Graph-based Retrieval Augmented Generation: A Case Study on the New Zealand Transport Agency

Felipe Mateo Navarro Cordero
School of Computer Science
The University of Auckland

Supervisor: Qian Liu

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of

Professional Studies in Data Science , The University of Auckland, 2024.

Abstract

This dissertation explores the development of a Retrieval-Augmented Generation (RAG) system based on two transport-related knowledge graphs utilizing large language models (LLMs). LLMs, neural network-based models trained on vast text corpora to generate text, have gained significant attention due to their capacity to generate and understand human language. While effective for general knowledge, they may produce false information (hallucinations) and struggle with domain-specific data without contextual grounding. This brings concerns about their use in higher-stakes enterprise scenarios.

The RAG addresses this problem by combining retrieval-based and generation-based models to contextualize the LLM with the domain data. We build a rag to allow users to query domain-specific information from the New Zealand Transport Agency (NZTA); we construct two knowledge graphs, one using Official Information Act (OIA) documents from NZTA and one based on LLM-generated multi-hop passages through self-prompted Chain-of-Thought (SP-CoT). We design a new sub-graph retrieval algorithm, Cosine-Similarity Multi-hop Retrieval (CSMR), to fetch query-related transport information from the knowledge graphs. Our algorithm demonstrates significant improvements in sub-graph retrieval compared to the original baseline method, as evidenced by metrics such as MRR, nDCG, and Precision. While enhancements in question-answering performance, evaluated through BLEU, ROUGE, and cosine similarity, are also observed, these gains are less pronounced than the improvements in retrieval. Although the system accuracy has increased, the overall performance remains limited, indicating that further refinement is necessary before it can be reliably deployed for enterprise applications.

In addition, we built a prototype chatbot application implementing the RAG engine and released it for public and research community exploration. It consists of a chat interface connected to the RAG with a chatbot capable of having regular conversations with the user and querying information from the knowledge graph to answer transport-related queries to the user.

Acknowledgments

I want to express my gratitude to my academic supervisor, Dr. Qian Liu, whose guidance, knowledge, and support were invaluable throughout this project. Her direction and encouragement during challenging moments were key for its completion. I am also grateful to my industry supervisor, Vikash Kumar, for allowing me to work on this project and for his consistent support. Finally, I would like to thank my friends and family.

Contents

Chapter 1

Introduction

In recent years, generative artificial intelligence (GenAI) powered by LLMs has proven helpful in many practical applications. Its primary adoption has been chatbot assistants, which can have human-like conversations with users. These models, such as the GPT-4 [1], LLaMa [2], Gemini [3], Claude [4], Mistral [5], are very knowledgeable in many domains; they can help with brainstorming, writing, coding, offering business client support, and more.

Despite these advancements, there are still challenges faced with using these models when it comes to generating domain-specific knowledge in a reliable way [6] . For example, enterprise applications often require specialized knowledge in law, finance, or transportation, which may not be well represented in the open data sources used to pre-train most LLMs [7]. This can lead to models producing responses that lack accuracy or relevance to specific business contexts, sometimes resulting in false information or "hallucinations" [8] —outputs where the model generates content that appears plausible but is factually incorrect.

Research into this problem has led to the development of retrieval-augmented generation RAG [7]. RAG addresses the issue by combining retrieval-based models with access to domain-specific data and generation-based models to produce a contextually relevant response. This significantly reduces the hallucination problem and allows the insertion of domain-specific data into the system with relative ease and without involving model fine-tuning [9].

Knowledge graphs effectively represent knowledge in a structured and interconnected format [10]. They allow querying information by exploring entities and their relations, so they can effectively enrich the context of an RAG [11]. In this work, we explore the integration of knowledge graphs with RAG to build a system that is able to answer questions specific to the domain of transport in New Zealand.

1.1 Research Background

Waka Kotahi – the New Zealand Transport Agency's (NZTA) mission is to develop and maintain an efficient, integrated, sustainable land transport system. It connects people and places throughout New Zealand, helping businesses grow and allowing Kiwis an engaged and social life. It collaborates with other organizations to extend its reach and the benefits it can provide to the community to its maximum capacity[12].

One of the responsibilities of NZTA is to be accountable and transparent in all its affairs according to the Official Information Act (December 17th, 1982) [13]. Working under the act, upon request of information from the public, NZTA delivers its responses in the form of letters and annexes made publicly available on its site. Although fully available, this data is not necessarily easily browsed or retrieved. Enabling users to query current online transport-related documents is an important application, and RAG is a promising method to address this issue.

1.2 Research Motivation

This research aims to improve how effectively Large Language Models (LLMs) can access and apply domain-specific knowledge. While LLMs offer impressive capabilities, they often fall short when required to deliver domain-specific information. We are particularly interested in the transportation domain. We build a RAG with a knowledge graph as its data source to address this limitation. The aim is to develop a system for the New Zealand Transport Agency (NZTA) that can retrieve and generate contextually relevant, accurate information, tailored to its unique needs.

1.3 Problem Statement and Scope

This research aims to address the question: **Is a Retrieval-Augmented Generation (RAG) system based on a knowledge graph an effective method for answering user queries related to transport?**

To investigate this question, we develop two RAG systems and corresponding knowledge graphs using two distinct transport-datasets, and we evaluate their performance. Additionally, we build a prototype chatbot application for the New Zealand Transport Agency (NZTA) utilizing the RAG engine, with the primary focus on enhancing the information retrieval algorithm.

A RAG system has several components, all containing much room for modification and parameter optimization. A core component is the information retrieval algorithm. Without a proper retrieval algorithm, relevant information in the database might fail to be presented as context to the LLM when requested. The focus of our work is to improve the LammaIndex [14] native retrieval algorithm. Upon reviewing it, inefficiencies in data extraction were noted, the details of which will be shared in section 3.7 (Query and retrieval). We identified three suboptimal processes:

1. The retrieval of neighbor relations lacks sorting to promote relevant relations to be ranked higher than irrelevant ones.
2. The retrieved text chunk corresponding to a node is the last text chunk parsed in the graph-building process, which contained that node, not necessarily the most relevant text chunk.
3. The context built upon the retrieved information lacked multi-hop relation chains.

By addressing these issues, we will be able to measure an improvement in the performance of the retrieval system.

There are other components in the RAG with the potential of being systematically optimized, which would improve the system's performance; however, they fall out of the project's scope. Some of these are data prepossessing and ingestion, graph disambiguation, prompt optimization, and evaluation of different embedding and LLM models at the task. We followed sensible procedures and parameter settings for these elements but did not perform a systematic and involved optimization of them; we leave this as future work.

1.4 Contributions

To address the challenges identified in the existing retrieval methods of LlamaIndex, we propose a novel model **Cosine-Sorted Multi-Hop Retrieval (CSMR)**. This model aims to optimize retrieval by incorporating relation-query cosine similarity node sorting, selecting the most relevant text chunks, and including multi-hop relation chains for richer contextual responses. Our main contributions are summarized below:

- We improve neighbor node selection by sorting neighbor triplets through relation-query cosine similarity.
- We enhance text chunk selection by modifying the retrieval algorithm to select the most contextually relevant text chunk associated with the head node of the triplet through cosine similarity.
- We incorporate multi-hop relation chains into the context provided to the LLM.
- We evaluate the effectiveness of the proposed retrieval method against the original LlamaIndex by conducting tests on two knowledge graphs: one constructed from public NZTA OIA documents and another generated using the SP-CoT method [15].
- We demonstrate the practical impact of our model by showing improved retrieval precision and LLM answer quality in the context of domain-specific queries related to transport.

Chapter 2

Related Work

2.1 Semantic Representation

Two core machine learning and computer science developments that make RAG systems possible are pre-trained large language models (LLMs) and word/sentence embeddings. LLMs can generate coherent text in a human-like manner; they power chatbots like OpenAIs Chatgpt, Anthropic's Claud, Google's Gemini, Meta's Llama, and others. Embeddings are the technology that allows information to be extracted based on semantic similarity. Hence, by combining both technologies, one can build the RAG by using the user's query to extract information with embedding technology and giving it to the LLM as context to get answers based on direct information.

2.1.1 Word Embeddings

A word embedding is the vectorized representation of the word. In 2013, Mikolov et al. introduced two pivotal and foundational techniques to create word embeddings [16]. These methods are the continuous bag of words (CBOW) and the continuous Skip-Gram models (Skip-gram). Both of these are referred to as word2vec. They consist of a very simple yet effective neural network architecture composed of an input layer, a single hidden layer, and an output layer, Figure 2.1.

In the CBOW model, the word embedding is the ith row of the weight matrix, which projects the input of one hot encoded words into the hidden layer [16]. The skip-gram model is reciprocal; the input is the target word, and the prediction is the window of words surrounding the target word. Once training is done, the resulting word embedding is the product of the projection matrix and the one hot encoded representation of the target word. A scheme of both architectures is shown below in Figure 2.1

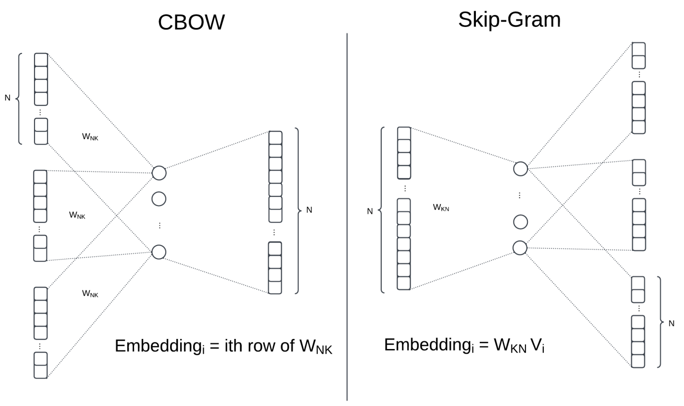
CBOW
Skip-Gram
N
W_{NK}
W_{NK}
W_{NK}
N
N
W_{KN}
N
Embedding$_i$ = ith row of W_{NK}
Embedding$_i$ = W_{KN} V_i

Figure 2.1: Illustration of the CBOW and Skip-Gram architectures in word2vec. The CBOW model predicts a target word from its surrounding context words. The embedding for a word in the CBOW model is represented as Embedding$_i$ $=$ ith row of W_{NK}, where W_{NK} is the weight matrix projecting the input to the hidden layer. On the other hand, the Skip-Gram model predicts the context words from a given target word. The embedding for a word in the Skip-Gram model is computed as Embedding$_i$ $= W_{KN}V_i$, where V_i is the one-hot vector representation of the target word.

Mikolov et al. [17] demonstrate that this method is very effective at representing the meaning of words. It allows measuring the similarity of meaning between two distinct words with metrics like cosine similarity, which is one of the primary utilities of word embeddings.

Progress in the area continued. Two of the most notable developments following Mikolov's work came with GloVe [18] and fastText [19] by Meta, formerly Facebook. GloVe introduced the concept of global vector representation. Unlike CBOW and Skip-Gram, which consider single local context windows a time, GloVe consists of the factorization of the co-occurrence matrix of all words across all text corpora. GloVe demonstrated to be better at finding similarities according to meaning than word2vec, and it demonstrated the importance of the inclusion of global context [18]. FastText made a significant contribution by utilizing sub-word embeddings rather than whole words. FastText splits words into smaller chunks, letter-n-grams, and embeds them rather than the entire word. The word embedding is then found by adding the embedding of its subword components. This way uncommon or even new words not found in the training set can also be adequately represented by adding the subword components [19].

2.1.2 Pre-trained Language Models

The next significant improvement came along with context-aware embeddings. For example, the embedding from Language Models ELMo [20]. Unlike the previous models, this architecture can handle polysemy, i.e., generating different embeddings for the same word when it has other meanings determined by the context, like a bank, for example, which can refer to a financial institution or the side of a river.

After introducing context-aware embeddings, the next major leap came with the invention of the transformer architecture by Vaswani et al. [21]. While ELMo used bidirectional LSTMs to achieve context sensitivity, transformers replaced this with a self-attention mechanism that is more efficient and scalable. This method consists of a process that identifies how closely related words around the target word are to weigh their contribution to the meaning of the target word proportionately. This method was implemented in newer models like BERT [22] and RoBERTa [23]. Transformers became the foundation for modern NLP models.

The development of embeddings continued with the development of models like Sentence-BERT (SBERT) [24] and the Universal Sentence Encoder (USE) [25], both developed by Google. Unlike previous embedders, SBERT and USE allow the vector representation of entire texts. This enables semantic similarity measurements between sentences or bigger text chunks, making it possible to index documents based on the semantic meaning of their content. This approach is central to how RAG systems perform document retrieval through semantic similarity.

These advancements in word and sentence embeddings laid the foundations for representing semantic relationships in text. However, these models cannot generate coherent responses like a human would. The solution to this problem would be in development in parallel with the creation of LLMs.

2.1.3 LLMs

The transformer's development paved the way for the large language models we use today. Radford et al. [26] created the first instance of this model in 2018 with the generative pre-trained transformer (GPT). As its name suggests, the nature of this model is to be generative. Unlike embedding models, whose purpose is to create vector representations of words or sentences, the task of a GPT is to get text as input and output the next most probable word (token) to follow. Suppose the initial output is appended to the input and re-fed into the model, and we repeat this process multiple times. By doing that, we get a system that generates sentence or text completion.

A simple but clever trick can turn a text completion model into a chat agent. To do this, an initial passage that sets the context as the conversation between an assistant and a user is pre-set. Then, when the user writes the first message, it is appended to the pre-set passage as the user's comment. An additional text is appended to this concatenation, an entry that implies that it is now the assistant's turn to respond. This way, the GPT will respond as a chatbot assistant when completing the text.

GPT matched or improved state-of-the-art models for tasks like question answering and common sense reasoning, semantic similarity identification, and text classification. GPT-2 followed. It is a slightly modified version to the original GPT but with increased parameters and training time [27]. It demonstrated the capability of LLMs to learn machine translation, question answering, reading comprehension, and summarization without being explicitly supervised. In addition, they show that just scaling this model results in much better performance. GPT-3 continued the development, with 175 billion parameters, more than 100 times the parameters of GPT2. It demonstrated improvements on all tasks over GPT-2, and it is the first GPT version that generated text where human evaluators have trouble distinguishing it from human-written text [28].

The creation of these models by OpenAI sparked the interest of other companies and independent researchers in developing their generative LLMs. Google developed its own models LaMDA (2021) [29], PaLM (2022) [30], as well as Meta with OPT (2022)[31] and LLaMA (2023) [32], Anthropic with Claude (2023) [4], Mistral with Mistral 7b (2023) [33]. The now well-known website Hugging Face turned into a hub for transformer model hosting and sharing, which resulted in hundreds of open-source models and variations being accessible to anyone [34]. Since then, models have kept getting larger and broadly improving their capabilities. These advancements allow for exploring specialized applications, such as the retrieval-augmented generation system developed in this dissertation.

2.2 Open-Source Frameworks for LLM-Based Agents and Indexing

Given the capabilities of LLMs previously discussed, such as text comprehension, writing [28], rational decision-making, instruction following [35], and programming [36], developers have begun creating tools to harness these advanced features effectively. To do this they have developed tools that allow LLMs to become agents capable of making decisions and being part of a program workflow. The two leading frameworks that allow this are Langchain [37]and llamaIndex [14]. Both of the frameworks have overlapping functionalities.

Langchain mainly provides tools for building apps with agentic workflows where LLMs make decisions through 'chains.' I.e., a structured sequence of tasks or operations linked together where the output of one task serves as the input of the next. For example, one might have a sequence of agents that do the following: agent1 receives a list of tweets about a company and tags them with the sentiment as positive or negative. These tweets are separated by sentiment and handed over to agent 2. Agent2 summarizes all of the positive and negative tweets independently. Finally, Agent 2 passes its output to Agent 3, the last agent, who summarizes both results and gives the user a broad overview of the negative and positive.

LlamaIndex also provides tools for chain agentic processes, but it stands out for its tools allowing knowledge retrieval from diverse databases, which we need to build a RAG. Llama index provides a library with a modular architecture that facilitates the connection of LLM agents to different databases and classes that are customizable in how data is ingested and retrieved. For this reason, we chose to work with LlamaIndex on this project.

2.3 Groq & Replicate

Much research is directed to building open-source LLMs that are as capable as enterprise models, such as OpenAI's Chatgpt Anthropic's Claud and Google Gemini. Considerable progress has been made; companies like Meta and Mistral have released models comparable to Enterprise in state-of-the-art benchmarks [38, 39]. However, a challenge is associated with offering the public the capability of running these models; Medium to big models require high computing power and resources to function with fast inference times way over the capabilities of regular personal desktop computers or laptops. Groq, a startup based in California, USA, developed a processor known as a Tensor Streaming Processor that allows parallel processing of large-scale AI models [40]. The company offers free inference endpoints with an extensive collection of open-source models. Their API was used in this project to run the LLM models That build the knowledge graph. The model selected was a version of Meta's llama3 model fine-tuned for tool use labeled as llama3-groq-70b-8192-tool-use-preview by Groq.

Like Groq, Replicate [41] has emerged in the LLM API market for the exact causes. Replicate offers inference endpoints, fine-tuning services, and supports hosting user-created models. Unlike Groq, Replicate charges for its services but offers higher limits for inference. We used services offered by Replicate sparingly when Groqs free service rates were insufficient to perform necessary tasks.

2.4 Hugging Face

Hugging Face [42] is an online platform that supports various machine learning services oriented towards Large Language models as well as supporting a community of open source contributors and LLM developers. They offer API for inference and provide open-source LLms and embedding models. For this project, we used the embedding model BAAI/bge-small-en-v1.5 obtained from Hugging Face.

Developed by the Beijing Academy of Artificial Intelligence (BAAI) [43], BAAI/bge-small-en-v1.5 is explicitly designed to generate dense embedding that captures semantic meaning. It is optimized for text similarity and retrieval tasks, providing efficient performance in cosine similarity-based applications. The model's lightweight architecture makes it suitable for large-scale embedding generation without compromising accuracy.

2.5 Graph Databases and Neo4j

Some data have an inherent structure that is not best represented by traditional databases. An example is the data related to a social network where people have friends and followers. A more suitable approach to represent this data rather than a tabular format is storing people as entities and their relations as edges in a graph representation; other examples where graph-structured databases are useful are in recommendation systems and fraud engines detection [44]. For use cases like this and many others, Graph databases were created. In this research, we use a graph database to represent people, projects, organizations, and other entity types belonging to the NZTA knowledge domain.

Neo4j is built on the property graph model, where data is stored as nodes (entities), relationships (edges), and properties (key-value pairs) that can be attached to both nodes and relationships. This flexibility makes Neo4j an efficient choice for large-scale graph traversal and querying tasks. The Neo4j platform provides ACID compliance and a rich query language, Cypher, simplifying querying complex graphs. More details about Neo4j can be found on their official website [45].

Neo4j was selected for this project because it offers performance and scalability that are sufficient for the project and is scalable for production use cases. In addition, it has native support by LlamaIndex, which allows seamless integration, making it a very suitable choice for our system. In addition, Neo4j is built around comprehensive and well-documented resources, which is helpful for fast prototype building and troubleshooting.

2.5.1 Graph Based RAGs

Graph Neural Networks (GNNs) have been developed to encode the structural information of knowledge graphs, analogous to how embedding provides a vector representation of words, phrases, or texts. RAG methods that utilize GNNs typically involve extracting a sub-graph from the knowledge graph relevant to the query and then applying the GNN to generate embeddings that encapsulate the graph's structural and semantic information. These embeddings are then passed to an internal layer of the LLM to generate a response.

Although effective [46–49], this method introduces additional complexity and computational overhead. The GNN must be trained, and its output must be inserted into the LLM's internal layers, complicating the architecture and hindering the modularity and adaptability of the RAG components.

2.5.2 Non-GNN Retrieval Methods

In contrast, non-GNN does not require the architecture related to training GNNs. Instead, the knowledge extracted from the graph database is passed as context as a text-formatted string to an LLM agent that processes the data.

In addition, the graph rag system can be classified in two more ways, **Iterative Exploration Guided by LLMs** and **Direct Subgraph Extraction Methods**.

2.5.2.1 Exploration Guided by LLMs

One potential method for information retrieval from a graph database is to employ an LLM to determine which nodes and relations to explore in order to answer a query. In one approach, the LLM generates a predefined sequence of reasoning steps, specifying the node types and relations to follow [50–53]. This strategy is particularly suitable for graphs with well-defined node and relation types, where the LLM outputs a structured exploration plan based on its understanding of the query.

Alternatively, the LLM can be given a query along with an initial seed node and its connected relations. In this iterative approach, the LLM assesses the available relations and dynamically decides which path to follow at each step, continuing the exploration until a termination condition is met, such as reaching a specified depth or identifying a satisfactory answer.

2.5.2.2 Direct Subgraph Extraction Methods

The alternative to LLM path exploration is a direct subgraph extraction from the knowledge graph [49, 54–56]. This requires a strategy to identify starting (seed) nodes in the graph and then a strategy for selecting the relations to explore around these initial seed nodes. Once the subgraph is extracted, it must be converted into an appropriate text format for LLM consumption.

Llamaindex native retrieval is based on this second direct sub-graph method, and our improvement over it is based on an additional embedding similarity sort over neighbor triplets and text selection and including multi-hop chains into the context. We discuss the details in the Methods section.

2.6 Evaluation of Retrieval-Augmented Systems

A proper evaluation of an RAG is needed to identify strengths and weaknesses in the system and compare various methods. Evaluation can take place at two levels: information retrieval and response generation. Several methods have been created and adapted for this purpose. We outline them below.

2.6.1 Mean Reciprocal Rank

One of the evaluation methods we use to compare performance in node ranking retrieval is Mean Reciprocal Rank or MRR [57]. This metric evaluates how well an information retrieval system ranks the first relevant item. Its definition is the following:

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$

Where Q is the set of all queries, $|Q|$ the number of queries in the set, and $\textbf{rank}_i$ is the rank of the first relevant item retrieved.

2.6.2 Normalized Discounted Cumulative Gain

A metric that takes into consideration the overall distribution of relevant items in the retrieved list is the Normalized Discounted Cumulative Gain (nDCG) [58]. To understand this metric, let us first define its components.

Discounted Cumulative Gain The discounted cumulative gain of a list is the sum of the relevances of the items divided by the logarithm of 1 plus the item's rank. For the work in this project, the relevance is binary (1 for relevant items, 0 for non-relevant items).

$$\text{DCG@}k = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)}$$

This metric results in higher values for lists with a greater number of relevant items and lists whose relevant items are higher in the ranks.

Ideal Cumulative Discounted Gain The Ideal Cumulative Discounted Gain (IDCG) is the DCG of the results of a query sorted in the true best order according to relevance. It follows the same definition as above:

$$\text{IDCG@}k = \sum_{i_{\text{ideal}}=1}^{k} \frac{\text{rel}_{i_{\text{ideal}}}}{\log_2(i_{\text{ideal}}+1)}$$

Normalized Discounted Cumulative Gain The Normalized Discounted Cumulative Gain is the ratio between the DCG and IDCG of a query result:

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}$$

2.6.3 Precision

We now have metrics to measure the first appearance of a relevant item and the ordering of relevant items in the result. However, we still need a metric to compare the proportion of relevant items retrieved. For this, we use precision, which is defined as:

$$\text{Precision@k} = \frac{\text{Number of Relevant Items Retrieved in Top k}}{k}$$

The next component of the RAG that we evaluate is the responses the LLM agent generates based on the retrieved context. Given that the responses to the questions proposed in this project are composed of multiple words or phrases that we can write differently, direct string comparison between reference and RAG system answers is inadequate. This is a well-known issue, and several metrics have been developed for this purpose. We go over them below.

2.6.4 BLEU

The Bilingual Evaluation Understudy evaluation metric (BLEU) was originally developed to evaluate machine translation [7]. However, given its parallels with evaluating QA responses, it has been widely adopted to evaluate question-answering systems like a RAG [7]. BLEU is an n-gram-based precision metric; that is, It evaluates how many n-grams in the response are correct out of all the n-grams in the response. It does not measure how complete the response is, i.e., how many correct n-grams are found in the response out of all the n-grams that are found in the actual answer. A phrase n-gram breaks up a sentence into consecutive groups of n words. For example, the n-grams for n 1-4 for the sentence ”The quick brown fox jumps over the lazy dog.” are:

1-gram (unigrams) Each word in isolation: The, quick, brown, . . .

2-gram (bigrams) Pairs of consecutive words: The quick, quick brown, brown fox, . . .

3-gram (trigrams) Sequences of three consecutive words: The quick brown, quick brown fox, brown fox jumps, . . .

The definition of the BLEU score is the following:

$$\text{BLEU} = BP \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

N is the maximum n-gram length (4 in this research). w_n is the weight for each n-gram order (in this research, all weights are equal, $w_n = \frac{1}{4}$). p_n is the precision for the n-grams (proportion of n-grams in the candidate text that appears in the reference). And BP is the Brevity Penalty, which penalizes responses that are too short and might otherwise end up with a high precision score. It is defined as follows:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases}$$

r is the length of the reference answer, and c is the length of the LLM answer (candidate answer).

To handle short answers, BLEU uses smoothing by adding 1 to the count of matched n-grams, preventing zero precision for higher-order n-grams. The adjusted precision calculation is:

$$p_n = \frac{\text{matches}_n + 1}{\text{candidate}_n + 1}$$

This method ensures fair evaluation for short responses by avoiding zero scores when few n-grams are present. This technique was used with the SP-CoT QA dataset which has responses as short answers.

2.6.5 ROUGE

Like BLEU is analogous to precision Recall-Oriented Understudy for Gisting Evaluation (ROUGE) is another n-gram-based metric analogous to recall [59]. ROUGE was originally developed for summary evaluations, but it has since been adopted widely for RAG system answer evaluations. There are two variations of it. The first, known as ROUGE-N, is defined as

$$\text{ROUGE-N} = \frac{\sum_{S \in \text{ReferenceSummaries}} \sum_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S \in \text{ReferenceSummaries}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)}$$

Where $\text{Count}_{\text{match}}(\text{gram}_n)$ is the highest count of n-grams that appear both in the candidate summary and across the set of reference summaries. For this research, there is only one reference set, which is the correct answer, so we can reduce the equation to

$$\text{ROUGE-N} = \frac{\sum_{\text{gram}_n \in \text{Reference}} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{\text{gram}_n \in \text{Reference}} \text{Count}(\text{gram}_n)}$$

The second version of ROUGE is known as ROUGE-L. It is defined as follows:

$$\text{ROUGE-L} = \frac{LCS(\text{Reference}, \text{Candidate})}{\text{Length of the Reference}}$$

Where $LCS(\text{Reference}, \text{Candidate})$ is the longest common sub sequence between the reference and the candidate answer.

2.6.6 Cosine Similarity

Finally, the last method we use to evaluate the model response to the reference is the cosine similarity between the embedding response and the candidate response. The cosine similarity is a measure of how similar two vectors are. For vectors pointing in the same direction, the cosine similarity is equal to one; for vectors pointing in opposite directions, the cosine similarity is equal to negative one; orthogonal vectors have a cosine similarity of 0. Given that embeddings capture semantic meaning, the cosine similarity between embedding vectors measures how similar the two responses are. We calculate it as follows:

$$\text{Cosine Similarity} = \frac{\vec{R} \cdot \vec{C}}{||\vec{R}||||\vec{C}||}$$

where $\vec{R}$ is the embedding of the reference answer and $\vec{C}$ is the embedding of the candidate answer.

2.7 Statistical tests

We evaluate the significance of our improvements with statistical tests performed in the paired difference of the metrics calculated. Traditionally, a t-test has been used for this type of evaluation. T-test assumes normality in the distribution of the data. Upon testing our results with a Shapiro-Wilk Normality test [60], we found that the distributions were not normal. For this reason, instead, we relied on the sign rank test [61].

2.7.1 Shapiro-Wilk Normality Test

Shapiro and Wilk created a test to evaluate the normality of a distribution [60]. Its method consists of comparing the empirical ordered values of the sample:

$$[y_1, y_2, \ldots, y_n]$$

With the expected value of each if the data were normally distributed:

$$[m_1, m_2, m_3, \ldots, m_n]$$

If the distribution of y_i is normal, then it can be expressed as

$$y_i = \mu + \sigma x_i \quad (i = 1, 2, \ldots, n).$$

With V being the covariance of m The best linear fit for μ and σ can be found solving:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} y_i = \bar{y}, \quad \text{and} \quad \hat{\sigma} = \frac{m^T V^{-1} y}{m^T V^{-1} m}.$$

This leads to the following definition of the statistic W

$$W = \frac{R^4 \hat{\sigma}^2}{C^2 S^2} = \frac{b^2}{S^2},$$

where

$$R^2 = m^T V^{-1} m, \quad C^2 = m^T V^{-1} V^{-1} m, S^2 = \sum_{i=1}^{n} (y_i - \bar{y})^2,$$

For normally distributed y, b will approximate S, resulting in $W \approx 1$. For non normally distributed $y, b < S$ and so $W < 1$. The distribution of W is found empirically through simulations, and the p-value is derived out of this distribution [62, 63]. The Shapiro-Wilk test can be interpreted as measuring the straightness of the Q-Q plot of y.

2.7.2 Sign Test

To test for significance, we will use the sign test applied to the distribution of the difference between the results of the new vs. original system. The sign tests are suitable to test nonnormal distributions. It works the following way. $[x_1, x_2, \ldots, x_n]$ are Radom samples from a population. We want to test whether their median M is equal to some reference value M_0

$$H_0 : M = M_0 \quad \text{vs.} \quad H_a : M \neq M_0$$

We then note that under the null hypothesis, if we take the sign of the difference $[x_1 - M_0, x_2 - M_0, \ldots, x_n - M_0]$. The distribution of the positive signs follows a binomial distribution with $p = 0.5$

$$P(x_i > M_0) = 0.5$$

To test the difference in medians of the two distributions with confidence α. we find the rage r, s such that.

$$P(K \leq r) = \sum_{i=0}^{r} \binom{n}{i} (0.5)^n \leq \alpha/2$$

$$P(K \geq s) = \sum_{i=s}^{n} \binom{n}{i} (0.5)^n \leq \alpha/2$$

Then If the actual value of K falls either smaller than r or bigger than s we can be sure that the means are different with a confidence of $alpha$

[61]

2.7.3 McNemar Test

The McNemar test determines if there is a significant difference between correlated proportions in paired binary data [64, 65]. In other words, data that can be represented by the following contingency table.

	Condition 2: Success	**Condition 2: Failure**	**Row Totals**
Condition 1: Success	n_{11}	n_{12}	$n_{1+} = n_{11} + n_{12}$
Condition 1: Failure	n_{21}	n_{22}	$n_{2+} = n_{21} + n_{22}$
Column Totals	$n_{+1} = n_{11} + n_{21}$	$n_{+2} = n_{12} + n_{22}$	$n = n_{11} + n_{12} + n_{21} + n_{22}$

Table 2.1: Contingency table for paired binary data in McNemar's test

It is designed to test the following null hypothesis: that the proportion of condition one success and condition two failures is equal to condition one failure and condition two successes:

$$H_0 : p_{12} = p_{21}$$

The statistic used is

$$T = n_{12} + n_{21}$$

And so given the null hypothesis assumption the distribution of n_{12} is binomial T with probability 0.5

$$n_{12} \sim \text{Binomial}(T, 0.5)$$

We get the p-value by using the distribution and calculating the probability of $n_{12})$ being equal to or more extreme than its value. We will use this test on SP-CoT data when evaluating correct and incorrect answers for the original and improved method.

Chapter 3

Methodologies

The primary goal of this project is to assess the effectiveness of integrating a Knowledge Graph into a Retrieval-Augmented Generation (RAG) system to improve domain-specific question answering. To achieve this, we built two distinct knowledge graphs: one based on Official Information Act (OIA) documents from the New Zealand Transport Agency (NZTA) and another generated using a Self-prompted Chain-of-Thought (SP-CoT) LLM dataset. These knowledge graphs form the foundation of two RAG systems, which are then tested to evaluate their performance.

The evaluation focuses on how well each system retrieves relevant nodes from the knowledge graph and generates accurate responses. By comparing the original retrieval algorithm with the improved CSMR, we aim to determine the impact of our enhancements. This chapter outlines the detailed methodology, construction process, and evaluation criteria used in our experiments.

3.1 Key Terms Definitions

Before getting into detail, we will define some important terms in this project:

- **Text chunk**: When processing a piece of text to add its information to the knowledge graph, it is first broken down into smaller segments called text chunks. It is this text chunk from which we extract the nodes and relations and the text that we upload to the database as an independent text node.
- **Text Nodes**: Nodes in the Knowledge graph that correspond to the text chunks.
- **Entity Nodes**: These nodes in the knowledge graph correspond to entities extracted from the text chunk.
- **Relation**: Describes the association or logical connection between nodes within the knowledge graph.
- **Triplet**: Two nodes with the relation that connects them are called triplets.
- **N-hop Neighbor**: An n-hop neighbor of a node ν_1 is a node ν_2 who's shortest path from ν_1 is of length n.

- **Agent**: An agent in the context of LLMs is an independent system that implements an LLM to perform a specific task, such as extracting triplets from a text chunk.

3.2 General Overview

A RAG architecture in its most basic form consists of a large language model and a data store connected in a way such that when a user performs a query, not only does the LLM receive the user query as input but also information retrieved from the store based on the user's query, Ideally this information is relevant to the user's query and the LLM can use the context to correctly and reliably answer. We present a setup diagram below in Figure 3.1. The document stores are typically vector databases; however, we use a graph database in this project.

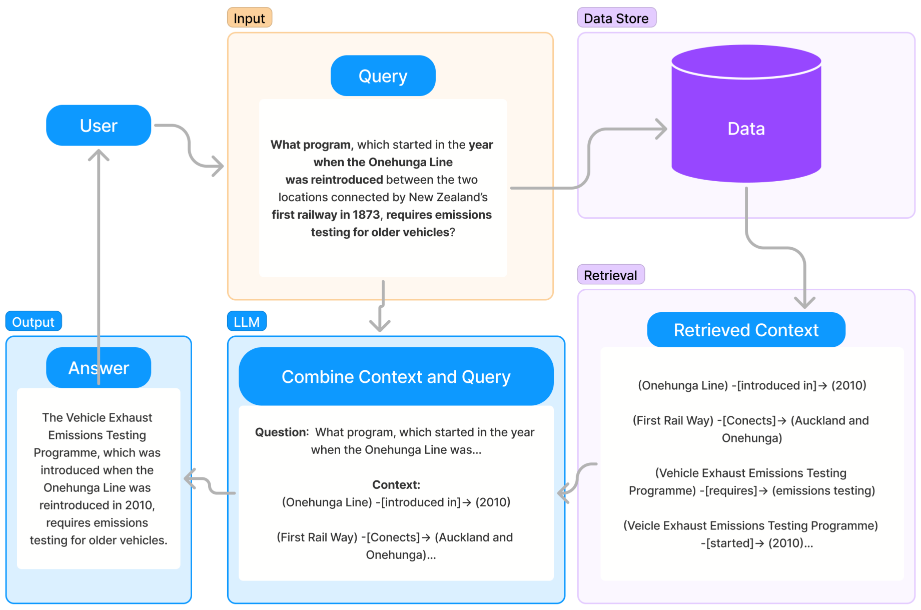
Input
Query
What program, which started in the year when the Onehunga Line was reintroduced between the two locations connected by New Zealand's first railway in 1873, requires emissions testing for older vehicles?
Data Store
Data
User
Retrieval
Retrieved Context
(Onehunga Line) -[introduced in]→ (2010)
(First Rail Way) -[Conects]→ (Auckland and Onehunga)
(Vehicle Exhaust Emissions Testing Programme) -[requires]→ (emissions testing)
(Veicle Exhaust Emissions Testing Programme) -[started]→ (2010)...
Output
Answer
The Vehicle Exhaust Emissions Testing Programme, which was introduced when the Onehunga Line was reintroduced in 2010, requires emissions testing for older vehicles.
LLM
Combine Context and Query
Question: What program, which started in the year when the Onehunga Line was...
Context:
(Onehunga Line) -[introduced in]→ (2010)
(First Rail Way) -[Conects]→ (Auckland and Onehunga)...

Figure 3.1: Overview of a basic RAG system workflow. 1. The user submits a query. 2. The query is processed to retrieve relevant context from the data store. 3. The retrieved context, along with the query, is combined and passed to the LLM for processing. 4. The LLM uses the combined information to generate an answer. 5. The answer is returned to the user.

A graph database is a type of database that stores information that has a relational structure. That means it stores nodes that represent entities that connect through relations. We refer to all the information in the database as the knowledge graph. In this investigation, we build two knowledge graphs. We construct one out of the content of 100 OIA NZTA files and a second out of an artificially generated dataset through Self Prompted Content Generation (SP-CoT) [15]. We populate the graphs with entities and relations the LLM finds in the texts. We use the knowledge graph to connect information about the same entity found in different documents, potentially improving the retrieval of relevant context from the database. In our implementation, not only the entities and relations extracted by the LLM are uploaded into the database, but also nodes representing the text chunks from where we extracted the entities. This way, we have the information found in the relation and the whole context from where we extracted it. We present a sketch of the structure of this knowledge graph below 3.2.

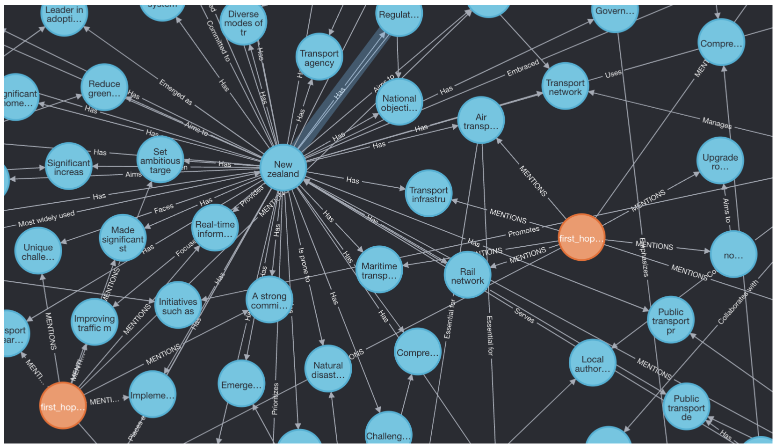

Figure 3.2: Our knowledge graph structure is composed of entity nodes (blue) and their relations as well as text chunk nodes (orange) that mention the entities.

3.3 Formal Procedure

We overview the general method before addressing the technical details of the procedures. The steps of what we want to accomplish are displayed below. The same is illustrated in Figure 3.3.

1. Given a set of documents or texts d_i, we break each document into chunks c_{ij}, where i indicates the document and j represents the chunk within that document.
2. We then extract meaningful entities and relations from each chunk c_{ij}

$$f(c_{ij}) \to E_{ij} \cup R_{ij}$$

where E_{ij} are the extracted entities and R_{ij} are the relations from chunk c_{ij}.

3. These entities, relations, and chunks are stored as a graph $\mathcal{D}$.

$$\mathcal{D} = \bigcup_{i,j} (E_{ij} \cup R_{ij} \cup \{c_{ij}\})$$

4. Given a query q with a true answer a, we create an extraction process $P(q, \mathcal{D})$ that retrieves a set of relations, a set of entity nodes, and a set of text nodes from $\mathcal{D}$ in other words a subgraph:

$$P(q, \mathcal{D}) \rightarrow \{R_{q\mathcal{D}}, E_{q\mathcal{D}}, T_{q\mathcal{D}}\}$$

   where $R_{q\mathcal{D}}$ denotes the retrieved relations, $E_{q\mathcal{D}}$ denotes the retrieved entity nodes, and $T_{q\mathcal{D}}$ denotes the retrieved text nodes.

5. If the true answer A exists in the database $\mathcal{D}$, then $P(q, \mathcal{D})$ should contain A within the retrieved elements, so that:

$$A \subseteq (R_{q\mathcal{D}} \cup E_{q\mathcal{D}} \cup T_{q\mathcal{D}}) \quad \text{if} \quad A \subseteq \mathcal{D}$$

6. Next, we define a process g that operates on the query q and the output of $P(q, \mathcal{D})$. The function $g(q, P(q, \mathcal{D}))$ generates an answer A'. If A is contained within the retrieved elements $R_{q\mathcal{D}} \cup E_{q\mathcal{D}} \cup T_{q\mathcal{D}}$, then A' should equal A, such that:

$$A' = A \quad \text{if} \quad A \subseteq (R_{q\mathcal{D}} \cup E_{q\mathcal{D}} \cup T_{q\mathcal{D}})$$

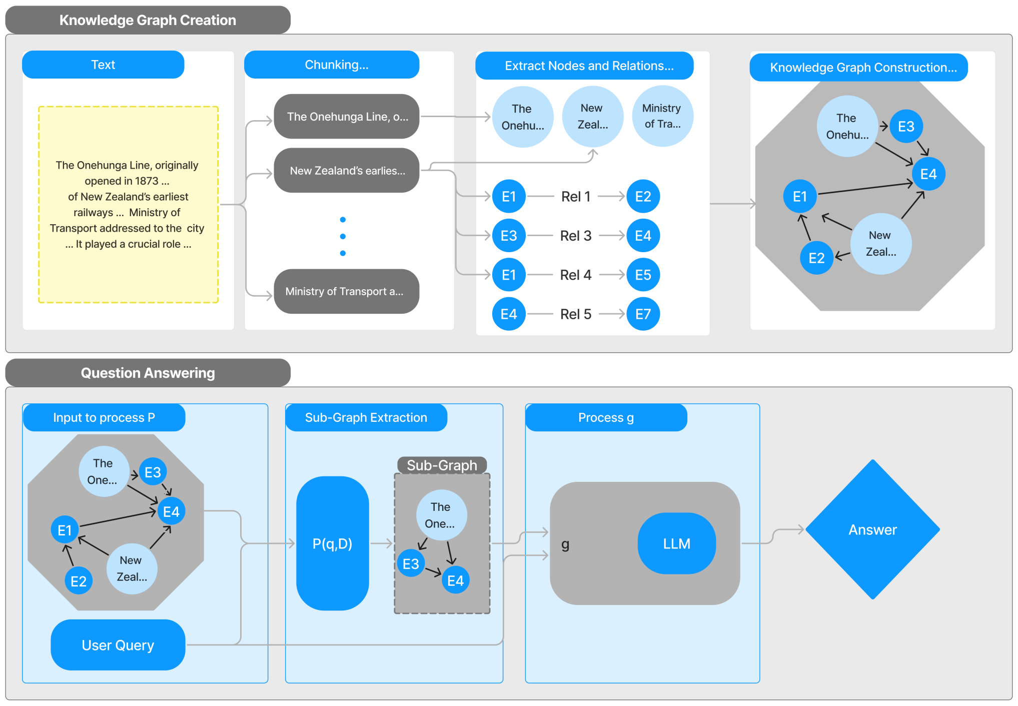

Figure 3.3: Overview of the general process for our method. Starting with text files d_i, we segment each document into chunks c_{ij}. Entities E_{ij} and relations R_{ij} are then extracted from each chunk c_{ij}, represented as $f(c_{ij}) \rightarrow E_{ij} \cup R_{ij}$. These extracted elements, along with the chunks, are used to construct a knowledge graph $\mathcal{D} = \bigcup_{i,j}(E_{ij} \cup R_{ij} \cup \{c_{ij}\})$. Given a query q, the process $P(q, \mathcal{D})$ retrieves a subgraph consisting of relations $R_{q\mathcal{D}}$, entities $E_{q\mathcal{D}}$, and text nodes $T_{q\mathcal{D}}$. If the true answer A exists within the database $\mathcal{D}$, it is expected to be part of the retrieved subgraph such that $A \subseteq (R_{q\mathcal{D}} \cup E_{q\mathcal{D}} \cup T_{q\mathcal{D}})$. Finally, the process $g(q, P(q, \mathcal{D}))$ generates an answer A', where $A' = A$ if the correct answer A is within the retrieved elements.

Following this section, we will describe the details and algorithms used for each of the steps above and the method for building the chatbot app implementing the RAG system.

3.4 OIA Documents Collection

To collect the data from OIA documents [66], we built a crawler, the code of which can be found in the NZTA GraphRAG GitHub project [67]. We manually inspected the files and observed that they could be categorized into two independent groups with two classes each: files with responses in an annexed file, files with responses in the same file, and files with responses in tabular format or text format.

Given that this project's focus was on improving the retrieval algorithm and that we had limited time, a sample of 100 random files with predominant textual answers and without annexed answers were selected. The process involved hand checks to ensure the selected files had enough textual content to generate questions with textual answers. The files were first scraped from the website with a ptyhon crawler, indexed and separated into annexed and not annexed OIA requests. Then, we downloaded a random sample of 200 of the non annexed files and manually inspected them to be adequate with enough text material to create two questions until we found 100 of them that satisfied the condition. We created the questions with the help of ChatGPT by uploading the files into the app and requesting it to generate two questions related to the content of the OIA letter and such that their answers could be found in the letter. The instructions guided the chat agent: 'You are encouraged to create your own questions, but it is ok if you output the same question made as a request to NZTA .' As a result, we obtained 200 questions with answers and the respective passages that we later uploaded into the graph database.

3.5 SP-CoT Transport Passages Generation

Having a suitable dataset for the evaluation of multi-hop questions can be a challenge in itself. A growing amount of research studies how one can artificially create such a set.

In this work, we generated the second datasets used for evaluation of the RAG systems following the the methods of 'Self-prompted Chain-of-Thought on Large Language Models for Open-domain Multi-hop Reasoning' Wang et al. [15]. Their work addresses the problem of generating high-quality datasets for multi-hop open-domain question answering. They propose a simple but clever solution to automate this process using LLMs, which would otherwise require considerable human work. SP-CoT relies on a multi-stage process where the LLM iteratively creates 2-hop question-answer pairs, which can then be combined to create longer multi-hop chains. The output of the process is a set of text passages that are independent of one another but interconnected through key words so that a question that requires reasoning steps across different texts can be formulated. The process is divided into two stages: 2-hop QA Via self-generation and Multi-hop QA via composition. Thanks to LLMs' general knowledge, the method can be applied to generate passages on practically any subject. Given our domain of interest, we oriented their approach to generating a dataset related to transport in New Zealand and the New Zealand Transport Agency.

There is a potential risk of generating inaccurate data using this method. However, this does not pose a concern for our study, as the dataset is solely intended to evaluate the RAG system's performance in answering questions given a provided context, rather than serving as a factual source of information. The specific generation steps are detailed below.

2-hop QA via Self-Generation. This process is broken down into four steps:

1. **Step 1 First Passage Generation:** In this step, the LLM is asked to generate a set of keywords k_1 related to a subject and then to generate a passage p_1 about each of those keywords.
2. **Step 2 QA Generation:** From the generated passages, a set of entities a_1 are extracted by prompting the LLM to do so and using Spacy2 and NLTK libraries. These entities will be the answers to questions. Then the LLM is prompted to generate a question q_1 where the answer is a_1 and that is based on the passage p_1. An explanation e_1 for the answer is also generated.

3. **Step 3 Second Hop Passage Generation:** In this stage, step 1 is repeated but using the answers from step 2 a_1 as the keywords k_2 to generate new passages p_2.

4. **Step 4 Secon Hop QA Generation** Similar to step 2, we generate candidate answers a_2 for the passages p_2 in step 3. However, we reject answers that are already keywords of the first hop answer set $a1$. Then again, the LLM is prompted to generate a question based on the passage $p2$ where the answer is $a2$ and where the question contains a_1. Again an explanation e_2 for the answer a_2 is also generated.

The first stage gives us the building block components to construct multi-hp QA pairs. A set of 2-hop quadruplets $(p_1, q_1, a_1, e_1) \rightarrow (p_2, q_2, a_2, e_2)$. With this approach, we can build multi-hop QA sets through a process called composition, which involves linking multiple 2-hop QA quadruplets. The detailed steps for this composition process are outlined below.

1. **Step 1 Composition:** In this step, quadruplets from the first generation are chained together in 6 different types as illustrated in figure 3.4. When doing this, it is ensured that if an entity/answer a_i to a question q_i is present in another question, it can only be present in the next hop question q_{i+1}. This way, shortcuts are avoided.

2. **Step 2 :** After step one, many of the generated multi-hop chains might be similar. In this step, we prune down the chains to avoid having sets of chains within the same type of hops that are too similar. We do this by keeping track of all the questions that have been included in the so far created multi-hop chain (of a particular type) and only allow the next multi-hop chain to enter the set if the questions it contains that already exist in the list is smaller than a threshold. The threshold was 1 for our purposes, allowing a max of 1 repeated question.

3. **Step 3** Yes/No Question Generation: In step 3, the original algorithm generates yes or no questions based on the already generated chains. In this paper, we do not make use of these question sets.

4. **Step 4 :** Now that the reasoning chains have been created, we can compose the questions to generate a single high-quality multi-hop question. The following iterative procedure is followed. If a_i is the answer to an intermediate question q_i then in the following hop question q_{i+1}, which by construction contains a_1, a_1 is replaced with q_1. This is followed until reaching the final question q_n.

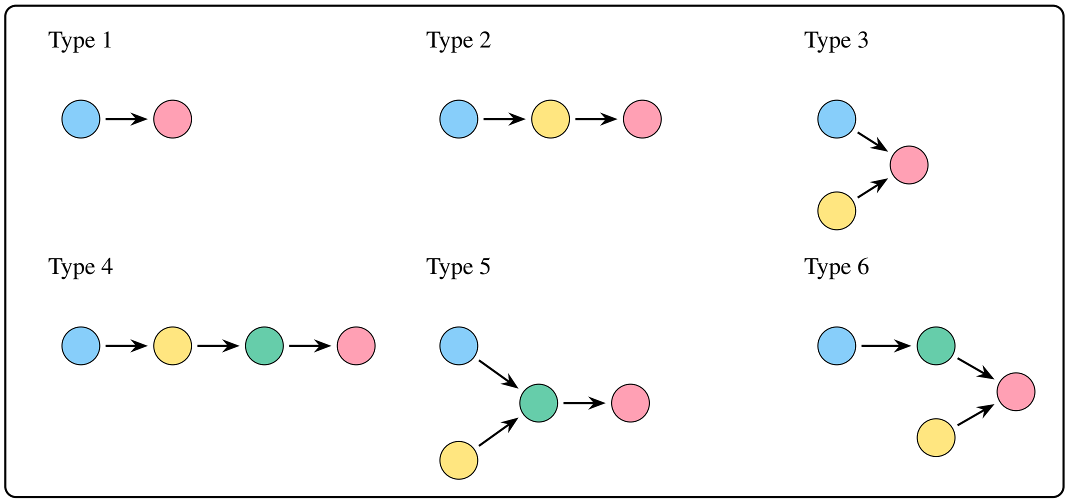

Figure 3.4: Illustration of different multi-hop question types for the reasoning chain generation process.

In each 2-hop quadruplet, the answer to the first hop (a_1) is a keyword that appears in the second question (q_2). This allows us to replace the keyword a_1 in q_2 with the first question q_1. This substitution creates a new composed question that requires answering q_1 first in order to obtain the necessary information to answer q_2. We can further extend this method to create even more complex multi-hop questions. Like the chains illustrated above.

Let us illustrate with an example what type of generated questions this logic results in. The first three are displayed here. For the remaining list, refer to the appendix A.1

Figure 3.5: Examples of multi-hop questions (MHQ) and answers across different types.

Type 1
MHQ: `Which city is linked to the urban area in New Zealand with significant transport investment alongside Auckland and Wellington by SH1?`
Answer: Picton
Q1: `Which urban area in New Zealand has significant transport investment alongside Auckland and Wellington?`
A1: Christchurch
Q2: `Which city is linked to` **`Christchurch`** `by SH1?`
A2: Picton

Type 2
MHQ: `What program, which started in the year when the Onehunga Line was reintroduced between the two locations connected by New Zealand's first railway in 1873, requires emissions testing for older vehicles?`
Answer: Vehicle Exhaust Emissions Testing Programme
Q1: `What two locations were linked by New Zealand's first railway in 1873?`
A1: Auckland and Onehunga
Q2: `In what year was the Onehunga Line reintroduced between` **`Auckland andOnehunga`**`?`
A2: 2010
Q3: `What program, started in 2010, requires emissions testing for older vehicles?`
A3: Vehicle Exhaust Emissions Testing Programme

Type 3
MHQ: `What is the destination of the scenic passenger rail service in New Zealand known for showcasing stunning landscapes, particularly through the Southern Alps, departing from the urban area in New Zealand that has experienced significant investment in public transport infrastructure alongside Auckland and Wellington?`
Answer: Greymouth
Q1: `What is the name of the scenic passenger rail service in New Zealand known for showcasing stunning landscapes, particularly through the Southern Alps?`
A1: TranzAlpine
Q2: `Which urban area in New Zealand has experienced significant investment in public transport infrastructure alongside Auckland and Wellington?`
A2: Christchurch
Q3: `What is the destination of the` **`TranzAlpine`** `service departing from` **`Christchurch`**`?`
A3: Greymouth

A set of 200 passages and 116 multi-hop questions were generated through the SP-CoT method using GPT4o-mini. To generate the questions, we used the a modified version of the code from [15]. The original can be found at [68]. The script requires us to set a theme for keyword generation as well as as the number of key words to generate. The theme set for the generation was:

"New Zealand transport agency and New Zealand transport related topics (road, projects, safety, vehicles, personnel, organization, etc)"

We set the parameter to generate 100 initial key terms (k_1). However, we observed that the original SP-CoT code often produced many duplicate key terms. To address this, we made some modifications. Specifically, during the iterative process of generating key terms, we passed the current list of generated terms as context. The model was then prompted to produce additional key terms that were not already included in the list. The modified version of the code is available in our fork [69].

3.6 Data Pre-processing and Knowledge Graph Building

The steps for building the knowledge graph from the documents are as follows. First, read each document, tokenize it and separate it into chunks of 512 tokens with an overlap of 20 tokens. We give each chunk of text to an LLM agent instructed to extract triplets from the text. The full prompt can be found in Appendix A.3. In summary, the LLM is asked to:

1. **Extract Knowledge Triplets**: Extract up to 15 triplets from the passage which contain relevant knowledge.
2. **Set Ontology**: The preferred entity types and relations are listed: *People*, *Organizations*, *Locations*, *Processes*, *Agreements*, *Buildings*, *Streets*, *Projects*, *Events*, *Policies*, *Rules*, and *Laws*.
3. **Output Guidelines**: A Jason example of how to output the response is provided.
4. **Preliminary disambiguation**: For any reference to the "New Zealand Transport Agency," create the entity node with that exact name. To avoid variations like NZTA, NZ Transport Agency, and Waka Kotahi.
5. **General Pruning and Cleanliness**: A list of guidelines on which entity types to avoid, like the self-reference of documents or the term 'request,' common in all OIA files. In addition to instructions, stick to primitive data types like strings and numbers.

The model used for the triplet extraction agent was 'llama3-groq-70b-8192-tool-use-preview' offered for free by Groq. Through experimentation and trial and error, we found that out of the models offered by Groq, this model was the best at following the required format instructions and extracting triplets.

Before loading the data into the Neo4j database, we embed all nodes, triplets, and text from the text nodes. The triplet embedding is derived from the embedding of its associated relation. Each embedding is then stored as a property within its respective graph object. Specifically, the triplet embedding includes information from the head and tail entities of the relation. Node embeddings are created using the default LlamaIndex method, incorporating the entity name and metadata. Text chunks are also embedded and stored as independent “text nodes” in the graph database. Each text node is linked with an entity node through a ‘MENTIONS’ relationship, indicating which entities are referenced in the text. We used the ”BAAI/bge-small-en-v1.5” model for all embeddings. The related code is available at [70].

3.7 Query and retrieval

We will now go over the sub-graph retrieval process $P(q, \mathcal{D})$ and compare the LlamaIndex native method with our improved “CSMR”” method. The steps are the following.

1. **Get Seed Nodes**: To start, the query q is embedded, and the k most similar nodes based on cosine similarity are retrieved.

2. **Neighbor Extraction**: The neighbors up to a depth d for each seed node are fetched and kept as triplets

3. **Neighbor Triplet Sorting**:

Improved Method	**Original Method**
For each Seed node, rank its triplet neighbors according to cosine similarity with the user query q. Select up to 7 most relevant triplets for each seed node.	Order the extracted triplets by selecting all triplet neighbors of seed node 1, then the ones of seed node 2, and so on until seed node 4. Cut the list at the topmost 30. Those are the selected triplets.

4. **Text Chunk Selection**:

Improved Method	**Original Method**
For each triplet, we select the text chunk associated with its head node with the highest cosine similarity to the user query.	The text node of the last processed text chunk, which produced the head entity in the knowledge graph creation step, is selected.

5. **Preparing Triplets For Context**:

Improved Method	**Original Method**
We find the shortest path to each of the selected triplets from their seed node. If there is more than one shortest path, we find the sum of the cosine similarity of each triplet in the path and select the path with the highest total cosine similarity.	The triplets remain just as triplets.

6. **Context creation**: The paths (New method) or triplets (Original method) are grouped according to the text chunk each retrieved. They are organized as a list followed by the text chunk.

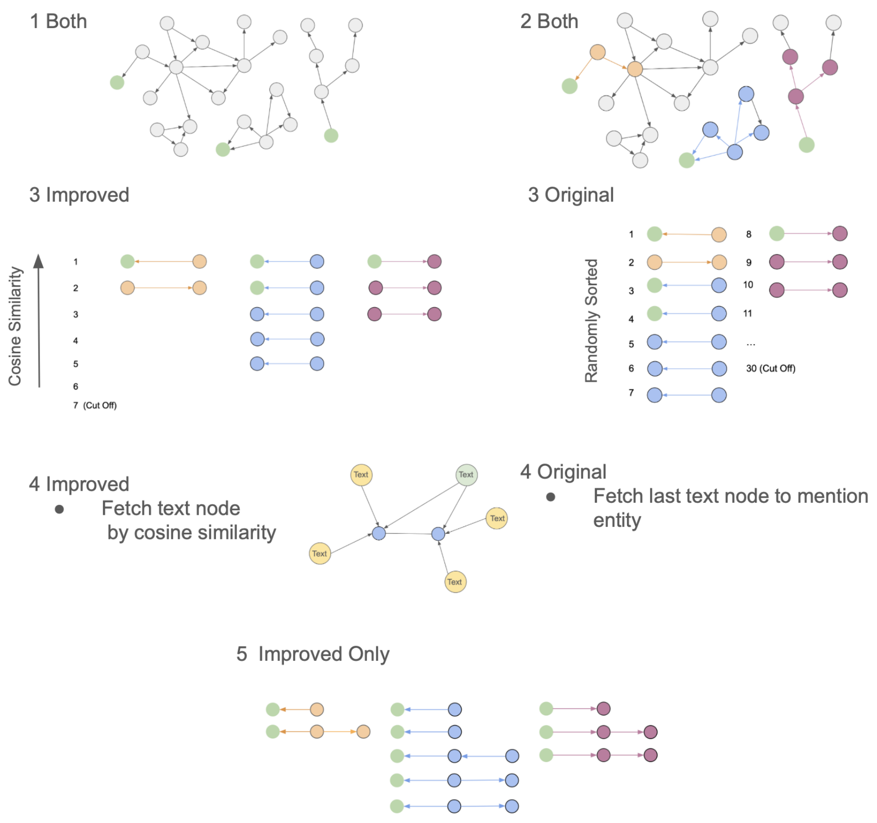

Figure 3.6: Comparison of Node Retrieval Processes for Original and Improved Methods. The node retrieval process consists of four main steps. (1) Both methods start by identifying the most similar nodes to the user query based on cosine similarity. (2) The neighbors of these seed nodes are retrieved up to a specified depth d. (3) In the improved method, triplets are sorted by query-triplet cosine similarity, and only the top 7 triplets per seed node are selected. In contrast, the original method retrieves all neighbors without sorting and selects up to 30 triplets, cutting off arbitrarily. (4) The improved method selects the text chunk associated with the head node of each triplet that has the highest cosine similarity with the user query. The original method, however, selects the text chunk based on the last processed mention of the entity. Finally the improved CSMR adds multi-hop chains to the extracted triplets, connecting neighbors with the seed nodes by cumulative relation cosine similarity.

The original method is not ideal, given that many irrelevant relations will likely be selected. At the same time, many relevant relations will be cut off due to the lack of relevance sorting and the limit cut at 30 triplets. In addition, in the original algorithm, the text node retrieved with each triplet is the last text node associated with the head of the triplet when creating the knowledge graph, which is not necessarily the most relevant text node.

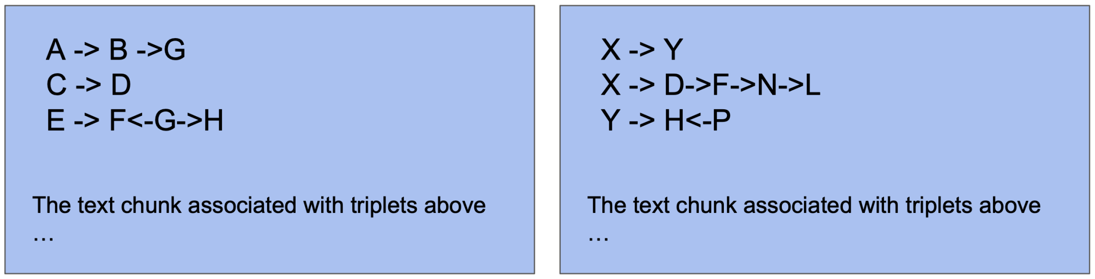
A -> B ->G
C -> D
E -> F<-G->H
The text chunk associated with triplets above
…
X -> Y
X -> D->F->N->L
Y -> H<-P
The text chunk associated with triplets above
…

Figure 3.7: All triplets that returned the same text node are grouped together along with the text as a single text structure. The number of these groups retrieved equals the number of text nodes retrieved.

In the next step, the extracted chunks go through a synthesizer. Before being processed, the context paths-text pairs are grouped to maximize the amount of context that can be given to the LLM by sending multiple context text chunks in one single go. This is another way, other than the multi-hop chains in the context, by which multi-hop relations can be revealed to the LLM model both for the original and CSMR methods. Although the original model does not include the full chain multi-hop paths in each context path-text pair, it may still be able to make multi-hop associations if the right combinations of context groups are passed to the LLM at the same time.

The synthesizer iteratively passes the context to the LLM agent responsible for extracting an answer. The first iteration follows a prompt that asks the model to answer the user query based on the context[71]. Every subsequent iteration follows a prompt that receives the query as input, the new context, and the answer the model gave in the previous iteration. This time, the model is asked to refine the original answer based on the new context. Both prompts can be found in the appendix A.4. When all context has been passed, the last answer from the agent is returned to the user.

3.8 Chat Agent Architecture

We built the RAG chat agent from 3 main components: A chatbot component, a query tool, and the knowledge graph. We now detail the tasks each component is assigned.

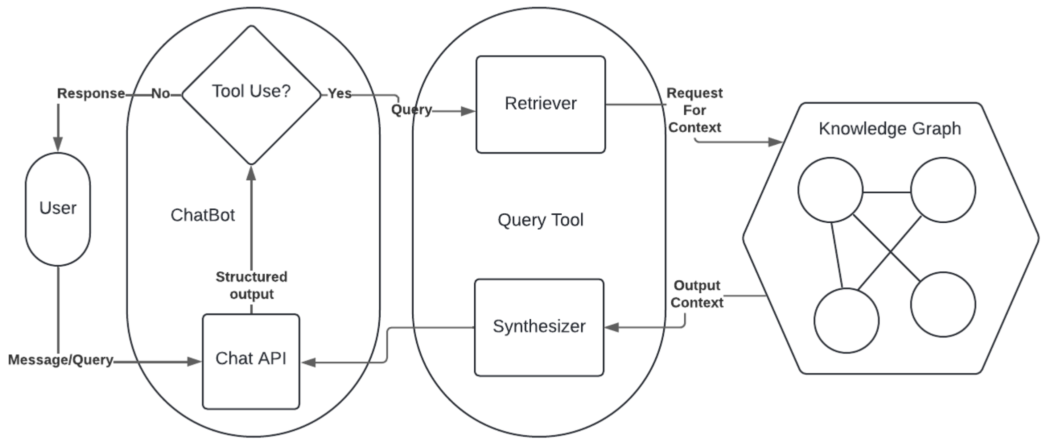

Figure 3.8: The RAG engine is comprised of three main components: The Chatbot module, in charge of receiving a user message and evaluating if it needs a tool-call to respond; The query-tool, contains the components that retrieve data from the Knowledge graph and also the synthesizers which process the output from the Knowledge graph to get a response; Finally the Knowledge graph.

The chatbot module is the component the user directly interacts with. It comprises an LLM inference endpoint wrapped around a LlamaIndex FunctionCallingAgentWorker class [72, 73]. The message the model receives has three parts: the system prompt, the chat history, and the new prompt, which can be either a user message or a response from a function call done by the agent. The system prompt states the purpose of the chatbot. In this case, it states that it is a handy chat assistant that assists users with queries related to transport in New Zealand and that it has access to a tool that can ask questions to a NZTA-related database (our knowledge graph). The system prompt also contains information on the tools it can use, and the the arguments it requires. Based on this information, the LLM can judge whether the query requires a function call. The chat agent is also instructed to output its response in a structured format [74]. This way, we can identify whether it has decided to do a function call from its response. If it has, the arguments to the function are parsed from the response, and the function is called with the argument.

The second component of the chat engine is the query tool, which is made of a retriever and an LLM agent-synthesizer. The retriever receives the input query provided to the function call by the LLM, embeds it, and retrieves the context through the process described in Subection 3.7 Query and Retrieval.

Finally, the last component of the system is the graph database, which contains the nodes and relations extracted from the source OIA/SP-CoT Texts and the text chunks from where they came.

3.9 Chatbot App

We built the prototype chatbot application, implementing the chatbot engine in a simple setup consisting of a Javascript front-end web app through a Flask backend. The app is a toy model that exemplifies how one could deploy a rag system utilizing the rag engine previously described. It consists of a simple chat window with an input text below it. When a user types and sends a message, javascript calls the chat endpoint with the message as input. The endpoint returns the chatbot response, which the Javascript front end displays in the chat box. The setup is done with a Neo4j database, 'BAAI/bge-small-en-v1.5' embedding model and the model used for the chat agent as well as the query agent is llama3-groq-70b-8192-tool-use-preview provided by Groq API through a LlamaIndex wrapper class. The setup is modular; hence, any of the previously mentioned components could easily be changed to others if desired. This is possible thanks to llamaindex's broad support for different databases and its integration with LLM endpoint providers. The repository with the project can be found at [67].

3.10 Evaluation

To compare the performance of the different algorithms, we make comparisons at node retrieval and question answering. The sample questions were answered using a seed node retrieval limit of 4 and for the neighbor depths of 1-6.

The evaluation of the retrieval method consisted of the following process. First, for every question, the results of the retrieved nodes at stage 3 in figure 3.6 are saved along with the question. Then, the triplets of both sets (CSMR and Original) are given an ID and combined into a single set, removing duplicate pairs. We pass the set of triplets to ChatGPT through its API with a prompt that asks it to first rank the triplets according to their relevance with respect to the question and to assign binary score stating whether the triplet is relevant or not (1 or 0). We map back the relevance and rank to the individual lists of CSMR and Original retrieved triplets. With this information, we can proceed to calculate the metrics. The code and documentation are found at the experiment repository [70].

To evaluate retrieval, we use MRR, nDCG@28, Precision@28, Precision@14, and Precision@7, whose definitions can be found at 2.6 (Evaluation of Retrieval-Augmented Systems). We use BLUE, ROUGE, and cosine similarity metrics to evaluate and compare the model response with the reference answer. In addition to these metrics, we use GPT4o to evaluate whether the answers given by the model were correct. For the SP-CoT, which are QA paris with short answers, we directly ask GPT4o to tag it as correct, incorrect, or not found. For the OIA set, which contains long answer type of questions, simply asking to mark the response as correct or incorrect was not appropriate given that the response could be partially correct or incomplete. For this reason, we modified the approach to more qualitative analysis and asked GPT4o to tag the responses in the following way: We asked GPT4o to label the answer as correct, partially correct, wrong, or not found. In addition, we ask it to select which model outputs a better answer, the possibilities being 'sorted,' 'unsorted,' 'tie,' or 'neither.' The prompt used for this purpose can be found in the annex A.6.

Chapter 4

Experiment & Results

4.1 Experiment Set-Up

We now summarize the Experimental setup. The configuration for the data ingestion is shown in table 4.1.

Table 4.1: Knowledge Graph Creation Summary

Configuration Parameter	**Value**
Max Triplets per Chunk	15
Number of Files / Passages	OIA: 100, SP-CoT: 205
Number of Questions	OIA: 200, SP-CoT: 116
Chunk Size	512
Chunk Overlap	20
LLM Model	`llama3-groq-70b-8192-tool-use-preview`
Embedding Model	`BAAI/bge-small-en-v1.5`
Path Extractor	`DynamicLLMPathExtractor` from LlamaIndex

We looped over each question to evaluate the system's performance, passing it to the query-engine component directly, and stored the answer. The model used for synthesis and question answering was "meta/meta-llama-3-70b-instruct" provided by Replicate with a temperature of 0. Although the KG triplets were extracted with Groq and 'llama3-groq-70b-8192-tool-use-preview' due to rate limits and the project's time constraints, the evaluation was switched to Replicate to speed up the process.

The prompts for the SP-CoT data synthesizer were adapted to better suit the short response nature of this particular dataset. The modifications can also be found in the appendix A.4. This is the only difference in the experiment setup of the two datasets.

The process was repeated for a max seed node retrieval of 4 and neighbor depths $d \in \{1, 2, 3, 4, 5, 6\}$. As detailed in the methods section, the retrieved triplets were stored on each iteration, maintaining their rank for evaluation.

The retrieved triplets were tagged as relevant or not and ranked by relevance through the process described in 3.10 (Evaluation).

With the stored rank and relevance information on retrieved triplets, the MRR, nDGC@28, Precision@28, and Precision@14, Precision@7 were calculated. The BLEU, ROUGE, and Cosine Similarity scores were calculated with the responses at answer levelated.

To test for significance, first, a Shapiro-Wilk Normality [60] on the difference of the scores grouped by depth d was performed. We found that the distribution deviated significantly from a normal distribution. For this reason, a t-test was not adequate, and instead, a sign test [61] was performed.

For the SP-CoT data, answers were assessed for correctness using GPT-4o, which was prompted with both the reference and RAG-generated answers. To determine the statistical significance of the results, we performed a McNemar test. We attempted the same evaluation with the OIA dataset; however, due to the length and complexity of responses associated with that set, direct assessment by the LLM proved unreliable. The LLM's judgments were inconsistent, particularly when the answers contained partial responses or additional relevant information beyond the reference answer. Consequently, a qualitative analysis, as outlined in section 3.10 (Evaluation), was performed instead.

We now present the evaluation of the retrieval and responses for the original and CSMR algorithm. This Section is divided into two parts corresponding to the OIA dataset and the SP-CoT generated data.

4.2 Overall performance on OIA Dataset

We now present the results for the OIA dataset. We omitted nDCG@14 and nDCG@7 because they did not show significant information not already contained in nDCG@28. CSMR shows a small but significant improvement over the original algorithm for d = 1 and a much more marked improvement for greater neighbor depths d

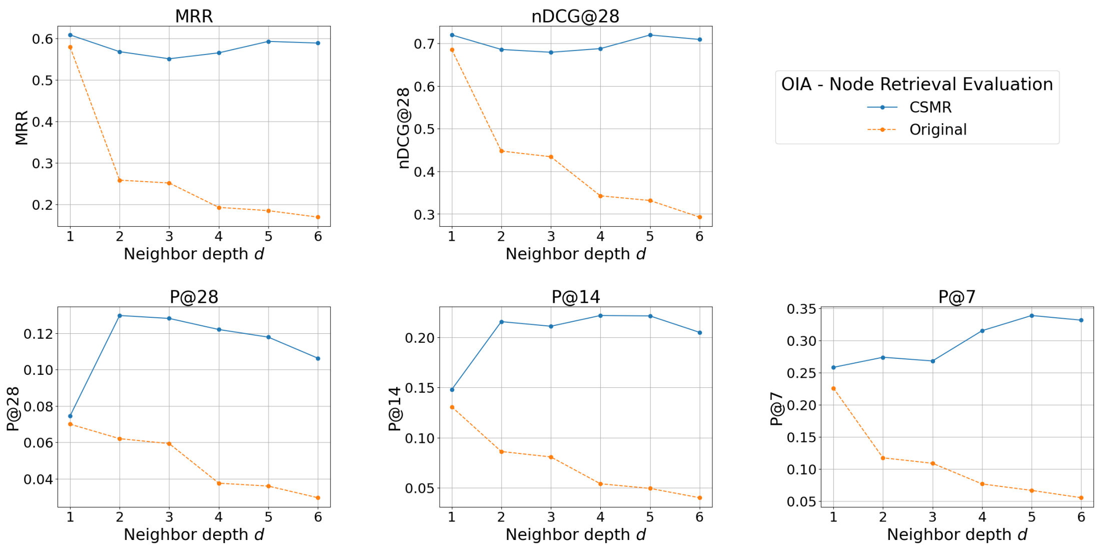

Figure 4.1: This are the results for MRR, nDCG@28, Precision@28, Precision@14 and Precision@7 comparing the original and improved method. The values are close for a depth of 1; however, as depth increases, our method can maintain the level of MRR and nDCG@28 and, at the same time, increase Precision.

The observed outcome aligns with theoretical expectations. The Original algorithm relies only on chance when retrieving relevant nodes, given that it has no sorting. The scores are also expected to be similar for $d = 1$. The native method retrieves up to 30 nodes and CSMR 28. For a depth of 1 neighbor, all the available nodes likely will be within the limit. In this case, the retrieved nodes are going to be the same. However, this is not guaranteed, and there are situations for a depth of 1 neighbor where sorting is relevant, which results in the improved method being significantly better at retrieving relevant triplets.

CSMR can increase the Precision of the retrieval of relevant nodes as depth increases. Additionally, the score improves further for the higher-ranked nodes, shown by the better performance of Precision@7 compared to Precision@14 and Precision@28. However, it only maintains the scores of MRR and nDCG. Improvement in Precision is an indication that more relevant nodes are gathered. A lack of improvement in MRR indicates that the first relevant node is still not correctly placed in the highest rankings as it should. Similarly, the relatively constant nDCG shows that there is still room for improvement in the sorting according to relevance. This highlights a limitation of the proposed method CSMR, suggesting that while it successfully incorporates more relevant nodes into the retrieval set, it does not yet optimize their ranking order effectively.

By doing a pairwise evaluation of significance with a sign test, we found that all differences in performance were significantly better with the new sorted retrieval method.

Table 4.2: Sign Test p-values for Differences in MRR, nDCG@28, P@28, P@14, and P@7 at Different Values of d

d	MRR p-value	nDCG@28 p-value	P@28 p-value	P@14 p-value	P@7 p-value
1	0.0153	0.0013	0.0428	8.21×10^{-4}	1.92×10^{-4}
2	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$
3	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$
4	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$
5	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$
6	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$	$p < 10^{-16}$

We now present the metrics for the system answer evaluation: BLEU, ROUGE-1, ROUGE-2, ROUGE-L, and cosine similarity.

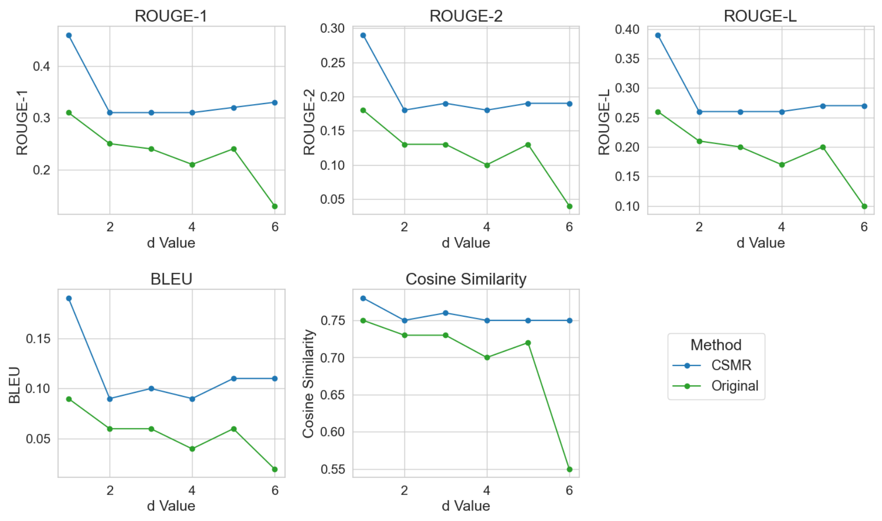

Figure 4.2: Answer evaluations for the NZTA OIA dataset through ROUGE, BLEU, and cosine similarity. The improved method results in better scores all around. However, the performance decreases with an increase in d, which is suboptimal.

The results for the evaluation at the answer level still show an overall improvement by the new method 4.5. However, there is a clear difference compared to the retrieval results in that the new method's performance also decreases as depth increases. This result is explainable by the OIA dataset having single-hop questions. Single-hop questions require just one node to reach the answer; the results suggest that noise is introduced into the context that obscures the relevant information as we increase depth.

For brevity, the p-value table for these metrics is omitted in this section and can be found in the appendix A.2. However, all differences are significant, with each p-value being smaller than 5×10^{-5}.

Figure 4.3 Shows the qualitative evaluation results performed with the LLM. For a single hop, results are comparable between the two models. As d increases, the CSMR remains constant in performance

while the original degrades. This is consistent with the evaluation results at the node retrieval level. Even so, the preferred answer is in favor of CSMR.

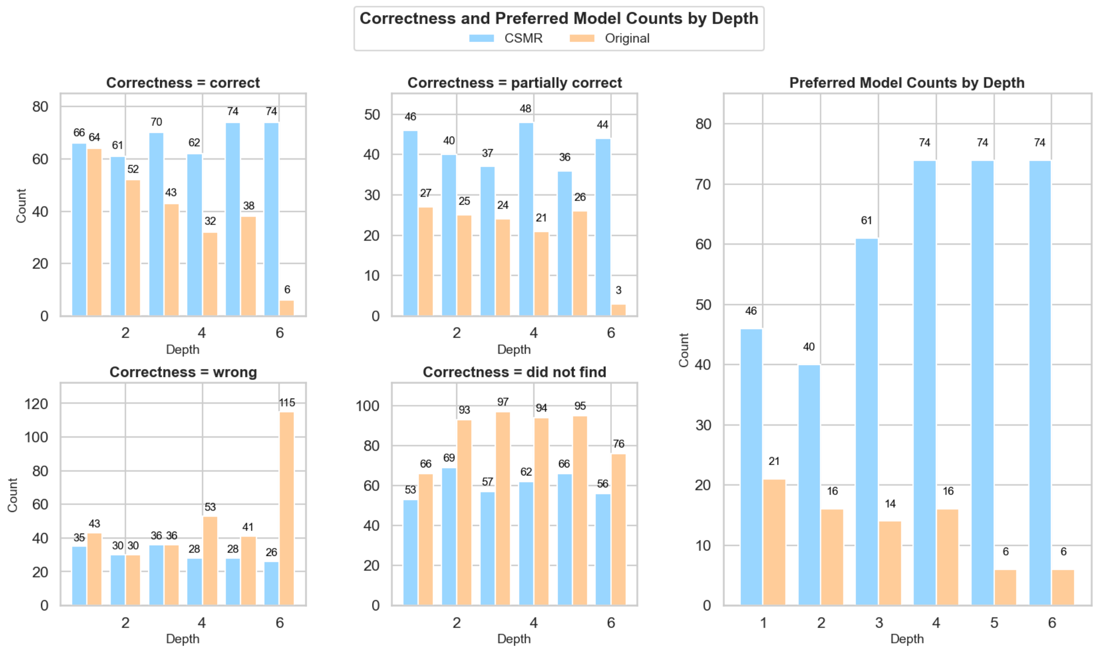

Figure 4.3: Retrieval metrics evaluation for the SP-CoT generated data.

Although there has been an improvement, the total accuracy of the system is still low, with an average of correct and partially correct answers combined 107 out of 200.

4.3 Overall performance on SP-CoT Dataset

We now examine the retrieval evaluation for the SP-CoT transport dataset. It resembles the OIA dataset, although there is a decline in MRR and nDCG@28 performance as we transition from single-hop to multi-hop retrieval. Only Precision@7 increases with d. This indicates that, on average, neighbors from seed nodes other than the highest-scored seed node are introducing noise into the set, implying that seed retrieval could benefit from limiting the number of seed nodes retrieved to fewer than four.

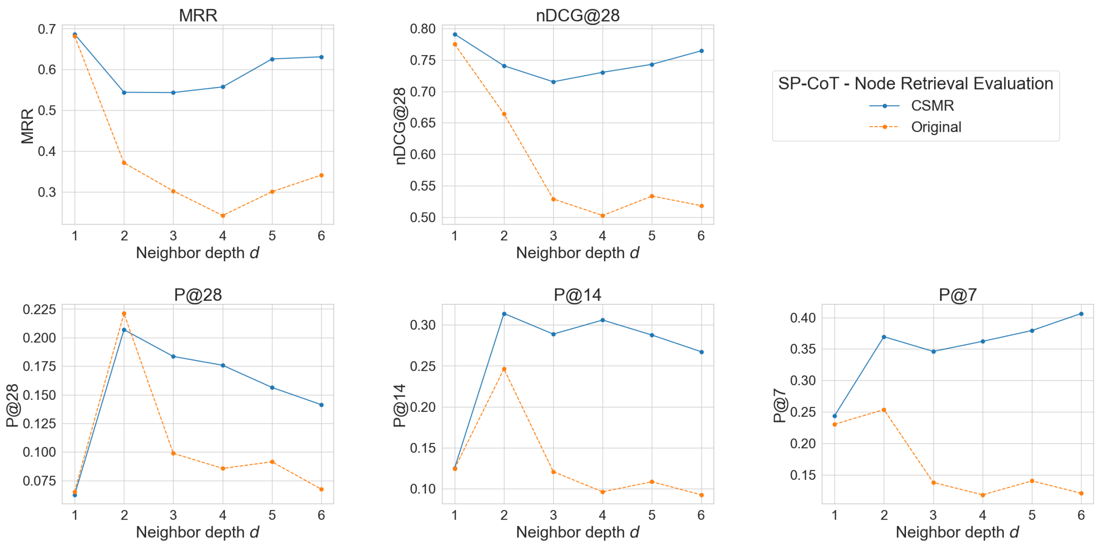

Figure 4.4: Evaluation metrics for the SP-CoT dataset, including MRR, nDCG@28, Precision@28, Precision@14, and Precision@7. The results show a similar trend to the OIA dataset but with a more pronounced decline in MRR and nDCG@28 performance.

The same observations and conclusions can be derived from these plots as they were for the OIA results. This time, there is no statistically significant difference for the single-hop retrieval. For this data set, there was a marked decrease in both MRR and nDCG. Precision@28 and Precision@14 seem more negatively affected by increasing depth d compared to the OIA dataset. Again, they suggest that there can be a benefit in lowering the limit of retrieved seed nodes.

Table 4.3: Sign Test p-values for Differences in MRR, nDCG@28, P@28, P@14, and P@7 at Different Values of d

d	**MRR p-value**	**nDCG@28 p-value**	**P@28 p-value**	**P@14 p-value**	**P@7 p-value**
1	0.754	0.424	0.375	0.97	0.289
2	$p < 10^{-5}$	$p < 10^{-5}$	0.470	$p < 10^{-5}$	$p < 10^{-5}$
3	$p < 10^{-5}$	$p < 10^{-5}$	$p < 10^{-5}$	$p < 10^{-5}$	$p < 10^{-5}$
4	$p < 10^{-5}$	$p < 10^{-5}$	$p < 10^{-5}$	$p < 10^{-5}$	$p < 10^{-5}$
5	$p < 10^{-5}$	$p < 10^{-5}$	$p < 10^{-5}$	$p < 10^{-5}$	$p < 10^{-5}$
6	$p < 10^{-5}$	$p < 10^{-5}$	$p < 10^{-5}$	$p < 10^{-5}$	$p < 10^{-5}$

Let us now look at SP-CoT data evaluations at the answer level. Unlike the OIA data, this time, the responses improve as the depth d increases. This result is consistent with expectations given that the nature of the data was multi-hop, so increasing the neighbor exploration retrieves the necessary context to answer the questions. However, there is something to note. The questions were, at most, 4-hop questions, and our model performance keeps increasing past the depth of 4. This implies that our method does not efficiently retrieve information through the shortest paths. While there is an overall improvement in the evaluation metrics, the statistical significance, found in the appendix A.2, varies across metrics depending on the retrieval depth, with some metrics showing significance at certain depths while others do not.

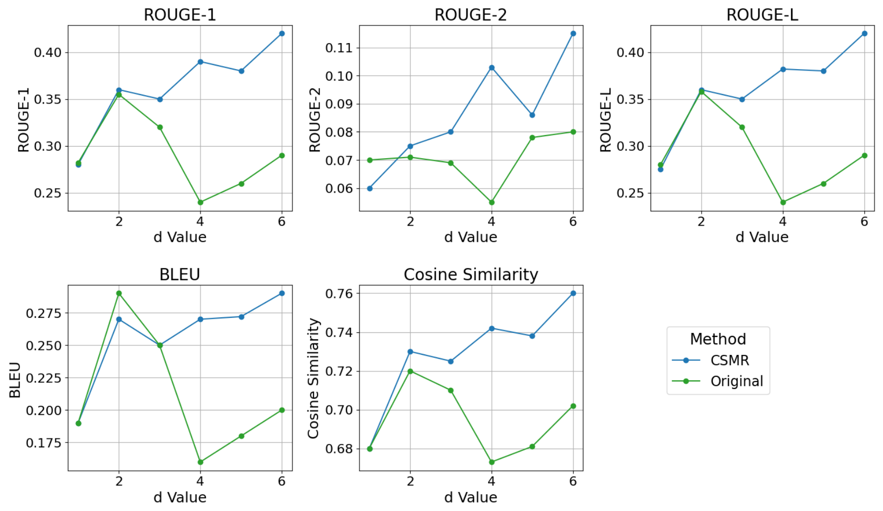

Figure 4.5: BLEU, ROUGE, and Cosine Similarity results for SP-CoT data. This time performance does increase as d increases.

GPT-4o classified the answers from the OIA dataset as either correct or incorrect. The comparison of both methods is presented below. Both methods show similar performance for neighbor explorations up to a depth of three, with CSMR demonstrating an improvement at greater depths. In contrast to the OIA results, the number of correct answers increases as depth increases, which aligns with the multi-hop nature of the new dataset.

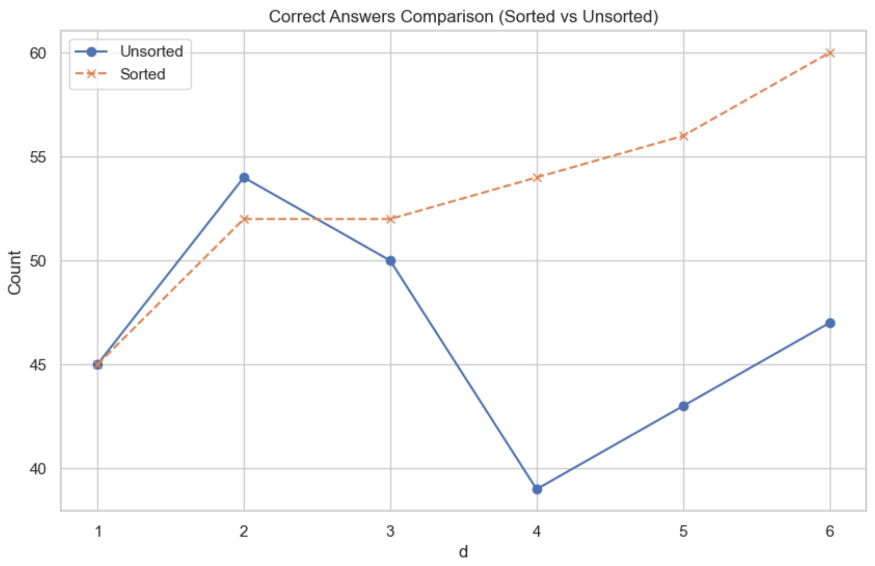

Figure 4.6: SP-CoT data correct answer count as a function of d. It can be seen that performance increases with d

Although the CSMR method demonstrates better performance, the overall accuracy remains relatively low, with a maximum of only 60 out of 116 questions answered correctly.

Table 4.4: McNemar's Test p-values for Exact Matches at Different Values of d

Neighbor Depth	1	2	3	4	5	6
p-value	1.0	0.69	0.86	0.01	0.03	0.03

4.4 Summary

The results demonstrate an improvement in performance with CSMR over the baseline method. MRR, nDGG, and Precision show statistically significant improvements in the sorted node retrieval over the unsorted node retrieval. This happens for the OIA document and the AI-generated SP-CoT datasets. Performance at the answer level is also improved with statistical significance. However, for the single hop OIA dataset, performance decreases with an increased depth of neighbor node retrieval, which is not a desired behavior. An ideal retrieval system should be able to handle multi-hop as well as single-hop retrieval simultaneously. In contrast, for the multi-hop dataset, performance increases with increased depth, which is expected and desirable but with a much lower degree of statistical significance. It is worth noting that the maximum multi-hop depth in questions is 4, and in our evaluation, depths greater than 4 keep improving the results. This suggests that increasing the number of explored neighbors increases the chance of the relevant passages being retrieved; however, the path exploration is not optimal because, in that case, maximum performance would be reached by just exploring neighbors up to a depth of 4. This indicates that sorting relations by cosine similarity between the user query and triplet relations is an effective way of increasing retrieval relevance; however, there is room for improvement. MRR remains constant and does not have a value of 1, meaning that the first relevant item is not the first in the ranking. Similarly, nDCG shows that the overall ranking of nodes still needs to be improved.

Chapter 5

Conclusions & Future Work

In this dissertation, we developed and refined a Retrieval-Augmented Generation (RAG) system tailored to the New Zealand transport domain. Building on the open-source tools provided by LlamaIndex, we proposed a novel sub-graph retrieval method, CSMR, which demonstrated improvements over the baseline algorithm. This method enhances retrieval through cosine similarity-based sorting of neighboring nodes, cosine similarity-based text node selection, and the inclusion of multi-hop relation chains in the context. We evaluated the system on two datasets: NZTA OIA response documents and SP-CoT generated transport-related data, each represented by its own knowledge graph. Performance was measured at both the node retrieval and answer levels, and our proposed method yielded statistically significant improvements on both fronts. However, the system's overall performance is not yet sufficient for production use.

Additionally, we developed a chatbot application with a modular architecture that connects the RAG agent to a front-end chat interface. This system can engage in natural conversations with users and retrieve transport-related information from the knowledge graph.

5.0.1 Limitations

This project was developed on a personal laptop (MacBook Pro M3 chip with 36 GB RAM). This is not ideally suited for running transformer based LLMs locally. For this reason and budget restrictions we relied on Groq's free inference API for the majority of the project. The best performing model they offered at the time according to our experimentation was "llama3-groq-70b-8192-tool-use-preview." It is an advanced model however it does not represent the best existing open-source options, it is surpassed for example by Llama3.1-405B which achieves higher benchmark scores overall [39, 75]. Although there are services that offer this models this are not free and hence our use of them was restricted. We cannot precisely estimate the improvement in performance using a higher end model would result in but it is reasonable to expect measurable improvements.

5.0.2 Future Work

While our CSMR method led to performance gains, it has three notable limitations:

1. MRR and nDCG metrics suggest that the ranking of retrieved nodes could be further optimized. Re-ranking method may yield enhanced results.
2. Answer-level performance declines as neighbor exploration depth increases in the single-hop (OIA) dataset.
3. Results on the SP-CoT dataset indicate that the method does not consistently retrieve the most direct paths to correct answers.

Addressing these issues will require further exploration and refinement.

There are other modules in the RAG not in the main scope of this project which have room for enhancement:

- **Knowledge graph disambiguation**: Implementing techniques to disambiguate entities in the knowledge graph could improve retrieval accuracy.
- **Parsing of tabular data**: Extending data parsing to include tabular information would broaden the scope of ingested documents.
- **Prompt optimization**: Optimizing prompts could refine the quality of responses generated by the RAG system.
- **Implementation of vector databases**: Using vector databases to store node and triplet embeddings could enhance scalability and retrieval speed.

References

[1] OpenAI. Gpt-4 technical report. Technical report, 2023. URL https://cdn.openai.com/papers/gpt-4.pdf. Accessed: November 4, 2024.

[2] Anthony Hartshorn, Aobo Yang, Austen Gregerson, Baptiste Roziere, Chris McConnell, Corinne Wong, Daniel Song, Emily Dinan, Gabriel Synnaeve, Graeme Nail, et al. The llama 3 herd of models. 2024.

[3] Ed Chi, Betty Chan, Dustin Tran, Jeremiah Liu, Yifeng Lu, Martin Baeuml, Jakub Sygnowski, James Lottes, Alban Rrustemi, Shantanu Thakoor, et al. Gemini: A family of highly capable multimodal models. 2023.

[4] Anthropic. Claude ai model, 2024. URL https://www.anthropic.com/claude. Accessed: November 4, 2024.

[5] Mistral AI Team. Mistral ai latest updates, September 2023. URL https://mistral.ai/news/. Accessed: November 4, 2024.

[6] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. Large language models struggle to learn long-tail knowledge. 2022.

[7] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. 2023.

[8] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the ai ocean: A survey on hallucination in large language models. *arXiv.org*, 2023. ISSN 2331-8422.

[9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. 2020.

[10] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. Graph retrieval-augmented generation: A survey. 2024.

[11] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. From local to global: A graph rag approach to query-focused summarization. 2024.

[12] New Zealand Transport Agency. About Us. https://www.nzta.govt.nz/about-us, 2024. URL https://www.nzta.govt.nz/about-us. Accessed: 2024-10-26.

[13] New Zealand Government. Official Information Act 1982. https://www.legislation.govt.nz/act/public/1982/0156/latest/DLM64785.html, 1982. URL https://www.legislation.govt.nz/act/public/1982/0156/latest/DLM64785.html. Accessed: 2024-10-26.

[14] LlamaIndex. Llamaindex - build ai knowledge assistants. https://llamaindex.ai/. Accessed: November 5, 2024.

[15] Jinyuan Wang, Junlong Li, and Hai Zhao. Self-prompted chain-of-thought on large language models for open-domain multi-hop reasoning. 2023.

[16] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 2013.

[17] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. 2013.

[18] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

[19] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051. URL https://aclanthology.org/Q17-1010.

[20] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1202. URL https://aclanthology.org/N18-1202.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv.org*, 2023. ISSN 2331-8422.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[23] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL http://arxiv.org/abs/1907.11692. cite arxiv:1907.11692.

[24] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

[25] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. In Eduardo Blanco and Wei Lu, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2029. URL https://aclanthology.org/D18-2029.

[26] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[27] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[28] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. 2020.

[29] Eli Collins and Zoubin Ghahramani. Lamda: our breakthrough conversation technology, May 2021. URL https://blog.google/technology/ai/lamda/. Accessed: November 4, 2024.

[30] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Gehrmann, et al. Palm: Scaling language modeling with pathways. 2022.

[31] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068, 2022. URL https://api.semanticscholar.org/CorpusID:248496292.

[32] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. 2023.

[33] Mistral AI Team. Why we're building mistral ai, September 2023. URL https://mistral.ai/news/about-mistral-ai/. Accessed: November 4, 2024.

[34] Hugging Face. Hugging face: The ai community building the future, 2024. URL https://huggingface.co/about. Accessed: November 4, 2024.

[35] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. 2022.

[36] Arghavan Moradi Dakhel, Vahid Majdinasab, Amin Nikanjam, Foutse Khomh, Michel C. Desmarais, and Zhen Ming (Jack) Jiang. Github copilot ai pair programmer: Asset or liability? *The Journal of systems and software*, 203:111734–, 2023. ISSN 0164-1212.

[37] LangChain. Langchain: Applications that can reason. https://www.langchain.com, 2024. Accessed: 2024-11-05.

[38] Mistral AI. Mistral 7b benchmarks, 2024. URL https://mistral.ai/news/mistral-large/. Accessed: 2024-11-10.

[39] Meta AI. Introducing llama 3.1: Our most capable models to date. https://ai.meta.com/blog/meta-llama-3-1/, July 2024. Accessed: 2024-11-07.

[40] Dennis Abts, Jonathan Ross, Jonathan Sparling, Mark Wong-VanHaren, Max Baker, Tom Hawkins, Andrew Bell, John Thompson, Temesghen Kahsai, et al. Think fast: a tensor streaming processor (tsp) for accelerating deep learning workloads. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 145–158, Piscataway, NJ, USA, 2020. IEEE Press. ISBN 1728146615.

[41] Replicate. Replicate: Community-driven ai model deployment platform, 2024. URL https://replicate.com. Accessed: 2024-10-31.

[42] Hugging Face. *Hugging Face Documentation*. URL https://huggingface.co/docs. Accessed: [Date].

[43] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 641–649, New York, NY, USA, 2024. ACM. ISBN 9798400704314.

[44] Renzo ANGLES and Claudio GUTIERREZ. Survey of graph database models. *ACM computing surveys*, 40(1):1–39, 2008. ISSN 0360-0300.

[45] Inc. Neo4j. Neo4j official website. https://neo4j.com, 2024. Accessed: 2024-10-08.

[46] Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. Scalable multi-hop relational reasoning for knowledge-aware question answering. 2020.

[47] Yifu Gao, Linbo Qiao, Zhigang Kan, Zhihua Wen, Yongquan He, and Dongsheng Li. Two-stage generative question answering on temporal knowledge graph using large language models. *arXiv preprint arXiv:2402.16568*, 2024. URL https://arxiv.org/abs/2402.16568.

[48] Zhen Han, Yue Feng, and Mingming Sun. A graph-guided reasoning approach for open-ended commonsense question answering. *arXiv preprint arXiv:2303.10395*, 2023. URL https://arxiv.org/abs/2303.10395.

[49] Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. 2024.

[50] Mathav Raj J, Kushala VM, Harikrishna Warrier, and Yogesh Gupta. Fine tuning llm for enterprise: Practical guidelines and recommendations. 2024.

[51] Tiezheng Guo, Qingwen Yang, Chen Wang, Yanyi Liu, Pan Li, Jiawei Tang, Dapeng Li, and Yingyou Wen. Knowledgenavigator: leveraging large language models for enhanced reasoning over knowledge graph. *Complex Intelligent Systems*, 10(5):7063–7076, 2024. ISSN 2199-4536.

[52] Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. Structgpt: A general framework for large language model to reason over structured data. 2023.

[53] Jinhao Jiang, Kun Zhou, Wayne Xin Zhao, Yang Song, Chen Zhu, Hengshu Zhu, and Ji-Rong Wen. Kg-agent: An efficient autonomous agent framework for complex reasoning over knowledge graph. 2024.

[54] Julien Delile, Srayanta Mukherjee, Anton Van Pamel, and Leonid Zhukov. Graph-based retrieval-augmented generation for large language models. *TBD*, 2024.

[55] Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. 2024.

[56] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. Grag: Graph retrieval-augmented generation, 2024. URL https://arxiv.org/abs/2405.16506.

[57] Mean reciprocal rank, 2018.

[58] Olivier Jeunen, Ivan Potapov, and Aleksei Ustimenko. On (normalised) discounted cumulative gain as an off-policy evaluation metric for top-n recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1222–1233, New York, NY, USA, 2024. ACM. ISBN 9798400704901.

[59] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

[60] Samuel Sanford Shapiro and Martin Bradbury Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965. doi: 10.2307/2333709. URL https://doi.org/10.2307/2333709.

[61] Kalimuthu Krishnamoorthy. *Handbook of Statistical Distributions with Applications*. Chapman and Hall/CRC, 1st edition, 2006. doi: 10.1201/9781420011371. URL https://doi-org.ezproxy.auckland.ac.nz/10.1201/9781420011371.

[62] J. P. Royston. An extension of shapiro and wilk's w test for normality to large samples. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(2):115–124, 1982. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2347973.

[63] Patrick Royston. Remark as r94: A remark on algorithm as 181: The w-test for normality. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 44(4):547–551, 1995. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2986146.

[64] Matilda Q. R. Pembury Smith and Graeme D. Ruxton. Effective use of the mcnemar test. *Behavioral ecology and sociobiology*, 74(11):1–9, 2020. ISSN 0340-5443.

[65] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. doi: 10.1007/BF02295996. URL https://doi.org/10.1007/BF02295996.

[66] Waka Kotahi NZ Transport Agency. Official information act responses, 2024. URL https://www.nzta.govt.nz/about-us/official-information-act/official-information-act-responses/. Accessed: 2024-11-11.

[67] Felipe Navarro. NZTA-GraphRAG: Retrieval-augmented generation system for the new zealand transport agency knowledge graph, 2024. URL https://github.com/fnavarro94/NZTA-GraphRAG. Accessed: 2024-10-31.

[68] Jinyuan Wang. SP-CoT: Self-prompted Chain-of-Thought. https://github.com/noewangjy/SP-CoT, 2024. Accessed: 2024-10-28.

[69] Felipe Mateo Navarro Cordero. SP-CoT: Self-prompted Chain-of-Thought (Fork). https://github.com/fnavarro94/SP-CoT, 2024. Accessed: 2024-10-28.

[70] Felipe Mateo Navarro Cordero. NZTA-GraphRAG Experiment. https://github.com/fnavarro94/NZTA-GraphRAG-Experiment, 2024. Accessed: 2024-10-28.

[71] LlamaIndex Contributors. Llamaindex, 2023. URL https://github.com/run-llama/llama_index/blob/d9a219e8226452ead4e4e78433438d616d1ebdc6/llama-index-core/llama_index/core/prompts/default_prompts.py.

[72] LlamaIndex. Function calling agent example, 2024. URL https://docs.llamaindex.ai/en/stable/examples/workflow/function_calling_agent/. Accessed: 2024-11-11.

[73] LlamaIndex Developers. Llamaindex function calling agent code. https://github.com/run-llama/llama_index/blob/6dc9b63f8b6b859c1fc72c00a0f9b7e5ffdc6a70/llama-index-core/llama_index/core/agent/function_calling/step.py#L86, 2024. Accessed: 2024-11-11.

[74] LlamaIndex. Structured outputs guide, 2024. URL https://docs.llamaindex.ai/en/stable/module_guides/querying/structured_outputs/. Accessed: 2024-11-11.

[75] Meta AI. Introducing meta llama 3: The most capable openly available llm to date. https://ai.meta.com/blog/meta-llama-3/, April 2024. Accessed: 2024-11-07.

Appendix A

Appendices

A.1 Example Questions and Decompositions

Type 4
MHQ: `Which region does the Northern Motorway from the major city in the country where the initiative aiming to achieve a significant reduction in road deaths and serious injuries by promoting a 'Safe System' approach to road safety is implemented provide access to?`
Answer: Northland
Q1: `What initiative aims to achieve a significant reduction in road deaths and serious injuries by promoting a "Safe System" approach to road safety?`
A1: The Road Safety Strategy
Q2: `In which country is` **`The Road Safety Strategy`** `implemented to reduce road traffic injuries and fatalities?`
A2: New Zealand
Q3: `What is one of the major cities in` **`New Zealand`** `known for its extensive bus network?`
A3: Auckland
Q4: `What region does the Northern Motorway from` **`Auckland`** `provide access to?`
A4: Northland

Type 5
MHQ: `What organization did the operator of the rail network in the country known for its critical approach to transport demand forecasting, utilizing statistical models and socio-economic indicators to predict future travel patterns, and responsible for the activity referred to by the term describing regular efforts to keep roads safe and functional, including vegetation control, litter removal, and drainage maintenance, emerge from during its establishment in 2008?`
Answer: New Zealand Railways Corporation
Q1: `What term describes the regular activities performed to keep roads safe and functional, including vegetation control, litter removal, and drainage maintenance?`
A1: Routine Maintenance
Q2: `What country is known for its critical approach to transport demand forecasting, utilizing statistical models and socio-economic indicators to predict future travel patterns?`
A2: New Zealand
Q3: `Who operates the rail network in New Zealand and is responsible for Routine Maintenance?`
A3: KiwiRail
Q4: `What organization did KiwiRail emerge from during its establishment in 2008?`
A4: New Zealand Railways Corporation

Type 6
MHQ: `Who is responsible for enforcing the Health and Safety at Work Act in the country that has a comprehensive framework of transport safety regulations covering road, rail, maritime, and aviation, in the same year that the project in Auckland that transformed underutilized spaces into vibrant cycling and pedestrian pathways officially opened?`
Answer: WorkSafe New Zealand
Q1: `What project in Auckland has transformed underutilized spaces into vibrant cycling and pedestrian pathways?`
A1: Te Ara I Whiti Lightpath
Q2: `In what year did Te Ara I Whiti Lightpathofficially open?`
A2: 2015
Q3: `What country has a comprehensive framework of transport safety regulations covering road, rail, maritime, and aviation?`
A3: New Zealand
Q4: `Who is responsible for enforcing the Health and Safety at Work Act 2015 in New Zealand?`
A4: WorkSafe New Zealand

A.2 Statistical Significance Results for OIA and SP-CoT Datasets

This section presents the statistical significance results for the OIA and SP-CoT datasets. The table below summarizes the p-values for differences in metrics between the sorted and unsorted models for different values of d. The significance is determined based on a p-value threshold of 0.05.

Table A.1: Significance Test Results for OIA and SP-CoT Datasets

d	Metric	OIA p-value	OIA Significant	SP-CoT p-value	SP-CoT Significant
1	BLEU	8.96×10^{-15}	TRUE	1.0	FALSE
2	BLEU	2.41×10^{-5}	TRUE	0.7011	FALSE
3	BLEU	2.65×10^{-5}	TRUE	0.5847	FALSE
4	BLEU	1.74×10^{-6}	TRUE	1.17×10^{-4}	TRUE
5	BLEU	1.64×10^{-7}	TRUE	8.79×10^{-4}	TRUE
6	BLEU	1.80×10^{-26}	TRUE	0.0070	TRUE
1	ROUGE-1	2.34×10^{-27}	TRUE	1.0	FALSE
2	ROUGE-1	4.68×10^{-5}	TRUE	0.7011	FALSE
3	ROUGE-1	7.09×10^{-6}	TRUE	0.5847	FALSE
4	ROUGE-1	2.58×10^{-9}	TRUE	0.0019	TRUE
5	ROUGE-1	1.74×10^{-6}	TRUE	0.0023	TRUE
6	ROUGE-1	2.57×10^{-35}	TRUE	0.0070	TRUE
1	ROUGE-2	4.18×10^{-20}	TRUE	1.0	FALSE
2	ROUGE-2	2.55×10^{-7}	TRUE	0.7782	FALSE
3	ROUGE-2	5.67×10^{-6}	TRUE	1.0	FALSE
4	ROUGE-2	4.04×10^{-7}	TRUE	0.0923	FALSE
5	ROUGE-2	7.06×10^{-5}	TRUE	0.3438	FALSE
6	ROUGE-2	2.82×10^{-32}	TRUE	0.3438	FALSE
1	ROUGE-L	9.15×10^{-26}	TRUE	1.0	FALSE
2	ROUGE-L	1.65×10^{-7}	TRUE	0.7011	FALSE
3	ROUGE-L	6.47×10^{-5}	TRUE	0.5847	FALSE
4	ROUGE-L	3.52×10^{-8}	TRUE	0.0019	TRUE
5	ROUGE-L	4.97×10^{-5}	TRUE	0.0023	TRUE
6	ROUGE-L	1.95×10^{-32}	TRUE	0.0070	TRUE
1	Cosine Similarity	4.99×10^{-5}	TRUE	0.6780	FALSE
2	Cosine Similarity	2.82×10^{-3}	TRUE	0.6029	FALSE
3	Cosine Similarity	1.74×10^{-6}	TRUE	0.0319	TRUE
4	Cosine Similarity	4.02×10^{-10}	TRUE	0.0006	TRUE
5	Cosine Similarity	8.32×10^{-7}	TRUE	0.2416	FALSE
6	Cosine Similarity	2.51×10^{-36}	TRUE	0.8151	FALSE

A.3 Extract Prompt Code

Listing A.1: Extract Prompt Code

```
extract_prompt = (
    "Extract up to {max_knowledge_triplets} knowledge triplets from the given
        text. "
    "Each triplet should be in the form of (head, relation, tail) with their
        respective types and properties.\n"
    "---------------------\n"
    "INITIAL ONTOLOGY:\n"
    "Entity Types: {allowed_entity_types}\n"
    "Entity Properties: {allowed_entity_properties}\n"
    "Relation Types: {allowed_relation_types}\n"
    "Relation Properties: {allowed_relation_properties}\n"
    "\n"
    "Use these types as a starting point, but introduce new types if necessary
        based on the context.\n"
    "\n"
    "GUIDELINES:\n"
    "- Output in JSON format: [{’head’: ’’, ’head_type’: ’’, ’head_props’:
        {...}, ’relation’: ’’, ’relation_props’: {...}, ’tail’: ’’, ’tail_type’:
        ’’, ’tail_props’: {...}}]\n"
    "- Use the most complete form for entities (e.g., ’New Zealand Transport
        Agency’ instead of abbreviations)\n"
    "For any reference to the New Zealand Transport Agency, including variations
        such as ’NZ Transport Agency Waka Kotahi’,   NZ   Transport Agency,
          National   Transport Agency,      Waka   Kotahi,    or any similar
        phrasing, create the entity node as ’New Zealand Transport Agency.’
        Ensure all variations of this entity are consistently mapped to ’New
        Zealand Transport Agency’ in the output.\n"
    "- Do not create entities like ’request’ or ’response’ as they appear
        frequently and do not contribute to meaningful graph structure.\n"
    "- Do not create entities that are numbers, prices, or non-essential terms
        (e.g., dates unless critical to the meaning).\n"
    "- Keep entities concise (3-5 words max).\n"
    "- Break down complex phrases into multiple triplets for better
        granularity.\n"
    "- Ensure the knowledge graph is coherent and easily understandable.\n"
    "- Ensure that all property values are primitive types (e.g., String,
        Integer, Float, Boolean) or arrays of these types. Do not use maps or
        other complex structures.\n"
    "---------------------\n"
    "EXAMPLE:\n"
    "Text: The New Zealand Transport Agency approved the funding for road
        maintenance on State Highway 1.\n"
    "Wellington City Council is collaborating with the New Zealand Transport
        Agency on improving road safety initiatives.\n"
    "Output:\n"
    "[{’head’: ’New Zealand Transport Agency’, ’head_type’: ’AGENCY’,
        ’head_props’: {’prop1’: ’val’, ...}, "
    "’relation’: ’APPROVES’, ’relation_props’: {’prop1’: ’val’, ...}, "
    "’tail’: ’funding for road maintenance’, ’tail_type’: ’FUNDING_DECISION’,
        ’tail_props’: {’prop1’: ’val’, ...}},\n"
    " {’head’: ’Wellington City Council’, ’head_type’: ’ORGANIZATION’,
        ’head_props’: {’prop1’: ’val’, ...}, "
    "’relation’: ’COLLABORATES_WITH’, ’relation_props’: {’prop1’: ’val’, ...}, "
    "’tail’: ’New Zealand Transport Agency’, ’tail_type’: ’AGENCY’,
        ’tail_props’: {’prop1’: ’val’, ...}},\n"
    " {’head’: ’road safety initiatives’, ’head_type’: ’PROJECT’, ’head_props’:
        {’prop1’: ’val’, ...}, "
    "’relation’: ’IMPROVES’, ’relation_props’: {’prop1’: ’val’, ...}, "
    "’tail’: ’State Highway 1’, ’tail_type’: ’INFRASTRUCTURE’, ’tail_props’:
        {’prop1’: ’val’, ...}}]\n"
    "---------------------\n"
    "Text: {text}\n"
    "Output:\n"
)
```

A.4 Synthesizer and Refinement Prompts for OIA and SPOT Datasets

A.4.1 OIA Dataset Prompts

Listing A.2: Contextual Answering Prompt for OIA Dataset

```
Context information is below.
---------------------
{context_str}
---------------------
Given the context information and not prior knowledge, answer the query.
Query: {query_str}
Answer:
```

Listing A.3: Answer Refinement Prompt for OIA Dataset

```
The original query is as follows: {query_str}
We have provided an existing answer: {existing_answer}
We have the opportunity to refine the existing answer (only if needed) with some
    more context below.
------------
{context_msg}
------------
Given the new context, refine the original answer to better answer the query.
If the context isn't useful, return the original answer.
Refined Answer:
```

A.4.2 SP-CoT Dataset Prompts

Listing A.4: Contextual Answering Prompt for SPOT Dataset

```
Context information is below.
---------------------
{context_str}
---------------------
Given the context information and not prior knowledge, answer the query.
Query: {query_str}
Respond as briefly as possible with one single word if possible unless the
    answer contains multiple words.
Do not provide an explanation, just the answer. If you do not find the answer,
    please just state 'answer not found yet'.
Answer:
```

Listing A.5: Answer Refinement Prompt for SPOT Dataset

```
The original query is as follows: {query_str}
We have provided an existing answer: {existing_answer}
We have the opportunity to correct the existing answer (only if needed) with
    some more context below.
------------
{context_msg}
------------
Given the new context, refine the original answer to better answer the query. If
    the context isn't useful, return the original answer.
If you decide to change the answer, it is because you have found significant
    evidence that it should be corrected.
Refined Answer:
```

A.5 Qualitative Model Comparison Prompt

Listing A.6: Model Comparison Prompt

```
The question is: {question}
The correct answer is: {correct_answer}
The sorted model's answer is: {sorted_answer}
The unsorted model's answer is: {unsorted_answer}

Instructions:
1. For both the sorted and unsorted model answers, assess each as one of the
    following: correct, partially correct, wrong, or did not find.
2. Based on the assessments, determine which model's answer is closer to the
    correct answer:
   - If one is correct and the other is partially correct or wrong, the correct
       one should be preferred.
   - If one is partially correct and the other is wrong, the partially correct
       one should be preferred.
   - If both are correct, decide which is preferred based on completeness,
       accuracy, or any additional information relative to the correct answer.
   - If both are wrong, "neither" should be preferred.
   - If both are identical in correctness (e.g., both correct, both partially
       correct, or both wrong with no preference), the result should be "tie."

Respond in the following format:
<preferred model>|<sorted correctness>|<unsorted correctness>

Definitions:
- <preferred model>: Choose one of the following: sorted, unsorted, tie, neither
- <sorted correctness> and <unsorted correctness>: Choose one of the following
    for each: correct, partially correct, wrong, did not find

Note:
- Ensure that the more correct answer (or the more complete/relevant answer) is
    always preferred, even if both are marked as correct.
- Provide only the specified response format with no additional text.
```

A.6 Retrieved Nodes Taggin Prompt

Listing A.7: Model Comparison Prompt

```
I am testing a knowledge graph retrieval system. The knowledge graph contains
    entities and relations extracted from New Zealand Transport Agency
    documents. I have retrieved some triplets from the graph, and I need your
    help to rank them.

Please do the following:

- Given the query and the list of triplets, return the rank of each triplet
    according to how relevant you think it is to answer the query.
- For each triplet, also indicate whether it is relevant to the query (1 for
    relevant, 0 for not relevant).
- Provide your answer in the format:

{{
  "rankings": [
    {{
      "triplet_id": "<triplet_id>",
      "rank": <rank>,
      "relevant": <1 or 0>
    }},
    ...
  ]
}}

- The 'triplet_id' corresponds to the unique identifier of the triplet.
- The 'rank' indicates the relevance ranking you assign (1 is most relevant).
- 'relevant' is 1 if the triplet is relevant, 0 if not.

For example:

- Query: "What is the speed limit in Auckland?"

- Triplets:

1: id: "Auckland-has speed limit-50 km/h", triplet: "Auckland -has speed limit->
    50 km/h"
2: id: "New Zealand-increased sales-Motorcycles", triplet: "New Zealand
    -increased sales-> Motorcycles"
3: id: "Wellington-has speed limit-40 km/h", triplet: "Wellington -has speed
    limit-> 40 km/h"
4: id: "Auckland driving rules-provides information-Speed limits", triplet:
    "Auckland driving rules -provides information-> Speed limits"

- Output:

{{
  "rankings": [
    {{
      "triplet_id": "Auckland-has speed limit-50 km/h",
      "rank": 1,
      "relevant": 1
    }},
    {{
      "triplet_id": "Auckland driving rules-provides information-Speed limits",
      "rank": 2,
      "relevant": 1
    }},
    {{
      "triplet_id": "Wellington-has speed limit-40 km/h",
      "rank": 3,
      "relevant": 0
    }},
    {{
      "triplet_id": "New Zealand-increased sales-Motorcycles",
      "rank": 4,
      "relevant": 0
    }}
  ]
}}

- **Important:** Do not include any additional text or explanations in your
    response. Only output the JSON object as specified.
- **Important:** Output a valid JSON object with keys and strings properly
    quoted (use double quotes).
- Make sure to include an entry for every triplet provided.
- Make sure the output has the same total number of entries as the number of
    triplets provided, this is crucial for evaluation.

Here is the query:

{query}

Here are the triplets:

{triplets}
```