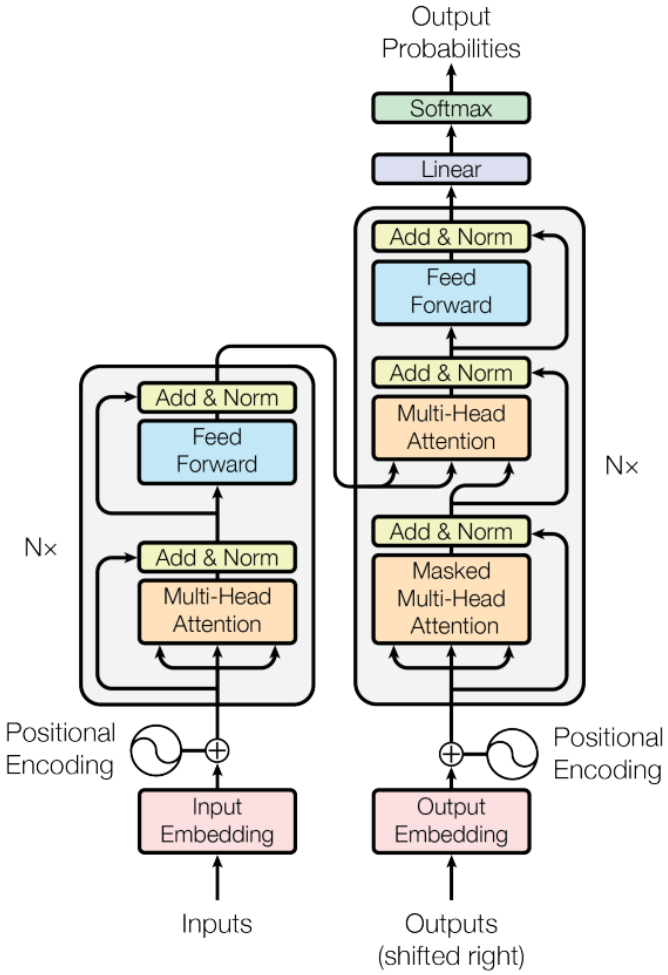

Figure 2.1: The architecture of transformers presented in the research paper "Attention is all you need" [2]

tributes to the output vector, based on the similarity between the query vector Q and the key vector Q. The similarity is measured by the dot product, which is scaled down by the square root of the dimension of the key vector d_k. The scaled dot products are then normalized by a softmax function to get the attention weights. The output vector is the weighted sum of the value vectors.

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{(d_k)}})V \tag{2.2}$$

Multi-head Attention: It applies the attention function multiple times in parallel, each of them is called a head, using different projections of the query, key and value vectors. This allows the model to capture different aspects of the input

and output sequences. The outputs of the multi-head attention are concatenated and projected to get the final output.

$$MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O \tag{2.3}$$

$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2.4}$$

Position-wise Feed-forward Networks: This is a way of transforming the output of the attention layers using a simple neural network that consists of two linear layers with a ReLU activation in between. The network is applied to each position separately and identically, meaning that it does not depend on the order of the sequence. The network has the same input and output dimension as the attention layers, but a larger hidden dimension.

$$FFN(x) = max(0, x \cdot W_1 + b_1) \cdot W_2 + b_2 \tag{2.5}$$

2.3 Large Language Models

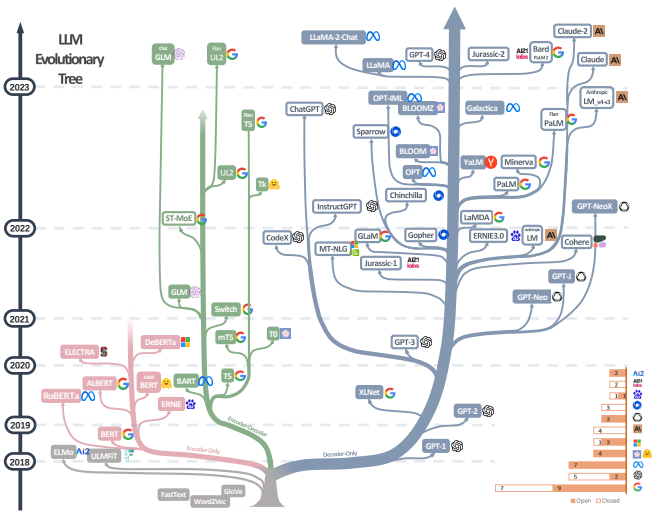

Figure 2.2: The evolutionary tree of modern LLMs traces the development of language models in recent years [8]

Large Language Models that are pre-trained on large-scale corpus have demon-

strated significant potential in various natural language processing tasks . Most LLMs are based on the Transformer design, which consists of encoder and decoder modules that are empowered by a self-attention mechanism. Based on their architecture structure, LLMs can be classified into three categories: encoder-only LLMs, encoder-decoder LLMs, and decoder-only LLMs. Figure 2.2 summarizes several representative LLMs with different model architectures, model sizes, and open-source availabilities.

Encoder-only LLMs are a type of Large Language Models that use only the encoder to encode the sentence and understand the relationships between words. These models are typically pre-trained by predicting the mask words in an input sentence using large-scale corpus. To resolve downstream tasks, such as text classification and named entity recognition, an extra prediction head is added to encoder-only LLMs like BERT [9], RoBERTa [10], ALBERT [11], and ELECTRA [12]. These models are most effective for tasks that require understanding the entire sentence.

Encoder-decoder LLMs adopt both the encoder and decoder module. The encoder module is responsible for encoding the input sentence into a hidden space, and the decoder is used to generate the target output text. The training strategies in encoder-decoder LLMs can be more flexible. Encoder-decoder LLMs are able to directly resolve tasks that generate sentences based on some context, such as summariaztion, translation, and question answering.

Decoder-only LLMs only adopt the decoder module to generate target output text. These models are trained to predict the next word in a sentence. They can perform downstream tasks with few examples or simple instructions without adding prediction heads or fine-tuning. [13]. Many state-of-the-art LLMs follow the decoder-only architecture. Recently Llama-2 [14], Alpaca [1] and Vicuna

[1] https://crfm.stanford.edu/2023/03/13/alpaca.html

[2] are released as open-source decoder-only LLMs. These models are finetuned based on LLaMA [15] and achieve comparable performance with ChatGPT [16] and GPT-4

2.4 Knowledge graph

Knowledge graphs (KGs) store structured knowledge as a collection of triples $KG = \{(h,r,t) \subseteq E \times R \times E\}$, where E and R respectively denote the set of entities and relations [17]. There are four types of existing KGs based on the stored information: Encyclopedic KGs, Commonsense KGs, Domain-specific KGs, and multi-modal KGs.

Encyclopedic KGs are the most common type of KGs that store general knowledge about the world. They are usually created by integrating information from various sources such as human experts, encyclopedias, and databases. Wikidata is one of the most popular encyclopedic KGs, which contains a wide range of knowledge extracted from Wikipedia articles [18].

Commonsense KGs are designed to represent knowledge about everyday concepts such as objects and events, as well as their relationships. Unlike encyclopedic knowledge graphs, commonsense knowledge graphs often model tacit knowledge extracted from text, such as the relationship between "Car", "UsedFor", and "Drive". ConceptNet [19], one of the most popular KGs, is a commonsense knowledge graph that contains a wide range of concepts and relations, which can help computers understand the meanings of words people use.

Domain-specific KGs are created to represent knowledge in a particular domain, such as medicine, biology, or finance. These knowledge graphs are often smaller in size than encyclopedic knowledge graphs, but they are more accurate

[2] https://lmsys.org/blog/2023-03-30-vicuna/

and reliable. As an instance of a domain-specific knowledge graph, the Unified Medical Language System (UMLS) is a knowledge graph that contains biomedical concepts and their relationships [20].

Multi-modal KGs are different from conventional knowledge graphs in that they represent facts in multiple modalities such as images, sounds, and videos. For instance, IMGpedia [21], MMKG [22], and Richpedia [23] are examples of multi-modal knowledge graphs that incorporate both text and image information into the knowledge graphs.

2.5 Knowledge graph embedding

Knowledge graph embedding (KGE) aims to represent entities and relations in a knowledge graph as low-dimensional vectors, capturing their semantic relationships. This section focuses on two prominent approaches for KGE: translation models and pretrained language models.

2.5.1 Translation models

Translation models, pioneered by TransE [24], are based on the idea of representing relations as translations in the embedding space. For a triple (h,r,t) representing head entity h, relation r, and tail entity t, TransE enforces the following relation:

$$\mathbf{h}+\mathbf{r}\approx\mathbf{t} \tag{2.6}$$

where $\mathbf{h}$, $\mathbf{r}$, and $\mathbf{t}$ are the embedding vectors of h, r, and t, respectively. This implies that the embedding of the tail entity should be close to the embedding of the head entity translated by the relation vector. TransE utilizes a margin-based ranking loss to encourage correct triples to have lower scores than incorrect ones. Several extensions to TransE have been proposed to address its

limitations in modeling complex relations like one-to-many, many-to-one, and many-to-many. These include TransH [25], TransR [26], and TransD [27], which introduce relation-specific hyperplanes, projection matrices, and dynamic mapping matrices, respectively, to handle diverse relation patterns.

2.5.2 Pretrained language models

Pretrained Language Models (PLMs), such as BERT [9] and RoBERTa [10], have demonstrated remarkable capabilities in capturing contextualized word representations. Recently, PLMs have been adapted for knowledge graph embedding by treating entities and relations as special tokens within the language model. KG-BERT [28] utilizes BERT for link prediction by transforming it into a masked entity prediction task. Given an incomplete triple $(h, r, ?)$, KG-BERT constructs an input sequence:

$$x_t = \texttt{[CLS]}\, h\, \texttt{[SEP]} r\, \texttt{[SEP]}$$

The model then predicts the masked entity by ranking the probability of each entity in the knowledge graph:

$$p(t|x_t) = p(\texttt{[CLS]} = t|x_t; \Phi) \tag{2.7}$$

where Φ represents the model's parameters. KG-BERT is trained using a cross-entropy loss, encouraging the model to assign high probability to the correct tail entity. This approach effectively leverages the rich semantic information captured by PLMs to learn knowledge graph embeddings. Moreover, it allows for incorporating textual descriptions associated with entities and relations, further enhancing the expressiveness of the embeddings. By utilizing translation models or pretrained language models, knowledge graph embedding provides a powerful framework for representing and reasoning over knowledge graphs, enabling various downstream tasks such as link prediction, question answering, and rec-

ommendation systems.

2.6 Direct preference optimization

Direct preference optimization (DPO) is a learning paradigm that fine-tunes language models to align with user preferences, as introduced in [29]. Unlike traditional Reinforcement Learning from Human Feedback (RLHF) methods [30] that require training a separate reward model and then using reinforcement learning to optimize the policy, DPO directly optimizes the policy to match human preferences using a simple classification objective. The key insight of DPO is that the optimal policy for a KL-regularized reward maximization objective can be expressed in closed form. This allows us to directly parameterize the policy in a way that implicitly encodes the reward function, bypassing the need for a separate reward model. Consider a language model policy $\pi_\theta(y|x)$ that generates response y given prompt x and is parameterized by θ. We want to optimize this policy to maximize a reward function $r(x,y)$ while staying close to a reference policy $\pi_{ref}(y|x)$ (usually the initial supervised fine-tuned model). The standard RLHF objective is:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim D, y\sim\pi_\theta(y|x)}[r(x,y)] - \beta D_{KL}[\pi_\theta(y|x)||\pi_{ref}(y|x)] \tag{2.8}$$

where β controls the strength of the KL penalty. The optimal solution to this objective has the following form:

$$\pi_r(y|x) = \frac{1}{Z(x)}\pi_{ref}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right) \tag{2.9}$$

where $Z(x)$ is the partition function. DPO leverages this closed-form solution by rearranging it to express the reward function in terms of the optimal policy and

the reference policy:

$$r(x,y) = \beta \log \frac{\pi_r(y|x)}{\pi_{ref}(y|x)} + \beta \log Z(x) \tag{2.10}$$

Substituting this expression for the reward function into a preference model (like the Bradley-Terry model) allows us to express the probability of human preferences directly in terms of the policy and the reference policy. Crucially, the partition function $Z(x)$ cancels out, leaving a tractable objective. For the Bradley-Terry model, the probability of preferring response y_w over y_l given prompt x is:

$$p(y_w \succ y_l|x) = \sigma\left(r(x,y_w) - r(x,y_l)\right) \tag{2.11}$$

Substituting the reparameterized reward function and simplifying, we get:

$$p(y_w \succ y_l|x) = \sigma\left(\beta \log \frac{\pi_r(y_w|x)}{\pi_{ref}(y_w|x)} - \beta \log \frac{\pi_r(y_l|x)}{\pi_{ref}(y_l|x)}\right) \tag{2.12}$$

This leads to the DPO loss function, which is a simple binary cross-entropy loss:

$$\mathcal{L}_{DPO} = -\mathbb{E}_{(x,y_w,y_l)\sim D} \log \sigma\left(\beta\left(\log \frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)}\right)\right) \tag{2.13}$$

where D is the dataset of human preference data. By minimizing this loss, we directly optimize the policy π_θ to match the observed preferences, implicitly learning a reward function that aligns with those preferences. This eliminates the need for a separate reward model and RL, making DPO a simpler and more efficient approach compared to traditional RLHF methods.

Chapter 3

Related works

Large language models have emerged as a powerful tool for a variety of tasks. These models, trained on vast amounts of data, have demonstrated remarkable capabilities in understanding and generating human-like text [31]. However, despite their impressive performance, these models are not without their limitations. The need to enhance their knowledge base and improve their efficiency and accuracy is a pressing concern. In this context, Chapter 3 delves into several related works that explore different approaches to address these challenges. The chapter provides a comprehensive overview of these methods, highlighting their strengths and weaknesses, and sets the stage for our proposed solution.

3.1 Fine-tuning Large Language Model

Several recent works have explored the potential of fine-tuning for adapting large language models (LLM) to specific domains and tasks.

A research about adapting language models to domains and tasks [32] investigates whether it is beneficial to tailor a pretrained language model to the domain of a target task, by continuing to pretrain on domain-specific and task-specific data. The paper presents a study across four domains (biomedical and computer science publications, news, and reviews) and eight classification tasks, show-

ing that a second phase of pretraining in-domain (domain-adaptive pretraining) leads to performance gains, under both high- and low-resource settings.

Fine-tuning is a common technique to adapt large language models to specific tasks or domains. However, fine-tuning has a drawback about high computational and storage costs. To address these issues, parameter-efficient fine-tuning (PEFT) methods [33]–[36] have been proposed, which only fine-tune a small subset of parameters while freezing the rest of the LLM. PEFT methods can reduce the resource requirements, preserve the knowledge of the LLM, and improve the robustness of the model.

3.2 Retrieval-augmented generation

3.2.1 In-context learning

In-context learning is an emergent behavior of Large Language Models that allows them to perform various tasks based on a few input-output examples, without optimizing any parameters. This phenomenon has been observed in LLMs such as GPT-2 [37] and GPT-3 [31], which have been pre-trained on massive text corpora using a simple objective of predicting the next token given the preceding text. In-context learning relies on the ability of LLMs to store and access factual knowledge in their parameters, and to adapt to different input distributions and output formats by conditioning on the prompt.

3.2.2 Information retrieval

Information retrieval (IR) is the task of finding relevant information from a large collection of documents given a query. IR is an essential component for many NLP applications, such as question answering, dialogue systems. Traditionally, IR methods rely on sparse representations and term-matching techniques, such as TF-IDF [38] and BM25 [39], to rank documents based on their similarity to

the query. However, these methods have limitations in capturing semantic and contextual information, and may fail to retrieve relevant documents that do not share common words with the query.

To overcome these limitations, recent works have explored the use of dense representations and neural models for IR. In particular, three types of neural architectures based on pre-trained transformer models with the same architecture of BERT [9] have been proposed: Bi-encoder, Cross-encoder, and Poly-encoder [40]. Bi-encoder is efficient and scalable, as it can index and compare the encoded documents using cosine similarity. However, it may not capture the fine-grained interactions between the query and the document. In cross-encoder, the query and the document are concatenated and passed to a cross-encoder model, which produces a relevance score as the output. Cross-encoder is more accurate and expressive, as it can perform full attention over the query and the document. However, it is slow and impractical, as it has to recompute the encoding for each query-document pair. Poly-encoder combines the advantages of bi-encoder and cross-encoder models by using two separate model, one for the query and one for the document, and applying attention between them only at the top layer. Poly-encoder can achieve better performance than bi-encoder and faster speed than cross-encoder.

3.2.3 Unifying retriever and Large Language Model

One of the challenges of knowledge-intensive tasks, such as open-domain question answering and fact-based text generation, is how to access and integrate external knowledge sources with LLMs. A common approach is to use a two-stage pipeline, where the first stage is a retriever that selects relevant documents or passages from a large corpus, such as Wikipedia, and the second stage is a reader or a generator that produces the answer or the text based on the retrieved information illustrated in Figure 3.1.

A recent line of work [41] has proposed to unify the retriever and the LLM

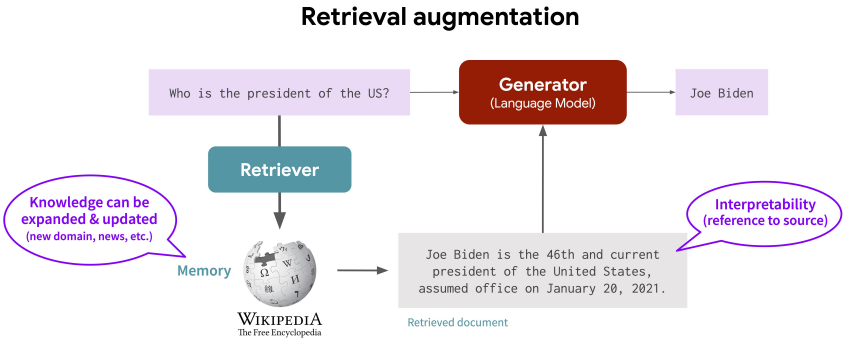

Figure 3.1: Illustration of two-stage pipeline in retrieval-augmented generation

in a single end-to-end model, which can jointly learn to retrieve and generate, which combines a pre-trained language model (PLM), BART, with a Dense Passage Retriever (DPR) following the Bi-encoder architecture. This model can use the input sequence to retrieve relevant passages from Wikipedia and use them as additional context when generating the output sequence.

3.3 Graph reasoning

Graph reasoning is a technique that uses knowledge graphs to enhance the reasoning capabilities of language models. Knowledge graphs are structured representations of entities and their relations, which can provide rich semantic information. Graph reasoning can leverage the graph structure and the logic rules to infer new knowledge, answer complex queries, and discover implicit relationships. Some of the related works that use graph reasoning for knowledge-aware question answering.

Multi-hop graph relation network [42]: This research proposes a knowledge-aware approach that equips pre-trained language models with a multi-hop relational reasoning module. It performs multi-hop, multi-relational reasoning over subgraphs extracted from external knowledge graphs. The proposed reasoning

module unifies path-based reasoning methods and graph neural networks to achieve better interpretability and scalability.

QA-GNN [43]: This work presents an end-to-end question answering model that jointly reasons over the knowledge from pre-trained language models and knowledge graphs through graph neural networks. The model addresses two challenges: identifying relevant knowledge from large knowledge graphs, and performing joint reasoning over the QA context and knowledge graph. It uses relevance scoring to estimate the importance of knowledge graph nodes relative to the given QA context, and connects the QA context and knowledge graph to form a joint graph, mutually updating their representations through graph-based message passing

3.4 Discussion

In this chapter, we have seen three distinct approaches to enhance the capabilities of Large Language Models. The first approach, fine-tuning Large Language Models, adapts a pre-trained model to a specific domain or task. This method can improve the model's performance in the targeted domain or task and leverage the general knowledge of the pre-trained language model. However, it can be costly in terms of computational resources and data requirements, and may lead to catastrophic forgetting or poor generalization to out-of-domain scenarios.

The second approach, retrieval-augmented generation, combines a pre-trained language model with a dense retriever that selects relevant information from a large corpus. This method can address the factuality issue of the Large Language Models, efficiently access and integrate external knowledge sources without fine-tuning. However, retrieval-augmented generation may have difficulty exploring a vast range of documents, especially if they are diverse, noisy, or outdated. Retrieval-augmented generation may also miss important information

that is not explicitly stated in the documents.

The third approach, graph reasoning leverages knowledge graphs to enhance the reasoning capabilities of pretrained language models. While knowledge graphs offer rich semantic information through structured representations of entities and relations, effectively integrating this information poses significant challenges. These challenges include identifying relevant knowledge within large knowledge graphs, performing joint reasoning over both natural language and graph inputs, and inferring implicit knowledge not explicitly stated in the graph.

Inspired by the strengths and limitations of existing approaches, this project aims to develop a method that efficiently and accurately augments the knowledge of a Large Language Model. We recognize the potential of knowledge graph embedding to address these challenges. Knowledge graph embedding compactly represents entities and relations as vectors, capturing their semantic relationships and enabling efficient knowledge integration within the LLM framework. This approach allows us to leverage the rich information contained in knowledge graphs while mitigating the challenges of missing or implicit knowledge that hinder other methods.

Chapter 4

Proposed Solution

As mentioned in Chapter 3, the enhancement of knowledge of Large Language Models is a crucial task for improving their performance on various natural language processing tasks. Chapter 4 presents the proposed solution for enhancing the knowledge of LLMs using retrieval-augmented generation and knowledge graphs.

4.1 Baseline methods

4.1.1 Text-based retrieval augmented-generation

Figure 4.1 illustrated the pipeline of Retrieval-augmented generation. It includes *Latent projection*, *Retrieval*, *Context pruning*, *Generator*.

Preprocessing data: Knowledge base contains short sentences. Each sentence is projected to latent space by using Bi-encoder in order to construct vector database.

Latent projection: The Bi-encoder is employed to map the query from the user to the same latent space of documents in vector database.

Retrieval: In this phase, cosine similarity is used to compare between each embedding vector of documents in vector database and embedding vector of query to select top-K documents based on similarity score.

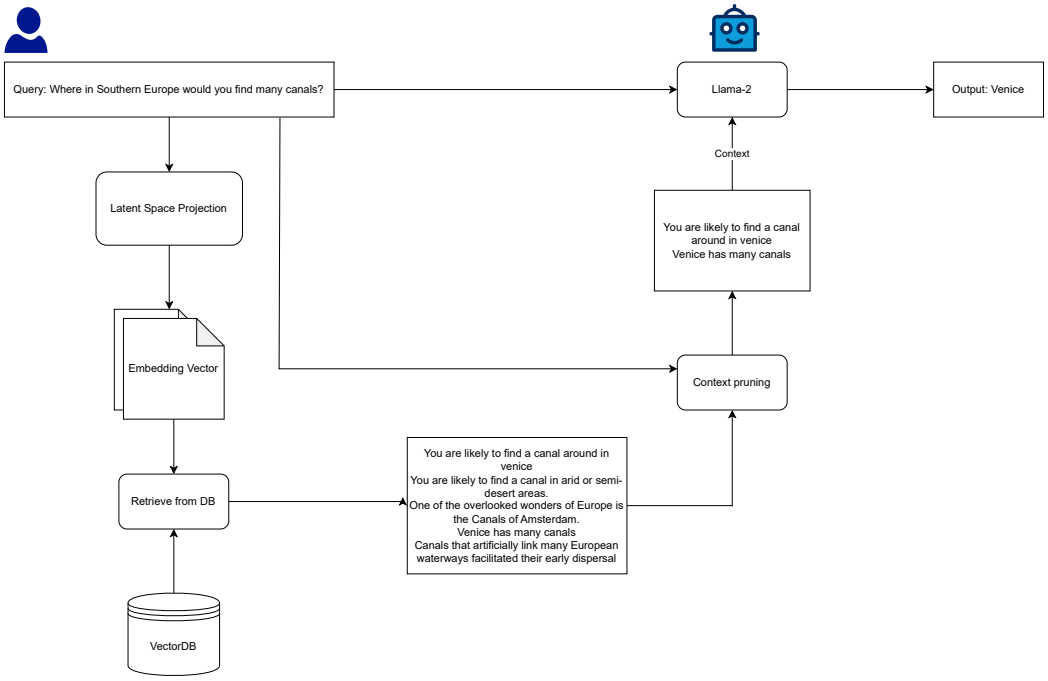

Figure 4.1: General 2-stage pipeline of retrieval-augmented generation

Context pruning: In this phase, the main objective is to perform reranking on the top-K documents after retrieval. This process helps to select the context documents more precisely and relevantly. The Cross-encoder is employed to compare the query with each top-K document. After that, top-K' documents (where K' $<$ K) are extracted and passed to the generator to generate the answer for the user.

4.1.2 Knowledge graph-based retrieval-augmented generation

Figure 4.2 illustrated the pipeline of Knowledge graph-based retrieval-augmented generation. It consists of *Entity extraction*, *Mapping entities*, *Context pruning*, *Multi-hop retrieval on KG*, *Generator*.

Preprocessing data: Each entity in knowledge graph is projected to latent space by using Bi-encoder in order to construct vector database.

Entity extraction: Named Entity Recognition (NER) is a fascinating task that involves identifying and classifying entities like names of people, organizations, locations, and other important terms within a given text. In this project, Llama-2 is used as an NER tool to detect entities in a sentence. The sample generation

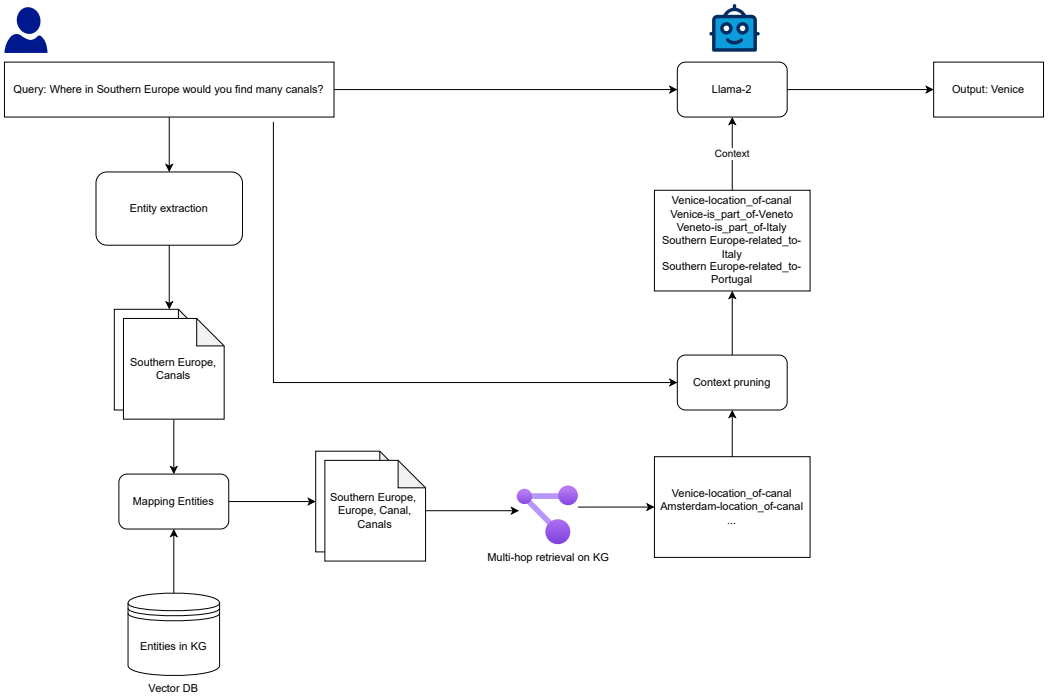

Figure 4.2: Pipeline of Knowledge graph-based retrieval-augmented generation can be referred in Appendix A.

Mapping entities: In this project, the Bi-encoder is used to map the extracted entities to the same latent space as the entities in the vector database, then compared with each entities' vector by cosine similarity. This phase helps to correct the entity, as in some cases, the extracted entity may not exist in the knowledge graph. Therefore, it can be found a entity which is similar to the extracted entity. If there are no extracted entities, the user's query can be used to find relevant entities by mapping the query to the latent space and then comparing it with each entity's vector.

Multihop-retrieval on KG: The next step is to find the relevant knowledge on the knowledge graph after mapping the entities. This can be done by selecting the top-K neighbor hops for each entity. The extracted knowledge is represented as triplets of the form (<Entity>,<Relation>,<Entity>). Each triplet is then converted into a sentence as shown in Figure 4.2.

Context pruning: Cross-encoder is employed to make comparison between query and each triplet, and after that top-K' triplets are extracted based on the similarity score and passed to generator to answer the the query from user.

4.2 Proposed solution: KEALLM - Knowledge Embedding Augmented Large Language Model

4.2.1 Architecture

This architecture, illustrated in Figure 4.3, seeks to combine the powerful language processing capabilities of a Large Language Model (LLM) with the rich contextual information embedded within a knowledge graph. The primary goal is to leverage both elements for enhanced language comprehension and generation.

The process begins with a user query X_q. This query is simultaneously fed into two distinct pathways. The first pathway involves a standard Embedding Layer, which converts the raw text of the query into language embedding tokens H_q. These tokens represent the semantic meaning of the query in a suitable format by the LLM.

The second pathway leverages the knowledge graph to provide contextual information. We assume that each user query X_q is associated with a corresponding query triplet (*head*, *relation*, ?), representing the knowledge sought by the query. This triplet is used to query the Knowledge Graph Embedding module, which extracts relevant knowledge features X_{kn}. These features encapsulate the relationships and entities within the knowledge graph that belong to the user's query. However, these knowledge features exist in a form incompatible with the LLM. To bridge this gap, a Projector module transforms the knowledge features X_{kn} into language embedding tokens H_{kn}.

The outputs from both pathways, the language embedding tokens H_q from the Embedding Layer and the projected knowledge embedding tokens H_{kn} from the

Projector, are then concatenated. This concatenation creates a single input vector that encapsulates both the linguistic content of the user query and the relevant contextual knowledge from the knowledge graph. This enriched input vector is then passed to the LLM component $f_\phi(.)$. The LLM, leveraging its internal parameters ϕ, processes this input and generates the next token X_a in the response sequence. This architecture's strength lies in its ability to infuse the LLM's language processing with contextual knowledge derived from a knowledge graph. This fusion allows the LLM to access information beyond the immediate query, potentially leading to more accurate, relevant, and informative responses.

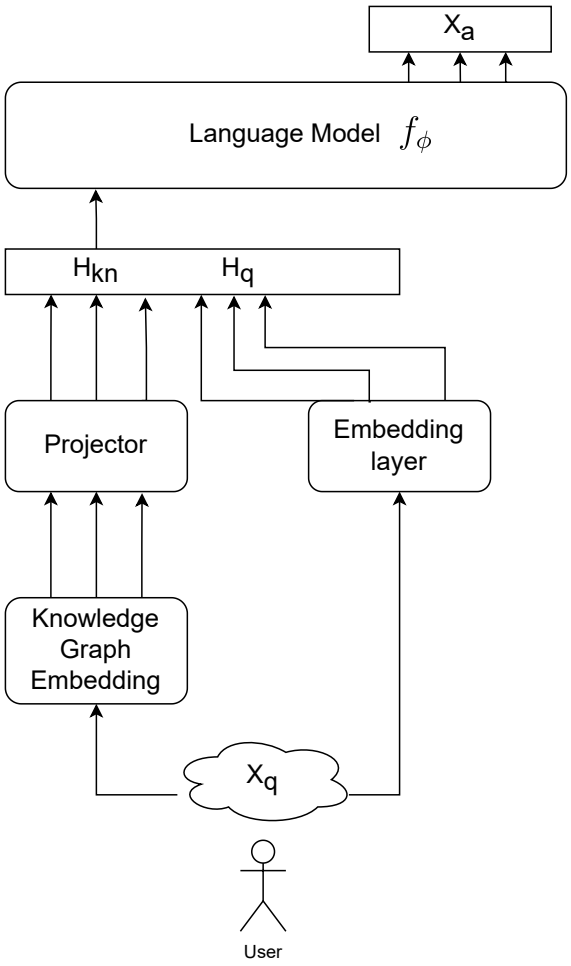

Figure 4.3: Overall of KEALLM architecture

4.2.2 Training

4.2.2.1 Pretrained language model knowledge graph embedding module

This project utilizes a pretrained language model (PLM) based approach for knowledge graph embedding, drawing inspiration from the work presented in KG-BERT[28]. This method leverages the power of PLMs to extract and embed knowledge from text, offering a robust and efficient solution for knowledge graph embedding. Each unique entity and relation is treated as a special token within the language model. This approach transforms link prediction into a masked entity prediction problem, building upon the work on k-NN KGE[44].

Masked Entity Modeling: A BERT-based model is employed as the entity predictor, converting link prediction into a masked entity modeling task. This approach utilizes structural information and textual descriptions to predict the missing entity in a triple. Given an incomplete triple $(e_i, r_j, ?)$, an input sequence x_k is constructed by concatenating these elements:

$$x_k = \texttt{[CLS]}\, e_i\, \texttt{[SEP]} r_j\, \texttt{[SEP]}\, \texttt{[MASK]}\, \texttt{[SEP]}$$

Through masked entity modeling, the correct entity e_k is identified by ranking the probability of each entity in the knowledge graph based on:

$$p(e_k|x_k) = p(\texttt{[MASK]} = e_k|x_k;\Phi) \tag{4.1}$$

where Φ represents the model's parameters
The model is trained by minimizing the following loss function:

$$\mathcal{L}_{kn} = -\mathbb{E}_{(x_k,e_k)\sim\mathcal{D}_{kg}} \log p(\texttt{[MASK]} = e_k|x_k;\Phi) \tag{4.2}$$

where D_{kg} is the training set and Φ represents the parameters of the model. The

Algorithm 1: Knowledge graph embedding training

Input: Knowledge graph $G = (E, R, T)$, where E is the set of entities, R is the set of relations, and T is the set of triples.
Input: Pre-trained language model Φ, e.g., BERT.
Input: Learning rate α, batch size B, number of epochs N.
Output: Trained model parameters Φ.

Initialize Φ with pre-trained weights.
Expand the vocabulary of Φ with unique entities in E.
for $epoch = 1$ to N **do**
  for each batch of triples $(e_i, r_j, e_k) \in T$ of size B **do**
    Construct input sequence $x_k = \texttt{[CLS]}\, e_i\, \texttt{[SEP]} r_j\, \texttt{[SEP]}\, \texttt{[MASK]}\, \texttt{[SEP]}$.
    Obtain probability distribution $p(e|x_k) = p(\texttt{[MASK]} = e|x_k; \Phi)$ for all entities $e \in E$
    Calculate the loss $\mathcal{L}_{kn} = -\mathbb{E}_{(x_k, e_k) \sim \mathcal{D}_{kg}} \log p(\texttt{[MASK]} = e_k|x_k; \Phi)$
    Update model parameters Φ using a gradient descent optimizer with learning rate α and the loss $\mathcal{L}_{kn}$: $\Phi \leftarrow \Phi - \alpha \nabla_\Phi \mathcal{L}$
  end for
end for
return Φ

process of training knowledge graph embedding is summarized in Algorithm 1. After training the knowledge graph embedding model, the feature vectors from the final hidden layer are used as the knowledge features.

4.2.2.2 Supervised Fine-Tuning

The supervised fine-tuning phase focuses on bridging the gap between the structured knowledge encoded in the knowledge graph embeddings and the linguistic understanding of the Large Language Model (LLM). This alignment is accomplished through a dedicated Projector module, which learns to translate knowledge embeddings into a format readily interpretable by the LLM. The Projector module acts as a bridge between the two worlds: knowledge graphs and natural language. It takes the knowledge embedding X_{kn}, representing the relevant entities and relationships extracted from the knowledge graph, and transforms it into a format compatible with the LLM. This transformation is achieved through a trainable projection matrix W, represented as the parameter θ in our model.

Specifically, the Projector module aims to maximize the likelihood of the LLM generating the correct target answer sequence X_a, given both the knowledge embeddings X_{kn} and the linguistic features of the user query X_q. This likelihood is formally defined in Equation 4.3:

$$p(X_a|X_{kn},X_q) = \prod_{i=1}^{L} p(x_i|X_{kn},X_q,X_{a<i};\theta) \tag{4.3}$$

where L represents the length of the answer sequence and $p(X_i|X_{kn},X_q,X_{a<i};\theta)$ denotes the probability of generating the i-th token in the answer, conditioned on the knowledge embeddings, user query and model parameters θ.

To optimize the Projector module, we utilize a supervised learning approach with a dataset (x,y), where x represents a user query aimed at eliciting a missing entity in a knowledge graph triple $(e_i,r_i,?)$, and y represents the correct target entity e_k. The training process employs a cross-entropy loss function, which measures the discrepancy between the LLM's predicted probability distribution over its vocabulary and the true target entity:

$$\mathcal{L}_{sft} = -\mathbb{E}_{(X_{kn},X_q,X_a)\sim\mathcal{D}_{sft}} \sum_{i=1}^{L} \log p(x_i|X_{kn},X_q,X_{a<i};\theta) \tag{4.4}$$

By minimizing this loss function, we effectively fine-tune the Projector module, enabling it to learn an optimal projection matrix W. This optimization ensures that the transformed knowledge embeddings align seamlessly with the LLM's internal language model, ultimately enhancing the LLM's capacity to access and utilize external knowledge for improved response generation.
Algorithm 2 demonstrates the process of fine-tuning the projector module.

4.2.2.3 Direct Preference Optimization

Direct Preference Optimization (DPO) further fine-tunes KEALLM by aligning its output with user preferences, achieved by training the model to prefer

Algorithm 2: Supervised fine-tuning of the Projector module

Input: Dataset $\mathcal{D}$ of (user query X_q, knowledge embedding X_{kn}, target entity X_a) tuples.
Input: Pre-trained LLM $f_\phi(.)$.
Input: Initialized Projector module with parameters θ.
Input: Learning rate α, batch size B, number of epochs N.
Output: Fine-tuned Projector module parameters θ.

for $epoch = 1$ to N **do**
  for each batch of tuples $(X_q, X_{kn}, X_a) \in \mathcal{D}$ of size B **do**
    Compute the concatenated input vector: $[H_q, H_{kn}]$
    Feed the concatenated vector to the LLM: $f_\phi([H_q, H_{kn}])$
    Obtain the LLM's predicted probability distribution over its vocabulary.
    Calculate the loss $\mathcal{L}_{sft} = -\mathbb{E}_{(X_{kn}, X_q, X_a)\sim\mathcal{D}_{sft}} \sum_{i=1}^{L} \log p(x_i | X_{kn}, X_q, X_{a<i}; \theta)$.
    Update Projector module parameters θ using a gradient descent optimizer with learning rate α and the loss $\mathcal{L}_{sft}$: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{sft}$.
  end for
end for
return θ

desired responses over undesired ones. We utilize a dataset D constructed from user queries and their corresponding preferred and less preferred responses derived from a combination of ground truth knowledge and the knowledge graph embedding. For each user query x, we obtain the ground truth entity as the preferred answer y_w, and an entity with the lowest probability from the knowledge graph embedding module as less preferred answers y_l.

The DPO loss function maximizes the difference in log probabilities between these preferred and less preferred responses:

$$\mathcal{L}_{DPO} = -\mathbb{E}_{(x, y_w, y_l)\sim D} \log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \tag{4.5}$$

where $\pi_\theta(y|x)$ is the probability of KEALLM generating response y given query x and model parameters of Projector module θ, $\pi_{\text{ref}}(y|x)$ is the probability of a reference model generating response y given query x, β scales the emphasis on the probability difference, $\sigma(.)$ is the sigmoid function.

Minimizing this loss encourages the model to assign higher probability to preferred responses compared to less preferred ones, relative to the reference model. This encourages KEALLM to generate responses aligned with the knowledge graph embedding's ranking of relevant entities. The process is summarized in Algorithm 3

Algorithm 3: Direct Preference Optimization for KEALLM

Input: Dataset D of (user query x, preferred responses y_w, less preferred responses y_l) tuples.
Input: Pre-trained KEALLM with parameters θ.
Input: Reference model π_{ref}.
Input: Learning rate α, batch size B, number of epochs N.
Output: Fine-tuned KEALLM parameters θ.

for $epoch = 1$ to N **do**
  for each batch of tuples $(x, y_w, y_l) \in D$ of size B **do**
    Compute KEALLM probabilities: $\pi_\theta(y_w|x)$, $\pi_\theta(y_l|x)$
    Compute reference probabilities: $\pi_{\text{ref}}(y_w|x)$, $\pi_{\text{ref}}(y_l|x)$
    Calculate the DPO loss: $\mathcal{L}_{DPO}$
    Update KEALLM parameters θ using a gradient descent optimizer with learning rate α and the loss $\mathcal{L}_{DPO}$: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{DPO}$
  end for
end for
return θ

4.3 Experiment setup

4.3.1 Baselines setup

Generator: Llama-2 7b [14] is chosen as the generator for this project because it is an open-source Large Language Model that can perform natural language generation and in-context learning. The configuration is described below:

- temperature: 0.5
- top_K: 5
- max_new_tokens: 100

Encoders: In this project, two types of encoders are used: Bi-encoder and Cross-encoder. Both of them use the MiniLM architecture, which is a distilled version of large pre-trained Transformer models [45]. The Bi-encoder, using the training weight utilizing the cosine similarity[1][46], maps sentences or paragraphs to a 384 dimensional dense vector space. On the other hand, the Cross-encoder uses the training weight utilizing the reranking task[2].
Retrieval-Augmented Generation experiment:

- Top-K documents: 20
- Top-K' documents: 5

Knowledge Graph-based Retrieval-Augmented Generation experiment:

- Top-K neighbor hops: 2
- Top-K' documents: 5

4.3.2 Data preparation

4.3.2.1 Knowledge graph

FB15k-237 [47]: FB15k-237 is a subset of the Freebase knowledge graph, specifically designed for knowledge graph completion tasks. It contains a curated set of entities and relations, addressing the issue of inverse relations present in the original FB15k dataset.
MetaQA[48]: MetaQA is a large-scale question answering dataset based on a knowledge graph extracted from Wikipedia. The knowledge graph contains factual triples, and the dataset includes a large set of multi-hop questions designed to test reasoning capabilities.
We preprocess both knowledge graphs to extract a set of triples for training the knowledge graph embedding module. The statistics of the preprocessed datasets are summarized in Table 4.1. Each knowledge graph dataset into train, devel-

[1]https://huggingface.co/sentence-transformers/multi-qa-MiniLM-L6-cos-v1
[2]https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-12-v2

Table 4.1: Knowledge graph description

Dataset	Entities	Relations	Triples
FB15k-237	14,505	237	310,079
MetaQA	43,238	9	138,989

opment, and test sets following the standard splits provided in their respective sources, as detailed in Table 4.2.

Table 4.2: Knowledge graph dataset splits

Dataset	Train Triples	Dev Triples	Test Triples
FB15k-237	272,208	17,535	20,336
MetaQA	121,213	8,821	8,955

4.3.2.2 Training dataset for Projector module

The Projector module plays a crucial role in KEALLM, bridging the knowledge encoded in the KG embeddings and the linguistic understanding of the LLM. To train this module, we construct a dedicated dataset derived from the knowledge graphs and user queries designed to elicit specific entities.

For MetaQA, we utilize the existing question provided in the dataset

For FB15k-237, we create a training dataset FB15k-237QA as follows:

- Triple Selection: Due to the limitation of computational resource, we randomly sample a subset of triples from the knowledge graph's training split (as outlined in Table 4.2) about 15,000 samples.
- Query Generation: For each triple (h, r, t), we generate a user query aimed at eliciting the tail entity t given the head entity h and the relation r. This query generation process is tailored to the specific characteristics of each knowledge graph. For simplicity, we use a template "What is the [relation] of [head entity]?"

Data Instance Creation: Each data instance consists of:

- User Query (X_q): The generated user query.

- Knowledge Embedding input $(e_i, r_i, ?)$: The tuple the head entity e_i and the relation r_i.

- Target Entity (X_a): The tail entity e_k of the triple.

This procedure results in a training dataset where each instance represents a user query seeking specific information that can be found in the knowledge graph. The Projector module is then trained on this dataset to learn a mapping between the knowledge embedding and a representation suitable for the LLM.

Table 4.3: Projector module dataset splits

Dataset	Train Instances	Dev Instances	Test Instances
FB15k-237QA	10,500	1,500	3,000
MetaQA (1-hop QA)	96,106	9,992	9,947

4.3.3 Implementation details

The training code is implemented in PyTorch[49]. We use Adam optimizer[50] for optimizing with a standard exponential decay learning rate scheduler. For the LLM component of KEALLM, we utilize the Llama-2-7b model [15]. This model, with 7 billion parameters, offers a balance between language modeling capabilities and computational feasibility. Due to limitations in computational resources, we employ 8-bit quantization to reduce the model's memory footprint and enable efficient training and inference. We use the pretrained BERT-base-uncased model for learning knowledge graph embeddings for both FB15k-237 and MetaQA. We fine-tune separate KG-BERT models for each knowledge graph using the training splits described in Table 4.2. The models are trained for 5 epochs with a batch size of 16 and a learning rate of 2×10^{-5}. The Projector module is trained for 10 epochs on each of the knowledge graph datasets described in Table 4.3. After training the Projector module, we further fine-tune again using DPO. We use the preferred and less preferred responses derived from the knowledge graph embedding, as described in the DPO section. The DPO training is performed for 5 epochs with a batch size of 8 and a learning

rate of 5×10^{-6}. The hyperparameter β, which controls the strength of the KL penalty in the DPO loss function, is set to 0.1. We also investigate applying DPO directly to the KEALLM architecture before the supervised fine-tuning (SFT) of the Projector module. This explores whether DPO can effectively guide the knowledge integration process even without initial supervised alignment.

4.4 Results and analysis

We evaluate KEALLM on two question answering benchmarks, FB15k-237 QA and MetaQA, comparing its performance against baselines, including the standard LLaMa-2 model, a retrieval-augmented generation (RAG) system, and a knowledge graph-augmented RAG (KG-RAG) system. The results are summarized in Table 4.4, with accuracy as the evaluation metric.

Table 4.4: Results on KEALLM, RAG and KG-RAG

Model name	**FB15K-237QA**	**MetaQA**
Llama-2	0.5242	0.3225
Llama-2 RAG	0.6159	0.5181
Llama-2 KG-RAG	0.6534	0.512
KEALLM-SFT	**0.6934**	0.557
KEALLM-DPO	0.2341	0.485
KEALLM-SFT-DPO	0.6824	**0.5883**

Some of findings:

- Baselines: The standard LLaMa-2 model achieves moderate accuracy on both datasets, indicating its limitations in accessing and utilizing factual knowledge. Both RAG systems, which retrieve relevant information from an external corpus, show significant improvements over the vanilla LLaMa-2. KG-RAG, which specifically retrieves information from a knowledge graph, achieves further gains, highlighting the benefit of structured knowledge for question answering

- KEALLM-SFT: KEALLM, after supervised fine-tuning (SFT) of the Projector module, outperforms all baselines on both datasets. This demon-

strates the effectiveness of directly integrating knowledge graph embeddings into the LLM, allowing for implicit knowledge access during response generation

- KEALLM-DPO: Directly fine-tuning KEALLM with DPO without the initial SFT stage leads to significantly lower accuracy. This suggests that the initial alignment of the Projector module through supervised learning is crucial for effective knowledge integration

- KEALLM-SFT-DPO: Combining SFT and DPO fine-tuning results in the best performance on MetaQA, indicating that DPO can further enhance the model's ability to align with user preferences and generate more accurate answers. However, on FB15k-237 QA, the performance slightly decreases compared to KEALLM-SFT, potentially due to overfitting or a mismatch between the DPO objective and the evaluation metric

Overall, the results demonstrate that KEALLM, by integrating knowledge graph embeddings directly into the LLM, provides a promising approach for knowledge-grounded response generation. The combination of SFT and DPO fine-tuning further enhances the model's performance, highlighting the benefit of aligning the model's output with user preferences. Further investigation is needed to understand the performance differences across datasets and fine-tuning stages

Chapter 5

Conclusion

5.1 Summary

This project explored the integration of knowledge graphs with Large Language Models (LLMs) to enhance their factual accuracy and knowledge retrieval capabilities. We analyzed existing retrieval-augmented generation (RAG) techniques, including Text-Based RAG and Knowledge Graph-Based RAG (KG-RAG). To overcome limitations of these methods, we proposed KEALLM - Knowledge Graph Embedding Augmented Large Language Model. Unlike other RAG approaches, KEALLM directly incorporates knowledge graph embeddings into the LLM architecture, enabling more efficient and integrated knowledge utilization. Experiments on one-hop question answering tasks demonstrated that KEALLM consistently outperforms standard LLMs and Text-Based RAG, achieving comparable performance to KG-RAG while offering a more streamlined architecture.

5.2 Future Work

Future research will focus on expanding and refining the KEALLM architecture to further improve its performance and applicability:

- Adapting to More Complex Question Answering: Extend KEALLM to handle multi-hop question answering, which requires reasoning over multiple relationships in the knowledge graph. This could involve incorporating graph neural networks or other reasoning mechanisms
- Dynamic Knowledge Integration: Explore ways to dynamically update the knowledge graph embedded in KEALLM. This would allow the model to adapt to new information and reflect changes in the real world, improving the model's accuracy and relevance
- Evaluating KEALLM on Diverse NLP Tasks: Evaluate KEALLM's performance on a wider range of NLP tasks beyond question answering, such as summarization, text generation, and dialogue systems. This will help to assess the model's generalizability and its potential in various domains
- Exploring Different Knowledge Graph Embedding Techniques: Experiment with alternative knowledge graph embedding techniques, such as those based on graph neural networks or complex relation representation models. This could lead to richer and more nuanced knowledge representations within KEALLM

References

[1] Manpreet Singh Lehal, "Comparison of cosine, euclidean distance and jaccard distance", *International Journal of Scientific Research in Science, Engineering and Technology*, vol. 3, 2017.

[2] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need", in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[3] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks", in *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014.

[4] Y. Wu, M. Schuster, Z. Chen, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation", *CoRR*, vol. abs/1609.08144, 2016.

[5] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation", in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, L. Màrquez, C. Callison-Burch, and J. Su, Eds., Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1412–1421.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[7] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization”, 2016.

[8] J. Yang, H. Jin, R. Tang, *et al.*, “Harnessing the power of llms in practice: A survey on chatgpt and beyond”, 2023.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding”, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[10] Y. Liu, M. Ott, N. Goyal, *et al.*, “Roberta: A robustly optimized bert pretraining approach”, 2019.

[11] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “Albert: A lite bert for self-supervised learning of language representations”, 2019.

[12] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, “ELECTRA: Pre-training text encoders as discriminators rather than generators”, in *International Conference on Learning Representations*, 2020.

[13] H. Liu, D. Tam, M. Mohammed, *et al.*, “Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning”, in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022.

[14] H. Touvron, L. Martin, K. Stone, *et al.*, “Llama 2: Open foundation and fine-tuned chat models”, 2023.

[15] H. Touvron, T. Lavril, G. Izacard, *et al.*, “Llama: Open and efficient foundation language models”, 2023.

[16] L. Ouyang, J. Wu, X. Jiang, *et al.*, "Training language models to follow instructions with human feedback", in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022.

[17] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 494–514, 2022.

[18] D. Vrandečić and M. Krötzsch, "Wikidata: A free collaborative knowledgebase", *Commun. ACM*, vol. 57, no. 10, pp. 78–85, Sep. 2014.

[19] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge", in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, 2017.

[20] O. Bodenreider, "The unified medical language system (umls): Integrating biomedical terminology", *Nucleic acids research*, vol. 32 Database issue, pp. D267–70, 2004.

[21] S. Ferrada, B. Bustos, and A. Hogan, "Imgpedia: A linked dataset with content-based analysis of wikimedia images", in *The Semantic Web – ISWC 2017: 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part II*, Vienna, Austria: Springer-Verlag, 2017, pp. 84–93, ISBN: 978-3-319-68203-7.

[22] Y. Liu, H. Li, A. Garcia-Duran, M. Niepert, D. Onoro-Rubio, and D. S. Rosenblum, "Mmkg: Multi-modal knowledge graphs", in *The Semantic Web: 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2–6, 2019, Proceedings 16*, Springer, 2019, pp. 459–474.

[23] M. Wang, H. Wang, G. Qi, and Q. Zheng, "Richpedia: A large-scale, comprehensive multi-modal knowledge graph", *Big Data Research*, vol. 22, p. 100 159, 2020.

[24] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data", in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26, Curran Associates, Inc., 2013.

[25] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, Jun. 2014.

[26] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion", *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, Feb. 2015.

[27] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix", in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong and M. Strube, Eds., Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 687–696.

[28] L. Yao, C. Mao, and Y. Luo, "Kg-bert: Bert for knowledge graph completion", *arXiv preprint arXiv:1909.03193*, 2019.

[29] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model", *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[30] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences", in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[31] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners", in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901.

[32] S. Gururangan, A. Marasović, S. Swayamdipta, *et al.*, "Don't stop pretraining: Adapt language models to domains and tasks", in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 8342–8360.

[33] N. Houlsby, A. Giurgiu, S. Jastrzebski, *et al.*, "Parameter-efficient transfer learning for nlp.", in *ICML*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, 2019, pp. 2790–2799.

[34] E. Ben Zaken, Y. Goldberg, and S. Ravfogel, "BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models", in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 1–9.

[35] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation", in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds., Online: Association for Computational Linguistics, Aug. 2021, pp. 4582–4597.

[36] E. J. Hu, Y. Shen, P. Wallis, *et al.*, "LoRA: Low-rank adaptation of large language models", in *International Conference on Learning Representations*, 2022.

[37] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners", 2019.

[38] "Tf–idf", in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 986–987, ISBN: 978-0-387-30164-8.

[39] G. Amati, "Bm25", in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 257–260, ISBN: 978-0-387-39940-9.

[40] S. Humeau, K. Shuster, M.-A. Lachaux, and J. Weston, "Poly-encoders: Architectures and pre-training strategies for fast and accurate multi-sentence scoring", in *International Conference on Learning Representations*, 2020.

[41] P. Lewis, E. Perez, A. Piktus, *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks", *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[42] Y. Feng, X. Chen, B. Y. Lin, P. Wang, J. Yan, and X. Ren, "Scalable multi-hop relational reasoning for knowledge-aware question answering", in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds., Online: Association for Computational Linguistics, Nov. 2020, pp. 1295–1309.

[43] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec, "QA-GNN: Reasoning with language models and knowledge graphs for question answering", in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, *et al.*, Eds., Online: Association for Computational Linguistics, Jun. 2021, pp. 535–546.

[44] P. Wang, X. Xie, X. Wang, and N. Zhang, "Reasoning through memorization: Nearest neighbor knowledge graph embeddings", in *Natural Language Processing and Chinese Computing*, F. Liu, N. Duan, Q. Xu, and Y. Hong, Eds., Cham: Springer Nature Switzerland, 2023, pp. 111–122, ISBN: 978-3-031-44693-1.

[45] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers", *Advances in Neural Information Processing Systems*, vol. 33, pp. 5776–5788, 2020.

[46] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks", in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019.

[47] K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases", in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, L. Màrquez, C. Callison-Burch, and J. Su, Eds., Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1499–1509.

[48] Y. Zhang, H. Dai, Z. Kozareva, A. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph", in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[49] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library", *Advances in neural information processing systems*, vol. 32, 2019.

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", in *3rd International Conference on Learning Representations, ICLR 2015,*

San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, Y. Bengio and Y. LeCun, Eds., 2015.

Appendix A

Prompt setting

A.1 NER prompt

Prompt: *As a good AI assistant, your mission is to extract entities in a sentence, your format answer is a dictionary json which key is entity, and value is type of entity.*
{fewshot}
Sentence: {sentence}
Answer:

A.2 Text-based RAG prompt

Prompt: *As a good AI assistant, your mission is to based on the context to answer the question as an entity, your format answer is a dictionary like “‘json { "answer" :‘Your answer‘} “‘*
{fewshot}
Context: ”’
{context}

”’

Question: {question}

Answer:

A.3 KG-based RAG prompt

Prompt: *As a good AI assistant, your mission is to based on the given relations to answer the question as an entity, your format answer is a dictionary like “‘json { "answer" :‘Your answer‘ } “‘*

{fewshot}

Relations: ”’

{context}

”’

Question: {question}

Answer:

Appendix B

2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics

A paper that we submitted to the 2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics that demonstrate our work was accepted . The url of the paper: https://arxiv.org/abs/2403.02715