Chapter 11

Evaluation

This chapter will evaluate the performance of our KGC pipeline through a number of different approaches.

All results, as well as the tools we created for evaluation, are available at https://gitlab.anu.edu.au/u1085404/anu-scholarly-kg/-/tree/master/llm-based/LLM-KGC-v0-Tests

11.1 Experiment Setup

To enable all different evaluations, the starting point is to use our KGC pipeline to build KGs. The test sets we have used to build KGs are as follows:

11.1.1 Dataset 1: Five Full Papers

This set consists of 4 full papers from ANU ASKG plus 1 full paper not in ANU ASKG.

The 4 papers from ANU ASKG are already semi-structured, i.e., consisting of a hierarchy of sections, paragraphs, and sentences, and they are in the TTL formats, which can be directly accepted by our KGC pipeline. The ones outside of ASKG are in raw PDF format. Therefore we performed a manual decomposition into a similar hierarchy structure as those in ASKG and stored the result in JSON, which can then also be accepted by our pipeline by skipping the initial TTL-to-JSON stage.

Specifically, the title of each paper is:

1. MEL: Metadata Extractor & Loader (Rodríguez Méndez et al., 2021)
2. Modeling Actuations in BCI-O: A Context-based Integration of SOSA and IoT-O (Rodríguez Méndez, 2018)

3. Building An Open Source Linux Computing System On RISC-V (Ince et al., 2019)
4. TNNT: The Named Entity Recognition Toolkit (Seneviratne et al., 2021)
5. A Pipeline For Analysing Grant Applications (Pan et al., 2022)

11.1.2 Dataset 2: SciERC

The dataset SciERC (Luan et al., 2018) contains the mappings between 500 semi-structured abstract-only academic papers and their corresponding KGs. We regard them as semi-structured because, as in most NLP datasets, the papers have been preprocessed and sentence-tokenised, stored in JSONs. Although the target KGs are supplied, as mentioned before, these KGs are considered the results of traditional KGC approaches with a fixed and predefined number of types and predicates. However, they are still valuable references during our manual evaluation in Section 11.5.

The dataset had already been split into the training set (350), test set (100), and development set (50) when we downloaded it. Therefore, we ran our pipeline on their given test set consisting of 100 abstract-only papers.

Although they are already semi-structured, a basic preprocessing was done to align their structure with those in ASKG so that they match the interface of our pipeline.

11.1.3 Platform

The platform to create KGs from the datasets and perform the later evaluation processes is a desktop with the following specifications.

- CPU: AMD Ryzen 5 5600X 6-Core Processor
- Memory: 16 GB
- GPU: GeForce RTX 3080 12GB
- OS: Pop! OS 22.04 LTS

11.1.4 LLMs

Three of the four LLMs used are the same as the development process. However, we also used GPT-4o-Mini as the decoder of our pipeline to create KGs in order to see the portability and compatibility of our pipeline with different LLMs. Although we have listed two encoders, they are not the variable in our testing as they are used in different stages for different purposes, as mentioned before. Only LLaMA and GPT are our variables and are switched during the testing.

Decoder 1:

- **Named**: `Meta-Llama-3-8B-Instruct.Q4_0.gguf`
- **Size**: 4.66GB

- **Context Limit**: 8192
- **Availability**: https://huggingface.co/QuantFactory/Meta-Llama-3-8B-Instruct-GGUF/tree/main

Decoder 2:

- **Named**: `gpt-4o-mini`
- **Size**: Unknown
- **Context Limit**: 128k
- **Availability**: https://platform.openai.com/docs/models/o1

Encoder 1:

- **Named**: `BAAI--bge-base-en-v1.5`
- **Size**: 438.9MB
- **Context Limit**: 512
- **Availability**: https://huggingface.co/BAAI/bge-base-en

Encoder 2:

- **Named**: `BAAI--bge-m3`
- **Size**: 2.29GB
- **Context Limit**: 8192
- **Availability**: https://huggingface.co/BAAI/bge-m3

11.2 General Evaluation

With 100 abstract-only papers plus five full papers with two different LLM decoders, we created 204 different KGs $[(100 + 5) * 2 = 210]$. The tables below show the general information about these KGs and the runtimes during the construction.

- **Pipeline Complexity**: Table 11.1
- **Five-Full-Papers + LLaMA**: Tables 11.2 and 11.3
- **Five-Full-Papers + GPT**: Table 11.4 and 11.5
- **SciERC + LLaMA**: Table 11.6 and 11.7
- **SciERC + GPT**: Table 11.8 and 11.9

The complexity table (Table 11.1) shows the theoretical complexity of each stage of the pipeline. They have been derived from the previous chapters (Chapters 6 to 10). The complexity is measured by the number of times when an LLM decoder is called. That is,

for a single LLM decoder call, it is $O(1)$. Since a single LLM decoder call is dominantly expensive, the complexity of all the other computations, e.g., the algorithmic processes, file IOs, and LLM decoder calls, can all be ignored. The meaning of each symbol in the complexity analysis is as follows:

- L: The length of the paper
- E: The number of entities
- M: The number of mentions
- R: The number of relations (triples between any two entities)

For the rest of the tables, the meaning of each column is as follows:

- **Length**: The length of the paper, measured in tokens.
- **Entities**: The total number of entities nodes in the output KG.
- **Mentions**: The total number of mentions nodes in the output KG. Each entity has at least one mention. Different mentions may correspond to the same entity.
- **Relations**: The number of relations (triples between any two entities) in the output KG.
- **Isolated Entities**: The number of entities that are not related to any other entities in the output KG. That is, there is no outward edge from these entity nodes except those pointing to their attributes, e.g., their labels, aliases, and descriptions. Commonly speaking, the fewer, the better.

The actual complexity of KGC using LLaMA is measured by time (minute). We have found in the Background Chapter (2.4.5) that the complexity of the LLM decoder is at least quadratic and potentially cubic. However, since each prompt our pipeline made is approximately the same length, and thus each LLM decoder call should take roughly the same constant amount of time, if we treat the time taken by each LLM decoder call as $O(1)$, the overall runtime should align with the theoretical complexity shown in Table 11.1.

The actual complexity of KGC using GPT is measured by cost (USD). Since each LLM call is done through the OpenAI API, and the network speed and the server reaction time can be very unstable, we did not use time as our measurement. However, we found that the cost for each LLM call through the OpenAI API is about linearly proportional to the length of each prompt. Since each prompt our pipeline made is approximately the same length, and thus each prompt should have a similar cost, if we treat the cost by each LLM decoder call as $O(1)$, the overall cost should also align with the theoretical complexity shown in Table 11.1.

Table 11.1: **Complexity** of the Pipeline

Stage	Complexity
0	$O(0)$
1	$O(9L)$
2	$O(2M + M^2)$
3	$O(3L + R_1)$
4	$O(L + Max(20^2, (0.1 \cdot E)^2) + R_2)$
5	$O(E + R)$
6	$O(0)$

Table 11.2: **Statistics** of the KGs constructed from the **Five-Full-Papers** dataset by **LLaMA**

Paper	Length (Token)	Entities	Mentions	Relations	Isolated Entities
1	1165	147	243	235	38
2	3161	351	747	860	73
3	3062	473	753	894	106
4	1486	201	339	518	41
5	5183	653	1329	1437	113
Mean/Std	2811 ± 1434	356 ± 184	682 ± 384	789 ± 404	74 ± 31

Table 11.3: **Runtimes** (in Minutes) of KGC from the **Five-Full-Papers** dataset by **LLaMA**

Paper	S0	S1	S2	S3	S4	S5	S6	Time
1	1.466E-6	6.336	20.18	3.461	12.58	4.775	2.078E-3	47.34
2	2.587E-6	18.75	117.7	9.636	38.59	13.28	3.851E-3	198.0
3	5.126E-6	20.36	79.03	11.47	24.09	12.65	4.214E-3	147.6
4	3.719E-6	8.152	42.93	4.585	29.84	7.782	2.089E-3	93.29
5	3.703E-6	29.72	354.6	16.22	37.43	16.79	6.411E-3	454.8
Mean/Std	3.320E-6 ± 1.228E-6	16.66 ± 8.574	122.9 ± 120.5	9.074 ± 4.665	28.51 ± 9.551	11.06 ± 4.256	3.729E-3 ± 1.604E-3	188.2 ± 142.6

Table 11.4: **Statistics** of the KGs constructed from the **Five-Full-Papers** dataset by **GPT**

Paper Index	Length (Token)	Entities	Mentions	Relations	Isolated Entities
1	1165	180	301	526	14
2	3161	313	743	1130	24
3	3062	473	803	1313	45
4	1486	209	395	837	18
5	5183	638	1444	1954	47
Mean/Std	2811 ± 1434	363 ± 172	737 ± 403	1152 ± 482	30 ± 14

Table 11.5: **Costs** (in USD) of KGC from the **Five-Full-Papers** dataset by **LLaMA**

Paper	Cost (USD)
1	0.21
2	0.62
3	0.53
4	0.32
5	1.45
Mean/Std	0.63 ± 0.44

Table 11.6: **Statistics** of the KGs constructed from the **SciERC** dataset by **LLaMA**

Paper	Length (Token)	Entities	Mentions	Relations	Isolated Entities
Mean/Std	134 ± 51	29 ± 10	37 ± 13	19 ± 11	10 ± 5

Table 11.7: **Runtimes** (in Minutes) of KGC from the **SciERC** dataset by **LLaMA**

Paper	S0	S1	S2	S3	S4	S5	S6	Time
Mean/Std	9.119 E-6 ± 1.731 E-6	9.989 E-1 ± 2.568 E-1	2.110 ± 1.078	5.834 E-1 ± 2.014 E-1	2.989 E-1 ± 2.125 E-1	9.623 E-1 ± 3.099 E-1	2.059 E-4 ± 1.346 E-4	4.954 ± 1.940

Table 11.8: **Statistics** of the KGs constructed from the **SciERC** dataset by **GPT**

Paper Index	Length (Token)	Entities	Mentions	Relations	Isolated Entities
Mean/Std	134 ± 51	28 ± 10	37 ± 15	33 ± 18	4 ± 2

Table 11.9: **Costs** (in USD) of KGC from the **SciERC** dataset by **GPT**

Paper	Cost (USD)
Mean	0.02

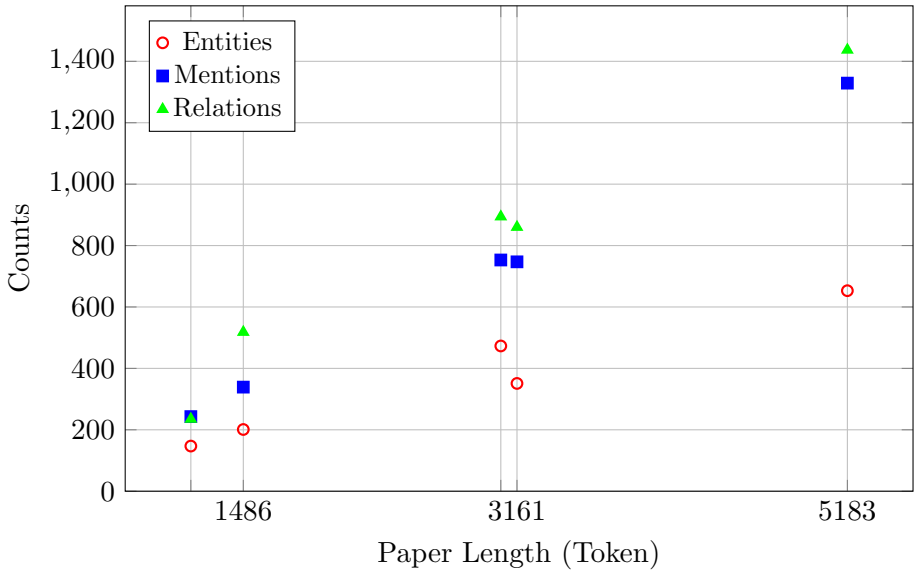

Figure 11.1: **Statistics** of the KG constructed from the **Five-Full-Paper** dataset by **LLaMA** vs. Paper Length

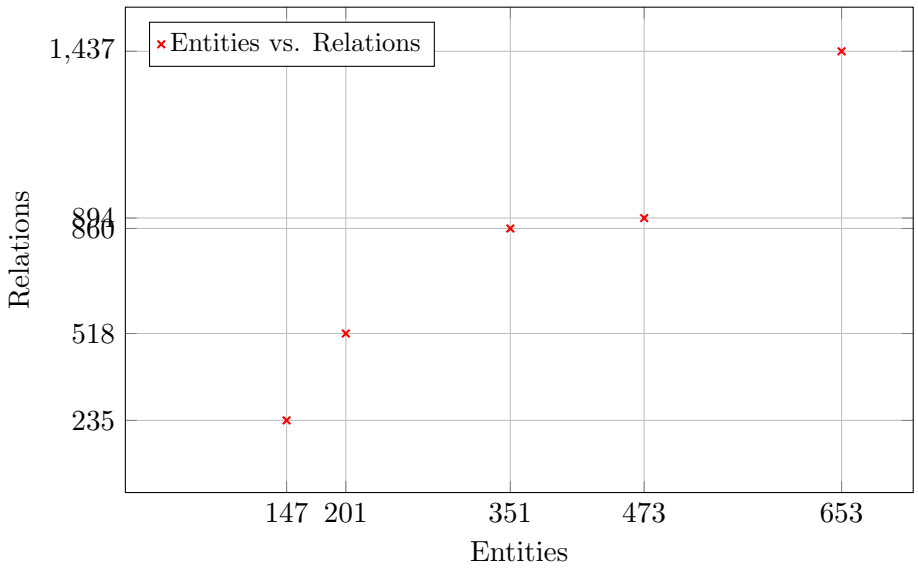

Figure 11.2: **Entities vs Relations** of the KG constructed from the **Five-Full-Paper** dataset by **LLaMA**

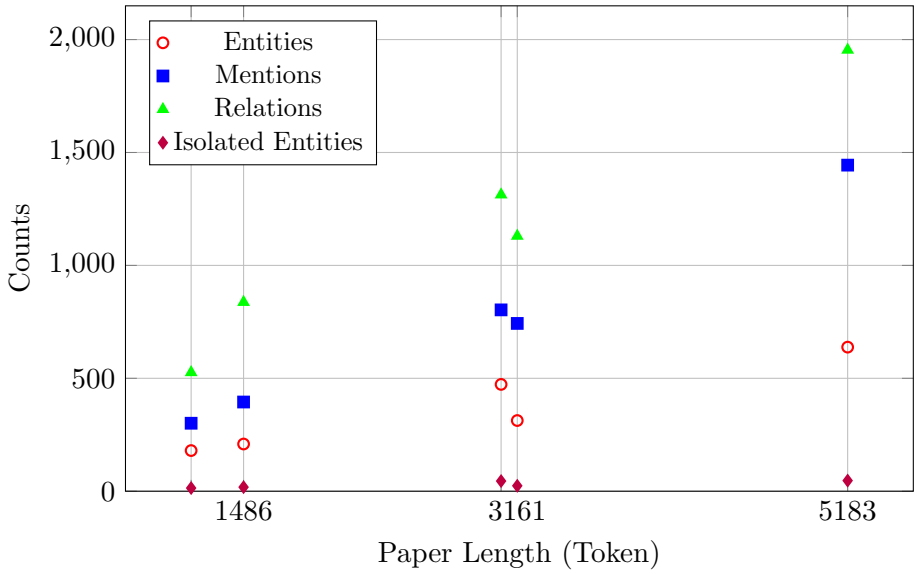

Figure 11.3: **Statistics** of the KG constructed from the **Five-Full-Paper** dataset by **GPT** vs. Paper Length

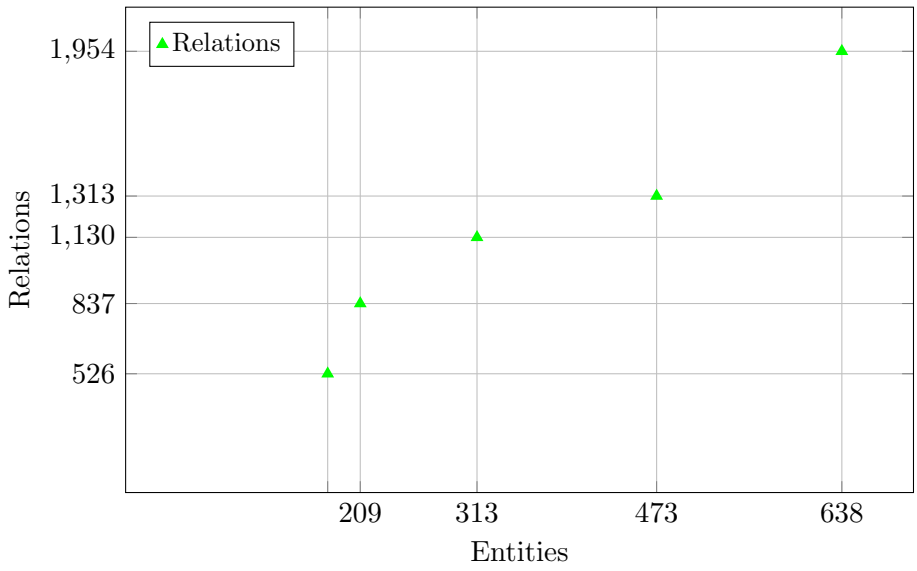

Figure 11.4: **Entities vs Relations** of the KG constructed from the **Five-Full-Paper** dataset by **GPT**

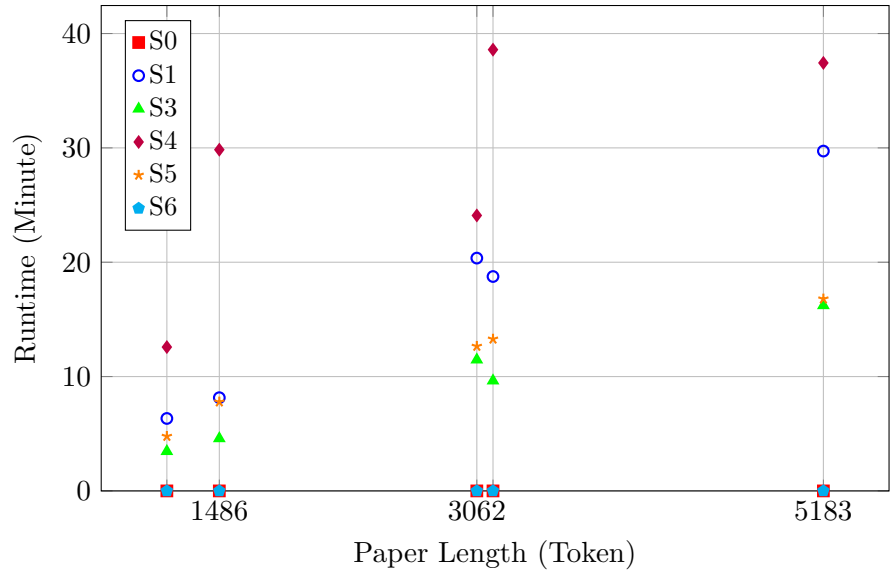

Figure 11.5: **Runtimes** of Stages **S0, S1, S3, S4, S5, and S6** of KGC from the **Five-Full-Paper** dataset by **LLaMA** vs. Paper Length

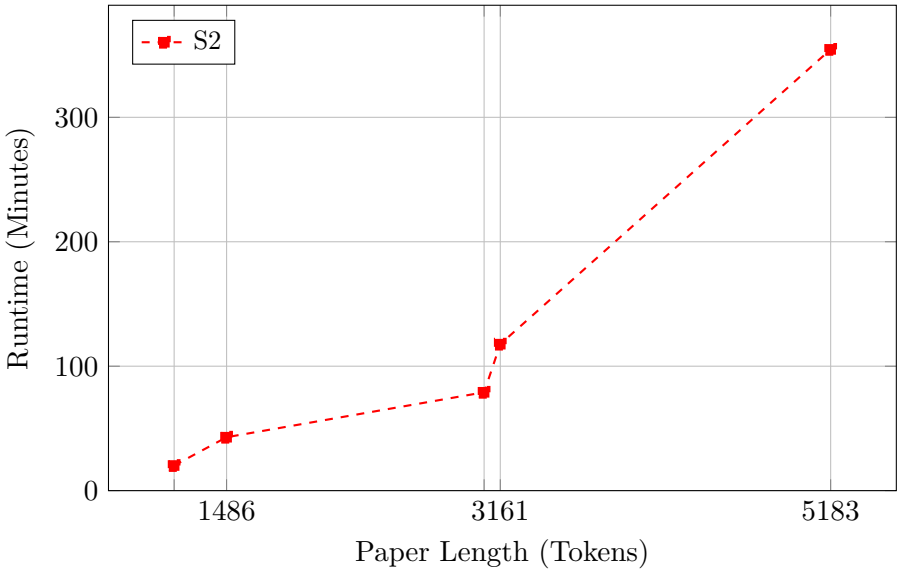

Figure 11.6: **Runtimes** of **S2** of KGC from the **Five-Full-Paper** dataset by **LLaMA** vs. Paper Length

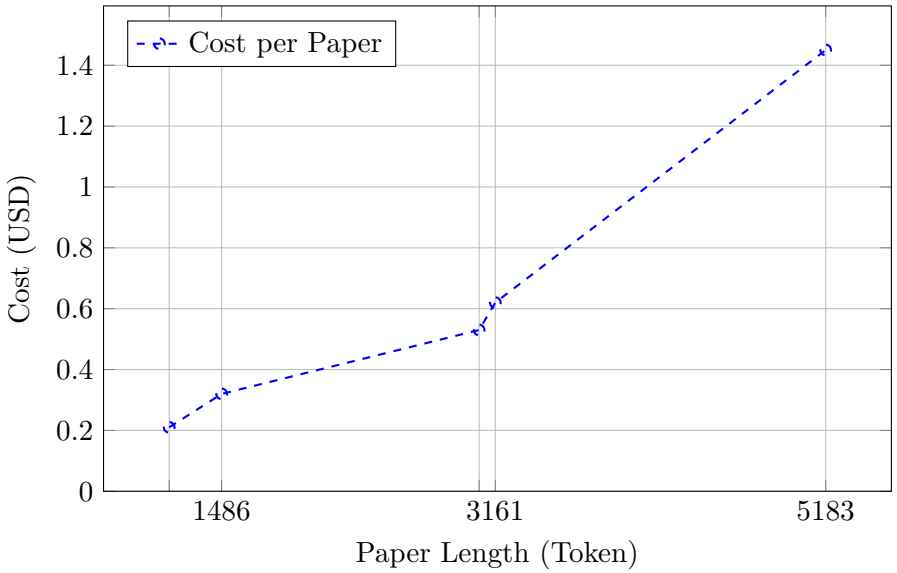

Figure 11.7: **Costs** of KGC from the **Five-Full-Papers** dataset by **GPT** vs. Paper Length

11.2.1 Discussion on the KGs

From the Figures 11.1, 11.2, 11.3, and 11.4 and Tables 11.2, 11.4, 11.6 and 11.8, some key observations can be drawn.

1. For both datasets and for both LLM decoder models (Tables 11.2, 11.4, 11.6, and 11.8), the pipeline is *capable* of extracting some entities and relations, although we cannot measure the quality of these entities and relations at the moment.

    a) The pipeline based on GPT seems to be *more capable* of finding relations between entities as the resultant KGs have statistically lower * isolated entities. This makes sense because the specific version of LLaMA we used is considered lightweight compared to GPT running on servers. Similarily for the SciERC dataset shown in Tables 11.6 and 11.8.

2. The number of Mentions is approximately *linearly proportional* to the length of the paper. The number of Entities is slightly *sub-linearly proportional* to the length of the paper. This can be shown in Figures 11.1 and 11.3 for both KGs constructed by LLaMA and GPT. The number of relations is *linearly proportional* to the number of entities, as shown in Figures 11.2 and 11.4.

    a) The *linear relation* between mention count and paper length is as expected because the longer the paper is, the more words or phrases it should have.

    b) Commonly speaking, the expected relation between entity count and paper length is indeed *sub-linear*. This is because most papers should have a central theme rather than keep introducing new things (entities). Although the longer the paper is, the more mentions of these things are, the number of things should not grow as fast as their mentions. Therefore, this is as expected. However, since the relation is only slightly sub-linear, this suggests that the Coreference Resolution stage (Stage 3) may *not be performed thoroughly*, i.e., some entity mentions (that should be merged into the same entity as they refer to the same thing) are treated as two separate entities.

    c) In a general sense, the expected number of Relations should grow *more than linearly proportionally* to the Entities. This is because a graph with more nodes can potentially have quadratically more edges. However, based on Figures 11.2 and 11.4, the number of relations seems to be linearly proportional to the number of entities. This suggests that there are potentially some relations *missed by the pipeline*.

* The Isolated Entities column of Table 11.2 [38, 73, 106, 41, 113] and the Isolated Entities column of Table 11.4 [14, 24, 45, 18, 47] have p-value 0.032

11.2.2 Discussion on the Complexity

From the Figures 11.5, 11.6 and 11.6 derived from Tables 11.2, 11.3, 11.4, 11.5, key observations can be drawn.

1. The runtimes of Stage 0 (KG Preprocessing) and Stage 6 (KG Postprocessing) are *approximately zero*, regardless of the length of the paper, as shown in Figure 11.5. This aligns with the $O(0)$ theoretical complexity of the pipeline as they do not involve LLM decoder calls.

2. The runtimes of Stage 1 (Entity Mention Extraction) and 3 (Local Relation Extraction) are approximately *linearly proportional* to the length of the paper. This also aligns with the theoretical results, $O(3L)$ and $O(L + R)$.

3. The runtimes of Stage 4 (Global Relation Extraction) and the length of the paper are *not quite correlated*. However, it is bounded by a constant, which is as expected because we have set the maximum of entities to compare with each other to 20 to prevent $0.1N$ from growing quadratically.

4. The runtime of Stage 5 (Schema Generation) tends to be *sub-linearly* related to the length of the paper. This is as expected because, as discussed previously (11.2.1), the measured relation between Entities and paper length is sub-linear, and the measured relationship between Relations and Entities is linear. Since the theoretical complexity of Stage 5 is $O(E + R)$ and both E and R are sub-linearly proportional to L, the measured runtime is, therefore, sub-linear.

5. The relationship between the runtime of Stage 2 (Coreference Resolution and Entity Disambiguation) is clearly more than linear, as shown in Figure 11.6. This aligns with the $O(2M + M^2)$ complexity, and since M is linearly proportional to L. In Stage 2, our pipeline filters out most of the unrelated mentions based on their embedding and only calls the LLM decoder whenever they are highly similar. Even with blocking, the nature of this substage is still a pairwise comparison, which can make the complexity of this stage scale very quickly in the worst case. Table 11.3 shows that the runtimes of all papers are dominant by this single stage, leading to non-linear growth of runtime (Table 11.3) or cost (Figure 11.7) with respect to the length of the paper. Therefore, it is worth checking if all components in Stage 4 are essential and if there are any possibilities to simplify Stage 4 and increase the pipeline performance without losing too much quality of the constructed KG. One solution is that whenever two entity mentions are similar in their embedding, the pipeline directly merges them into the same entities without double-checking with the LLM decoder. In this case, the expected complexity can be reduced to $O(2M)$. This led to an ablation study in Section 11.6.

11.3 Evaluation via Reverse Engineering

The last part of the evaluation mainly focuses on the shallow features of the KGs. They are important but not sufficient. Starting from this section, the evaluation will be directly based on the quality of the constructed KGs.

The main challenge of evaluating the quality of the KGs is that there are no references,

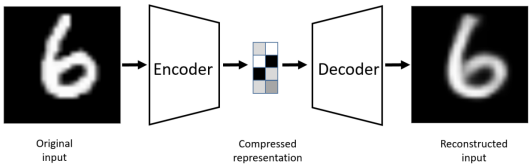

Figure 11.8: Structure of an Auto-encoder (Bank et al., 2020)

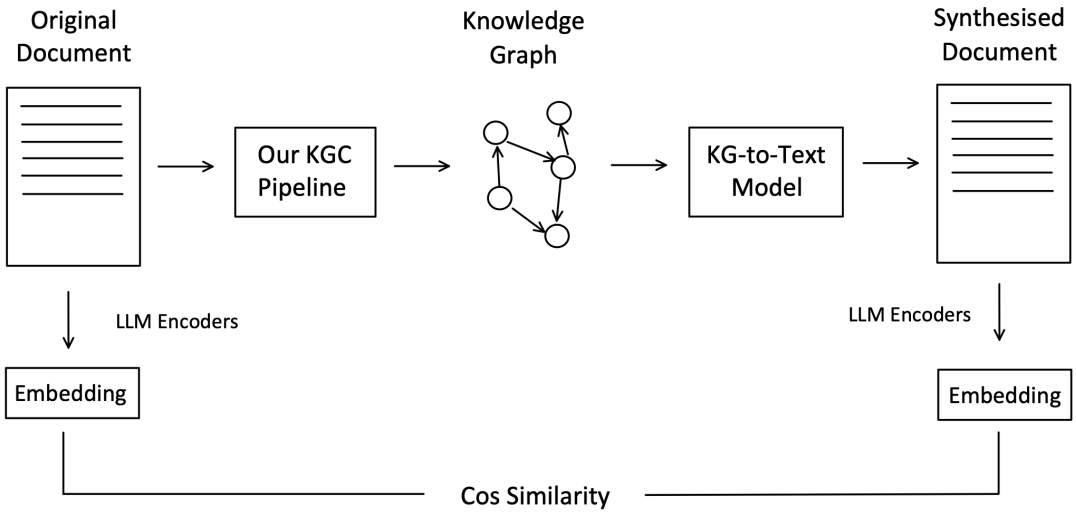

Figure 11.9: Evaluation via Reverse Engineering

i.e., the datasets are unlabelled. Even though SciERC is labelled, it does not contain full papers and targeted KGs in the dataset are considered KGs constructed through traditional KGC.

The first evaluation approach is inspired by how auto-encoder models are trained and tested.

In general, an autoencoder takes a source as input and converts it into some compressed representation through its encoder, also referred to as the feature extractor. After that, it tries to reverse the compressed representation back to the original source. By comparing the original source and the synthesised source, we can then know how good the feature extractors and decoder are. This is because a good auto-encoder should capture most of the key features of the input source so that it is possible to recover the source back from the key features with minimal loss. (Bank et al., 2020)

Similarly, our KGC pipeline can be viewed as the encoder part of an autoencoder, whereas the compressed representation is KG. Therefore, if we have a model to convert KGs back to text documents, we can then evaluate our pipeline and KG-to-Text model together based on the similarity between the original documents and the synthesised

documents. A common way to measure the similarity between two documents is by comparing the cosine similarity of their embeddings, as embeddings also represent the key features of the two documents.

In fact, KG-to-Text is another important field of study that is parallel with KGC in the broader field of KG (Pan et al., 2023). Because the study of KG-to-Text can be significantly deep, we only implemented a straightforward approach to convert our KGs back to text documents.

11.3.1 Evaluation Approach

This approach is also considered the baseline approach in many KG-to-Text studies, as illustrated in Pan et al. (2023)'s survey article. That is, we use an LLM decoder to concat all (Subject, Predicate, Object) triples of Entities in the constructed KG.

However, as a minor extension to the baseline, we process the triples in batches, and the most relevant triples are processed first, which ensures the most important knowledge of the document has been conveyed at the beginning, which aligns with the nature of academic writing. When concatenating triples, we also supply the LLM with the description of entities involved in the triples. The relevance of triple is calculated by the sum of the relevance score of its subject and object entities. The relevance scores of entities are calculated by the cosine similarity between the embedding of the entities and documents, which is already done in Stage 8: Global Relation Extraction, discussed in Chapter 9.

The algorithm for the KG-to-Text module and the Evaluation via the Reverse Engineering module are as follows:

KG-to-Text Algorithm

```
Input: KG
Output: Text Document

For each triple in the KG,

    calculate its relevance score based on the sum of the relevance
    scores of its subject entity and its object entity.

Sort the triples through their relevance scores.

For each batch of 20 triples in the KG,

    get all entities involved in these triples.

    [LLaMA Prompt_1] Given a list of (subject, predicate, object)
    triples in a graph, given the label, aliases, and description of all
    the entities involved in these triples, prompt the LLM to generate a
```

```
    paragraph describing these triples.

    Add the paragraph to the output document.
```

Prompt 1 (Simplified)

```
## Task Definition

You are a linguistic expert involved in the task of knowledge-graph-to-
text generation. Given a list of (subject, predicate, object) triples in
a graph, given the label, aliases, and description of all the entities
involved in these triples, your task is to generate a fluent paragraph
describing these triples.

You may use the information of entities for reference. However, the focus
of your generated paragraph should be the triples, i.e., the relations
between entities, rather than solely the label, aliases and description
of each entity.

## Output Format

...

## Entities

...

## Relations (Triples)

...

## Output

...
```

Algorithm for Evaluation via Reverse Engineering

```
Input: KG (including the input semi-structured document)
Output: Score

Flatten the semi-structured paper stored in the KG into plain text.

[M3] Embed the original paper.

Get the synthesised paper from the KG-to-Text model given the KG.

[M3] Embed the synthesised paper.
```

```
Score = The cosine similarity between the original paper and the
synthesised paper.
```

The reason we supply the description and aliases of the entities involved in a batch of triples is that the LLM may not know what the entities are since academic papers usually introduce new terms. With the supplement of the entity description, which is part of the output KG of our pipeline, we expect that the LLM can generate text from the triple better. However, we also insert an instruction to prevent the LLM from focusing too much on the description of each entity rather than the triples. This is because the sets of entities involved in each batch of triples are not disjoint. If the synthesised paragraph consists of too much information about the descriptions of each involved entity, the whole synthesised paper will contain a lot of repetitive text.

During the graph-to-text process, no matter which LLM decoder is used to construct the original KGs, the LLM decoder used to generate texts from KGs is always LLaMA (Meta-Llama-3-8B-Instruct-Q4-0), and the LLM encoder used is always BAAI–bge-m3 because of its larger context limit.

Figure 11.10 shows part of the text synthesised from the KG constructed from one of the documents in the Five-Full-Paper dataset for demonstration purposes.

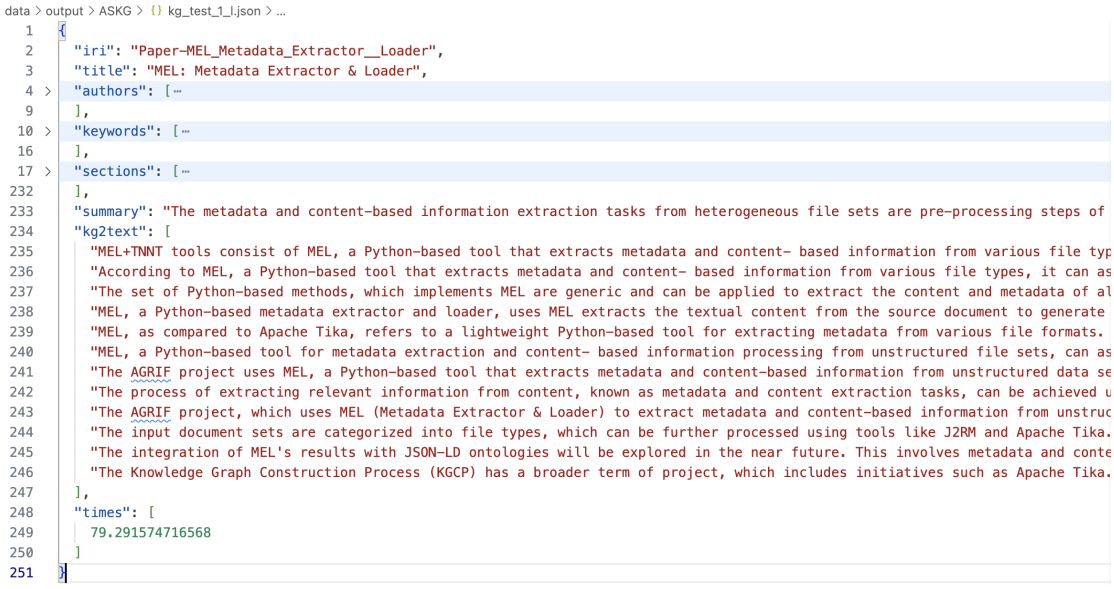

Figure 11.10: Synthesised Text Example

Table 11.10: **Similarity Scores** of the text documents synthesised from the KGs constructed from the **Five-Full-Papers** dataset by **LLaMA**

Paper	Length (Token)	Similarity Score
1	1165	0.9395
2	3161	0.8989
3	3062	0.8916
4	1486	0.9224
5	5183	0.8330
Mean/Std	2811 ± 1434	0.8971 ± 0.03626

Table 11.11: **Similarity Scores** of the text documents synthesised from the KGs constructed from the **Five-Full-Papers** dataset by **GPT**

Paper	Length (Token)	Similarity Score
1	1165	0.9155
2	3161	0.9165
3	3062	0.8633
4	1486	0.9302
5	5183	0.8726
Mean/Std	2811 ± 1434	0.8996 ± 0.02655

11.3.2 Evaluation Results

Tables 11.10, 11.11, and 11.12 collectively demonstrate relatively high similarity scores between the original documents and the synthesised documents from the KGs, across all datasets and LLM decoders used. It seems that the similarity score is not quite correlated with the length of the papers in general, although the longest paper (Paper 5) has relatively low similarity scores for both LLM decoders.

The similarity score for documents synthesised from the KGs constructed by GPT is also similar to those by LLaMA.

Although around 0.9 is considered a high similarity score. There is still room for improvement to ensure a more lossless conversion from text documents to KGs.

11.3.3 Remarks

This approach can reflect both the precision and recall of a constructed KG. This is because if the constructed KG does not capture enough information from the original document or the information captured is incorrect, the synthesised document based on the constructed KG cannot be highly similar to the original document.

However, this approach tests both the KGC pipeline and the KG-to-Text model together. It cannot reflect on the performance of our KGC pipeline alone. Since we only

Table 11.12: **Similarity Scores** of the text documents synthesised from the KGs constructed from the **SciERC** dataset by **LLaMA** and **GPT**

Paper	Length (Token)	LLaMA	GPT
Mean/Std	134 ± 51	0.8961 ± 0.02930	0.9041 ± 0.04099

implemented a simple KG-to-Text model for our evaluation purpose, the conversion from KGs to texts can also be lossy. The performance of the KGC pipeline can potentially be underestimated.

11.4 Evaluation via Application

The second valuation approach is inspired by the ways how (Edge et al., 2024) test their constructed KGs. Essentially, this approach comes from one of the largest applications of KGs, Retrieval-Augumented Generation (RAG) (Pan et al., 2023).

In general, in an RAG system, there is a chatbot, usually implemented by an LLM decoder, that is responsible for answering users' questions. However, instead of directly answering users' questions based on the common knowledge of the chatbot, the RAG system first searches through a knowledge base to find relevant sources. After the retrieval, the relevant sources serving as hints, together with the original user question, are inputted to the chatbot. With the supplied resources, the chatbot can then answer questions more accurately. (Pan et al., 2023)

Traditionally, these sources are in the form of text documents (Edge et al., 2024). However, KG can also serve as the knowledge base for retrieval. It is expected to be more effective since KGs are structured (Edge et al., 2024).

Therefore, by evaluating the RAG system that is based on our constructed KGs, we can then indirectly evaluate our KGs and, hence, our KGC pipeline.

However, the drawback is that we have to create an RAG system on top of our KGC pipeline. In fact, graph-based RAG is another field of study that is parallel to KGC in the topic of KG (Pan et al., 2023). Therefore, creating an advanced Graph-Based RAG system requires additional in-depth study.

Another issue is that neither the Five-Full-Paper dataset nor SciERC have any question-answer pairs to serve as the ground truth.

11.4.1 Evaluation Approach

The solution is to create a simple RAG system for our evaluation. To obtain question-answer pairs, we used one of the most SOTA LLM, gpt-o1-preview, which is considered more advanced but expensive than our currently using Meta-Llama-3-8B-Instruct-Q4-0 and gpt-4o-mini (OpenAI, 2024b). Specifically, we first let the GPT-O1 read the

whole document for all papers in the Five-Full-Paper dataset. This is practical because GPT-O1 has a wide input limit of 128K tokens. Then, we ask GPT-O1 to generate 10 question-answer pairs. These question-answer pairs are then served as the inputs and targeted outputs for the evaluation of our RAG system.

These question-answer pairs are expected to be highly accurate because they are generated by the LLM through reading the whole original document.

On the RAG side, the system can only use the constructed KG to retrieve relevant facts for the chatbot to answer questions. We also limit the number of hops the system can query the KG from an entity node relevant to the question. This is because it is somewhat impractical to use the whole KG as supplementary resources which would undermine the RAG approach of retrieving only the most relevant facts.

Since papers in SciERC are too short, this evaluation appraoch only underke on the Five-Full-Paper dataset.

The prompt used for creating the question-answer pairs is as follows:

Prompt for Generating Question-Answer Pairs (Simplified)

```
## Task

You are a university teacher who has recently assigned your students to
read an academic paper. To test their understanding, you will prepare 10
questions that are only answerable by referring to the paper. Your
questions should be objective, meaning each question should potentially
have a clear, concrete answer without the need for subjective
interpretation. You should also provide a sample answer for each question.

## Output Format

...

## Paper

...

## Your Response

...
```

We created the scenario that a university teacher is creating questions to test their student about the understanding of the paper because reading papers is a common activity for both teachers and students in universities.

We included the instruction that "questions should be only answerable by referring to the paper" to avoid our chatbot simply using its common knowledge to answer the question. If such a case happens, we then cannot effectively evaluate our KGs.

We included the instruction that "questions should be objective" to prevent the LLM from generating questions that require the opinions of the chatbot instead of solely based on the content of the paper.

Next, we developed a simple RAG system that is capable of answering questions regarding the papers.

Specifically, we have implemented both the text-based RAG and graph-based RAG for head-to-head comparison. Their algorithms are as follows:

Text-Based RAG

```
Input: A question regarding the paper
       A semi-Structured Paper
                        (This is the same as the input of our pipeline)

Output: An answer

[BGE] Embed all sentences in the input paper.

[BGE] Embed the question.

Retrieve the 10 most relevant sentences from the input paper based on the
similarities between the embeddings of sentences and the question.

[LLaMA Prompt_1] Given the question and 10 most relevant sentences in
the paper, ask the LLM decoder to generate an answer.
```

Graph-Based RAG

```
Input: A question
       The KG of a paper
                        (This is the same as the output of our pipeline)

Output: An answer

[M3] Embed all entities in the KG based on their labels.

[LLaMA Prompt_2] Extract all keywords from the questions.

Embed all keywords.

For each keyword,
    find the closest entity in the KG.
    Store in relevent_entities.

For each relevant entity in relevant_entities,
    find all one-hop triples that involve this entity.
```

```
    Store in relevant_triples.

For each relevant triple in relevant_triples,
    if the subject or the object involved is not in relevant_entity,
        add it into relevant_entity.

[LLaMA Prompt_3] Given a list of relevant triples, the label, aliases,
and description of the relevant entities, and the original question, ask
the LLM to answer it.
```

Prompt 3 (Simplified)

```
## Task

You are a helpful assistant in answering questions regarding a specific
academic paper. Given a question about an academic paper and some
recourses extracted from the knowledge graph of the paper, your task is
to try to answer the question. The recourses you are given are a list of
relevant entities and a list of (subject, predicate, object) triples
about their relations.

## Output Format

...

## Question

...

## Resources (Entities)

...

## Resources (Triples)

...

## Answer

...
```

We also supply the description of entities when supplying triples for a similar reason to the prompt used for KG-to-Text in Section 11.3. The entity's label alone can be inefficient and potentially ambiguous.

The text-based RAG here is highly important because it essentially tests how the current stage of ASKG, without the deep structured knowledge about the content of the paper,

can be used to perform RAG. If RAG is the main application of ASKG, this may lead to a discussion about whether it is necessary to put so much effort into creating KGs or we may maintain the current stage of ASKC, i.e. in the form of a metadata graph instead of KG of the content of papers.

Last, we check the similarity between the generated answers from our RAG system with the ground truth answers.

Overall, the diagram for our Evaluation via Application is shown in Figure 11.11. This diagram essentially shows the general picture of one of the potential applications of KGs.

Note that we have used GPT-O1-Preview to directly read and create questions and answers for a document. This is only for evaluation purposes to generate question-answer pairs and is not quite a practical approach to querying a document in real life. This is because a RAG system in real life usually searches through multiple documents instead of just one. In this case, it can be impractical to feed all documents into the LLMs to directly generate answers. Instead, we should rely on KGs or Metadata Graphs to effectively retrieve only the key information. This is why creating KGs is important even though the current state of LLMs can directly answer the question by looking at the whole single paper.

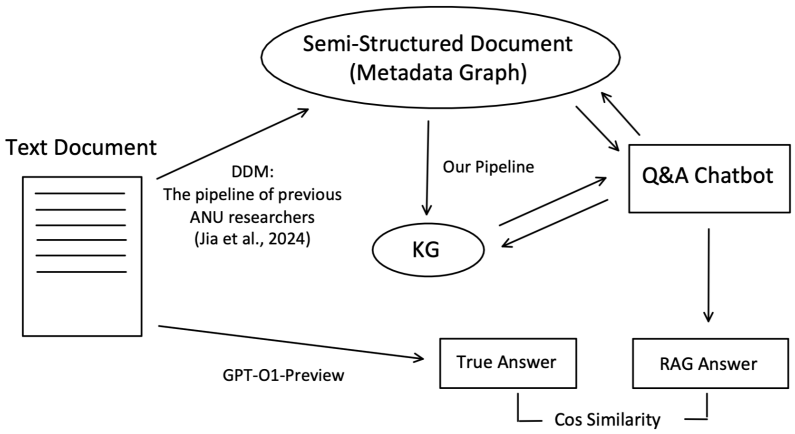

Figure 11.11: The process of evaluation via application

Figure 11.12 shows a sample list of tuples of question, target answer, RAG answer, and their similarity scores for demonstration purposes.

11.4.2 Evaluation Results

Figure 11.13 and Table 11.13 show the comparison of the similarity score between the target answers and the answer generated by the RAG system using the three different knowledge bases.

```
[
  {
    "question": "What is the main purpose of the MEL tool introduced in the paper?",
    "answer": "The main purpose of MEL is to implement a set of methods to extract metadata and content-based information from unst
    "LLM_answer": "The main purpose of the MEL tool introduced in the paper is to extract metadata and content-based information fr
    "similarity_score": 0.93798828125
  },
  {
    "question": "How does MEL integrate with TNNT, and what is the benefit of this integration?",
    "answer": "MEL is integrated with The NLP-NER Toolkit (TNNT) to automate the extraction of categorized named entities from the
    "LLM_answer": "MEL integrates with TNNT by combining MEL's capabilities for extracting metadata and content-based information f
    "similarity_score": 0.90478515625
  },
  {
    "question": "What are the main differences between MEL and Apache Tika?",
    "answer": "While both MEL and Apache Tika aim to extract metadata and content from various file formats, the main differences a
    "LLM_answer": "The main differences between MEL and Apache Tika are as follows: MEL is a Python-based tool specifically designe
    "similarity_score": 0.9609375
  },
  {
    "question": "How many different file types does MEL support, and what is the significance of this feature?",
    "answer": "MEL supports more than 20 different file types. This comprehensive support makes it a versatile tool capable of proc
    "LLM_answer": "MEL supports more than 20 different file types. This feature is significant because it allows the tool to extrac
    "similarity_score": 0.95849609375
  },
  {
    "question": "What are the four major contributions of MEL as highlighted in the paper?",
    "answer": "The four major contributions of MEL are: (1) the ability to extract metadata sets and content-based information from
    "LLM_answer": "The four major contributions of MEL highlighted in the paper are: 1) Comprehensive metadata extraction support f
    "similarity_score": 0.9462890625
  },
```


Figure 11.12: A list of tuples of question, target answer, RAG answer, and similarity score

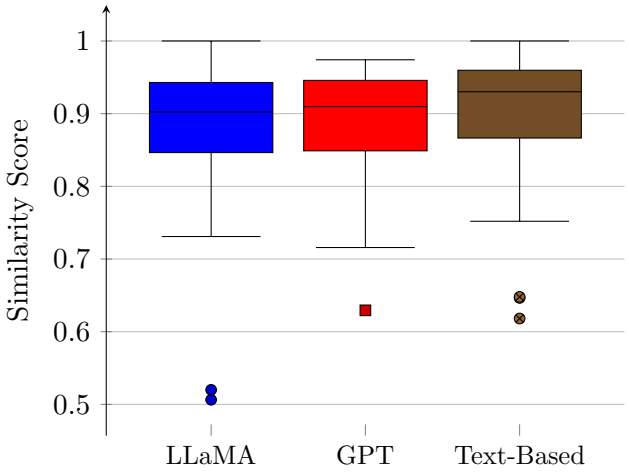

Figure 11.13: Comparison of the **Similarity Scores** between the target answers and the answer generated by the RAG system using the KG constructed by LLaMA (Graph-Base RAG), the KG constructed by GPT (Graph-Based RAG), and the original semi-structured document (Text-Base RAG)

The score of the RAG system based on the original semi-structured paper seems to be higher although it is not statistically higher.

Table 11.13: Comparison of the **Similarity Scores** between the target answers and the answer generated by the RAG system using different knowledge bases

RAG Method	Using KG by LLaMA	Using KG by GPT	Using the Original Semi-Structured Paper
Mean/Std	0.8842 ± 0.09523	0.8884 ± 0.07622	0.9018 ± 0.08873

However, with limited testing, we cannot say that one is better than others because there are many factors.

- RAG in real life is usually based on multiple documents; the text-based RAG may not scale well compared to Graph-Based RAG because, with proper Knowledge Graph Alignment, the same entities of different local KGs of papers should be linked to each other. Therefore, the keywords of the question can actually match the entities and start from the entities, many facts about the entities coming from different documents can be retrieved. In traditional RAG, there may not be such as global view of an entity.
- Both the Graph-Base RAG and Text-Base RAG are considered the baseline approach. Thus, they do not reflect on the true potentiality of Graph-Base RAG and Text-Base RAG.

Even though, it is still worthy to have such side by side comparsion.

Nevertheless, the test results at least show that the KG constructed by our pipeline is useful in RAG as it can be used as the knowledge base to help the chatbot generate answers highly similar to the target answer, reflecting on a potentially high quality of the constructed KG and hence the performance of the KGC pipeline.

11.4.3 Remarks

This evaluation approach should also reflect both the *precision* and the *recall* of the KG. This is because the questions are directly relevant to the documents. To allow the answer to be more correct, the KG should precisely and thoroughly capture the information of the original document.

However, similar to the Evaluation via Reverse Engineering approach in Section 11.3. This approach cannot test the KGC pipeline alone. Both the KGC pipeline and RAG system can affect the overall performance.

This evaluation approach also tests the *conciseness* of the KG, i.e. the performance of the Stage: Coreference Resolution and Entity Disambiguation (7), or how well the KGC pipeline can merge two mentions referring to the same thing into a single entity.

In our testing, we deliberately restricted our RAG system to retrieve only one closest

entity for each keyword. For example, if "ANU" is a keyword in the question, only the closest entity, e.g., "ANU", will be returned, and the triple returned is further based on the matched entities only. Therefore, if "The Australian National University" does not merge with "ANU" as a single entity, the information (triples) about "The Australian National University" will be not be retrieved.

Therefore, without proper Coreference Resolution, the resultant KGs may still achieve a high score in the Evaluation via Reverse Engineering but not in the current Evaluation via Application.

11.5 Manual Evaluation

The manual evaluation is our tertiary evaluation approach because it is subjective and requires human effort, which is not easy to undertake and potentially lacks reliability. However, it can serve as the use case to demonstrate graphically how the actual KGs constructed by our KGC pipeline look compared to KGs constructed by the traditional KGC approach.

Since the KGs for a full paper can be significantly large. The manual evaluation is performed on SciERC, the abstract-only paper dataset. Specifically, we selected *the best* paper and the *worst paper* during the Evaluation via Reverse Engineering 11.3 to have a close look at these KGs. "The best" here means the highest similarity score between the synthesised document from the KG and the original document.

The best KG constructed by the pipeline using LLaMA in SciERC is Paper NO.83. It achieved a similarity score of 0.98 in Evaluation via Reverse Engineering. It is shown in Figure 11.14

The worst KG constructed by the pipeline using LLaMA in SciERC is Paper NO.64. It achieved a similarity score of 0.77 in Evaluation via Reverse Engineering. It is shown in Figure 11.16

Their target KGs provided by SciERC is shown in Figure 11.15 and Figure 11.17, respectively.

The predefined entity types of the target KG are Method, Material, Task, Metric, Generic, and OtherScientificTerm.

The predefined predicates of the target KG are FEATURE-OF, PART-OF, CONJUNCTION, COMPARE, USED-FOR, EVALUATE-FOR, and HYPONYM-OF.

Their original texts are as follows (Luan et al., 2018):

Paper NO.83

```
In this paper we present our recent work on harvesting English-Chinese
bitexts of the laws of Hong Kong from the Web and aligning them to the
```

```
subparagraph level via utilizing the numbering system in the legal text
hierarchy.

Basic methodology and practical techniques are reported in detail.

The resultant bilingual corpus, 10.4 M English words and 18.3 M Chinese
characters, is an authoritative and comprehensive text collection covering
the specific and special domain of HK laws.

It is particularly valuable to empirical MT research.

This piece of work has also laid a foundation for exploring and harvesting
English-Chinese bitexts in a larger volume from the Web .
```

Paper NO.64

```
Techniques for automatically training modules of a natural language generator
have recently been proposed, but a fundamental concern is whether the quality
of utterances produced with trainable components can compete with hand-
crafted template-based or rule-based approaches.

In this paper, We experimentally evaluate a trainable sentence planner for a
spoken dialogue system by eliciting subjective human judgments.

In order to perform an exhaustive comparison, we also evaluate a hand-crafted
template-based generation component, two rule-based sentence planners, and
two baseline sentence planners.

We show that the trainable sentence planner performs better than the rule-
based systems and the baselines, and as well as the hand-crafted system.
```

For Paper NO.83, the KGs constructed by the pipeline consist of more entities and relations than the target KG provided by SciERC. Since the target KG only captures entities that fall within a predefined type, it does not capture entities like “this paper” and “Hong Kong”. The target KG also does not capture relations like (“The resultant bilingual corpus”, “is”, “an authoritative and comprehensive text collection”).

Both graphs capture the key information that the bilingual corpus is used for empirical MT research. The target KG contains the triple (“bilingual corpus”, ”used for”, “empirical MT research”), whereas the constructed KG contains the triples (“It”, “is valuable to”, “empirical MT research”).

The relations in the target KG are more general and concise, while the relations in the constructed KG are more specific to the text.

One key observation is that in constructed KG, the entity “It” does not successfully merge with the entity “bilingual corpus,” which reduces the conciseness of the graph. As

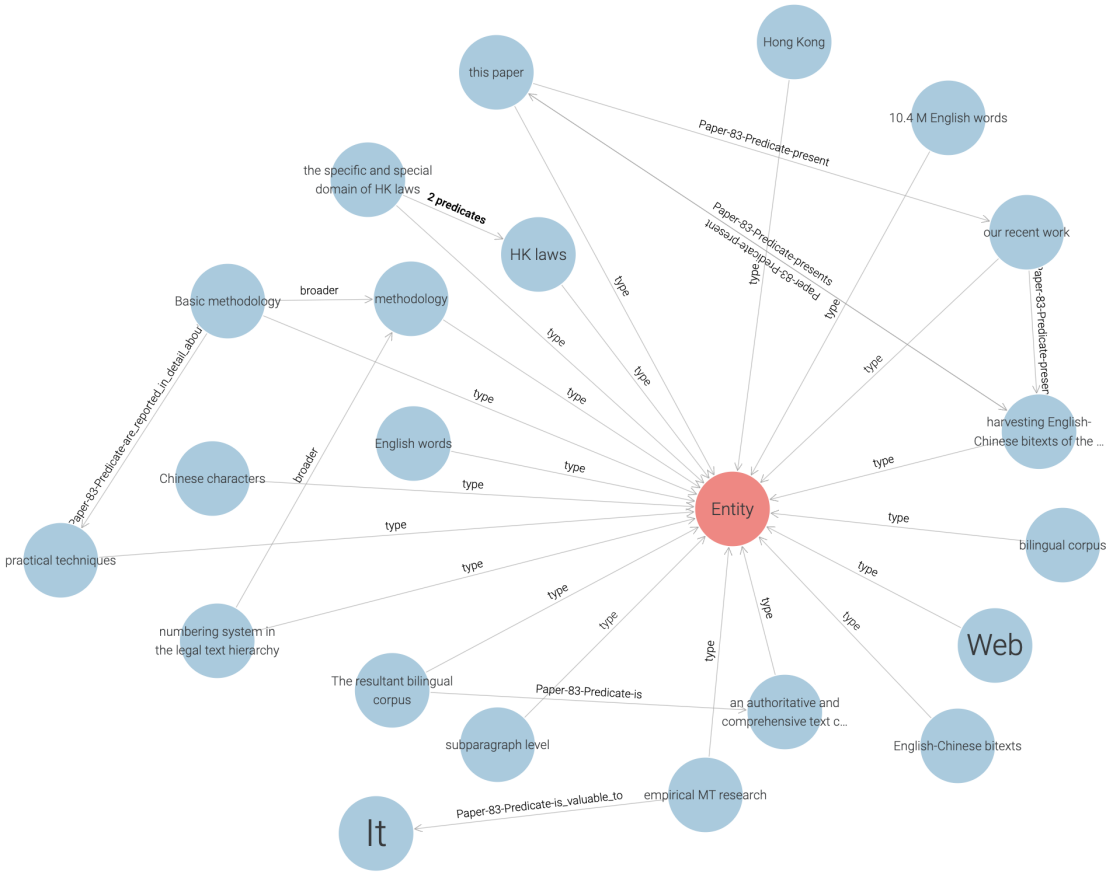

Figure 11.14: Constructed KG for Paper No.83

Figure 11.15: Target KG for Paper No.83

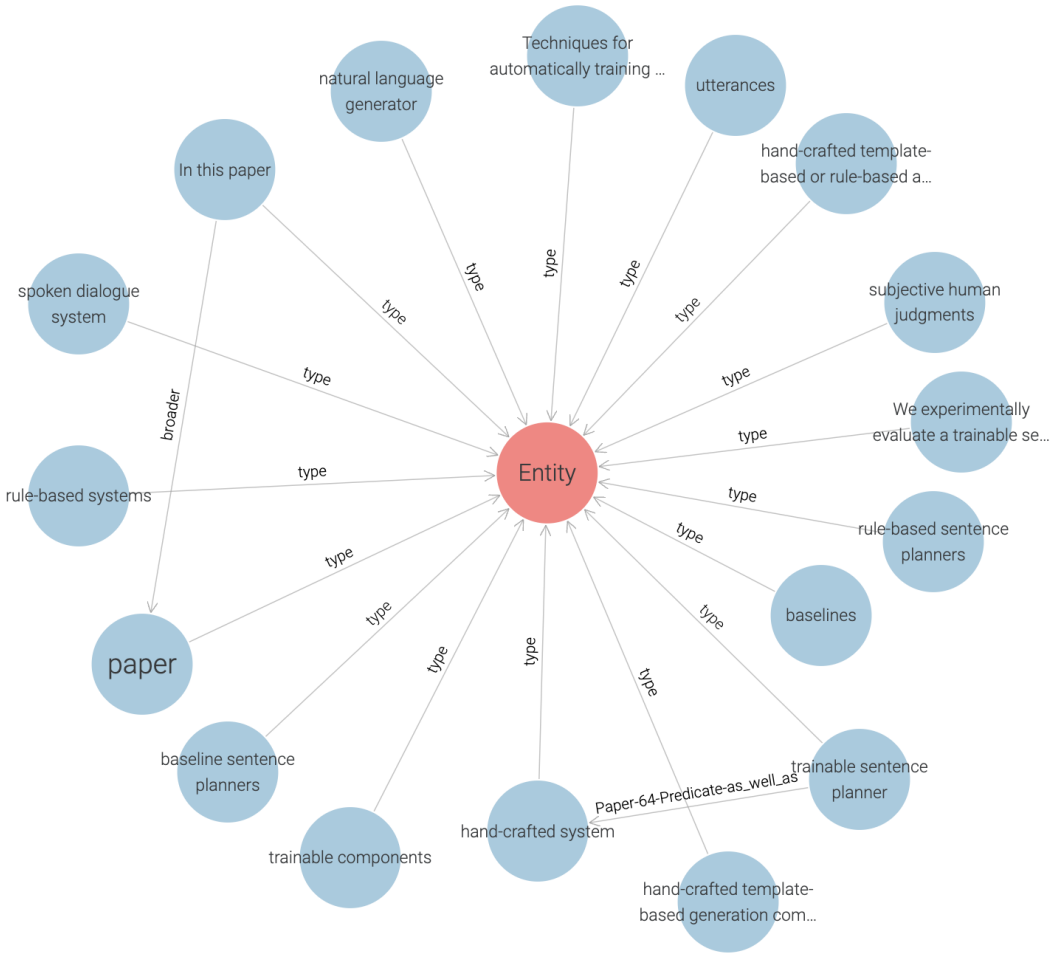

Figure 11.16: Constructed KG from Paper No. 64

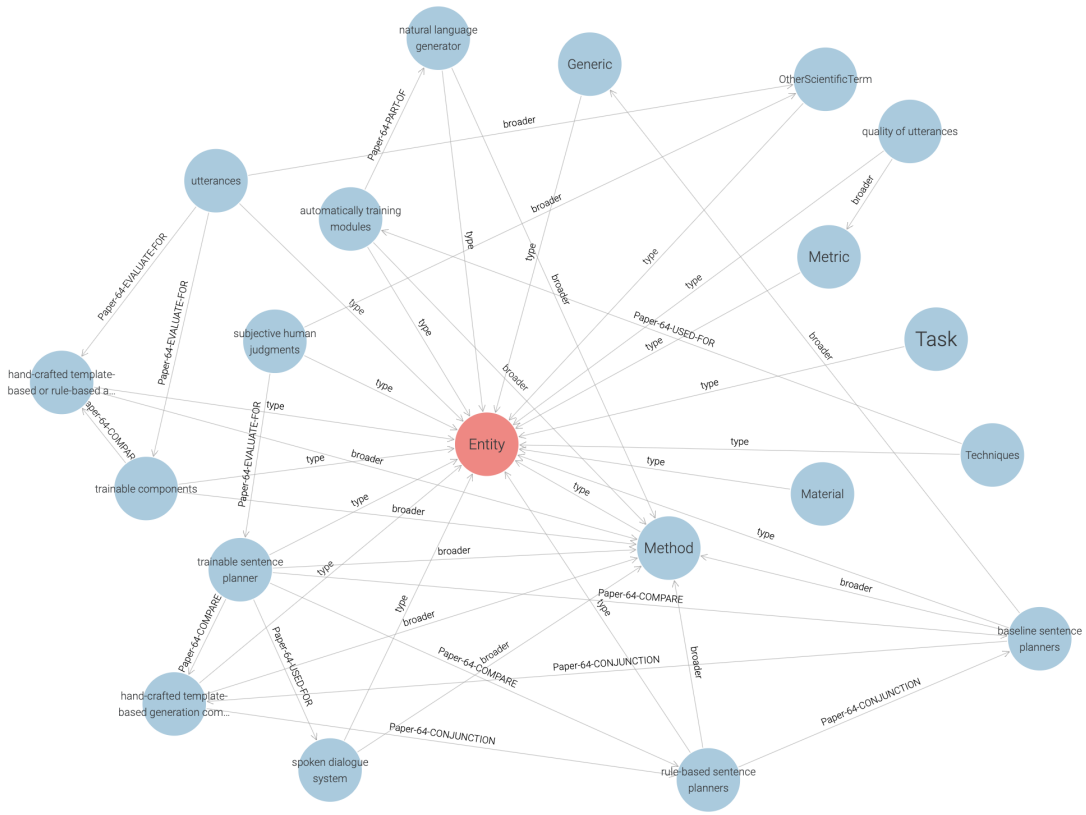

Figure 11.17: Target KG from Paper No. 64

Table 11.14: **Statistics** of the KGs constructed from the **Five-Full-Papers** dataset by **LLaMA**: Original and Ablation Study

Paper	Length (Token)	Entities	Mentions	Relations	Isolated Entities
Original	2811 ± 1434	356 ± 184	682 ± 384	789 ± 404	74 ± 31
Ablation Study	2811 ± 1434	332 ± 166	682 ± 384	662 ± 342	75 ± 32

mentioned in the last section (11.4), Evaluation via Reverse Engineering fails to evaluate the performance of the stage of Coreference Resolution. Even though this constructed KG achieves a high similarity score when reverted back to text, it does not mean the KG is concise.

For Paper NO.64, shown in Figure 11.16 and Figure 11.17, the target KG and the constructed KG contain a similar set of entities. However, the constructed KG fails to contain many relations the original paper has and includes in the target KG.

This is mainly due to the algorithm of Stage 3: Local Relation Extraction of the pipeline (3). We have forced all triples extracted by the LLM to appear explicitly in the document, both subject and object. The LLM somehow does not follow it, leading to many triples extracted being filtered out. However, if we did not filter out the triple with both its subject and object explicitly appearing in the original document, there is also a possibility that the triple is just hallucinated by the LLM and does not actually consist in the original document. Therefore, this is a trade-off.

11.6 Ablation Study

The general evaluation section (11.2.1) has suggested the removal of the LLM double-checking mechanism during the Coreference Resolution stage to aim to reduce the complexity from $O(2M + M^2)$ to $O(2M)$. Therefore, this section aims to see whether the LLM double-checking mechanism is significant to the pipeline by repeating the Evaluation via Reverse Engineering (11.3) and Evaluation via Application (11.4) on the KGs constructed by the pipeline with the removal of the LLM double-checking component.

Specifically, we only focus on the KGs constructed from the Five-Full-Paper dataset by the pipeline using LLaMA.

Tables 11.14 and 11.15 show the general statistics about the KGs constructed by the pipeline without the LLM double-checking mechanism and the KGs constructed by the original pipeline. It was found that the reduced pipeline has the same mentions as the original pipeline but fewer entities for all papers in the dataset. This is because the reduced pipeline merges multiple mentions into the same entity in the Coreference Stage

Table 11.15: The number of **Relations** of the KGs constructed from the **Five-Full-Papers** dataset by the **Original Pipeline** and **Reduced Pipeline** based on **LLaMA**

Paper	Relations (Original Pipeline)	Relations (Reduced Pipeline)
1	235	212
2	860	700
3	894	789
4	518	404
5	1437	1207
Mean/Std	789 ± 404	789 ± 404

Table 11.16: **Runtimes** (in Minutes) of KGC from the **Five-Full-Papers** dataset by **LLaMA**: Original and Ablation Study

Paper	S0	S1	S2	S3	S4	S5	S6	Time
Original	3.320E-6 ± 1.228E-6	16.66 ± 8.574	**122.9 ± 120.5**	9.074 ± 4.665	28.51 ± 9.551	11.06 ± 4.256	3.729E-3 ± 1.604E-3	188.2 ± 142.6
Ablation Study	3.320E-6 ± 1.228E-6	16.66 ± 8.574	**30.59 ± 19.84**	9.173 ± 4.702	27.49 ± 10.01	11.00 ± 4.325	4.531E-3 ± 2.239E-3	94.92 ± 45.31

more aggressively. However, it also led to fewer relations extracted. This is potentially due to two potential reasons.

1. One is that multiple mentions referring to different things are incorrectly merged into the same entity. Since only one description is kept during the merging, the description itself may not be correct. During both global and local relation extraction, descriptions are the key recourse to help the LLM understand an entity in order to extract the relation between this entity and other entities in a text. Since the description may not correctly describe the entity, LLM may be confused about what the entity is, and hence, fewer relations are extracted from this wrong entity.
2. The second reason is that the original LLM double-checking mechanism is too conservative, leading to some mentions referring to the same entity being treated as different entities. Therefore, some relations extracted can be redundant, e.g., ("ANU", "is a", "University") and ("The Australian National University", "is a", "University") if the mentions "ANU" and "The Australian National University" are not treated as the same entity. Therefore, there is a trade-off.

As shown in Tables 11.17 and 11.16, the runtime of Stage 2, Coreference Resolution and Entity Disambiguation, has been reduced significantly for all papers with the removal of the double-checking mechanism.

Table 11.17: **Runtimes** (in Minutes) of KGC from the **Five-Full-Papers** dataset by the **Original Pipeline** based on **LLaMA**

Paper	Paper Length	S2 (Original Pipeline)	S3 (Reduced Pipeline)
1	1165	20.18	8.374
2	3161	117.7	34.31
3	3062	79.03	34.13
4	1486	42.93	12.25
5	5183	354.6	63.92
Mean/Std	2811 ± 1434	122.9 ± 120.5	30.59 ± 19.84

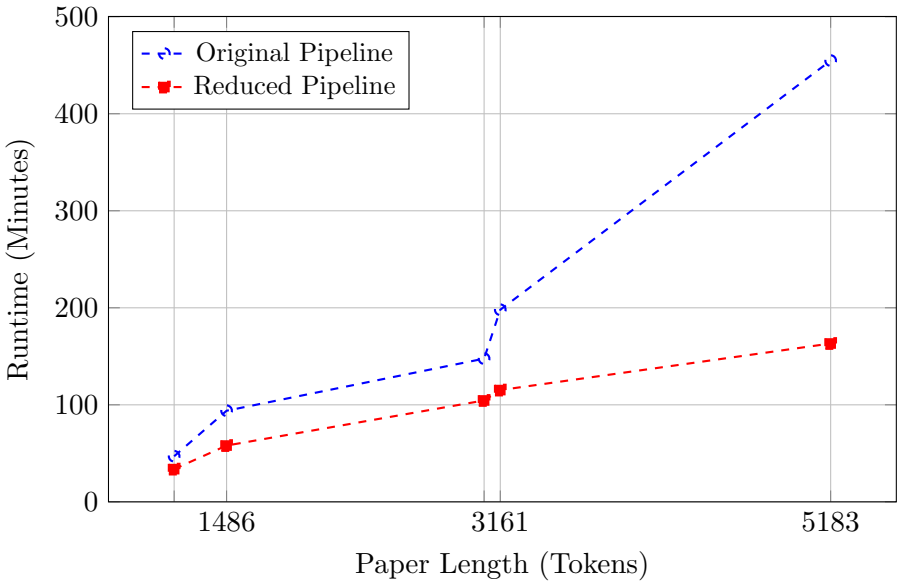

Figure 11.18: Runtime Comparison of Original and Reduced Pipelines vs. Paper Length

Figure 11.18 shows the total runtime of the pipeline with respect to papers of different lengths. It was found the runtime became approximately linearly proportional to the paper length. This aligns with the estimated pipeline complexity. By reducing $O(2M + M^2)$ to $O(2M)$, all *quadratic* components in the pipeline are removed. The complexity of the rest of the stages is all linearly proportional to the paper length, as discussed in the General Evaluation section (11.2.1). Even so, the time used to construct KG from papers is quite long, e.g., 2.72 hours for a 5183-tokens paper. Therefore, this is due to the fact that the pipeline is mainly based on the LLM decoder, and the LLM decoder itself has high complexity, leading to a high proportionality constant of the pipeline complexity. Therefore, further reducing complexity can be part of future work, e.g., reducing the constant proportionality or targeting a sub-linear relationship.

Table 11.18: Comparison of the **Similarity Scores** between the **original papers** and the **papers synthesised from the KGs** constructed by the **original pipeline** and the **reduced pipeline** based on **LLaMA**.

Paper	Original Pipeline	Reduced Pipeline
1	0.93945	0.9272
2	0.89893	0.8515
3	0.8916	0.8477
4	0.9224	0.9185
5	0.8330	0.7573
Mean/Std	0.8971 ± 0.03626	0.86044 ± 0.06116

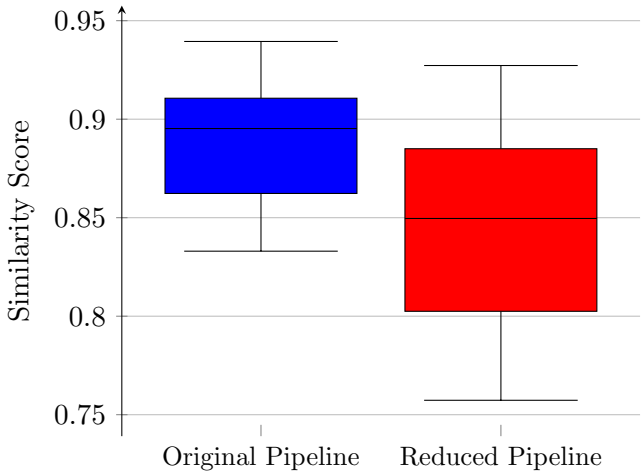

Figure 11.19: Boxplot of the **Similarity Scores** between the **original papers** and the **papers synthesised from the KGs** constructed by the **original pipeline** and the **reduced pipeline** based on **LLaMA**.

Table 11.18 and Figure 11.19 show the similarity scores between the original papers and the papers synthesised from the KGs constructed by the original pipeline and the reduced pipeline. Both the original pipeline and the reduced pipeline are based on LLaMA. Both KGs are constructed from the Five-Full-Paper dataset. It was found that the scores slightly decreased for all papers for the reduced pipeline. From the boxplot, it was found that the overall score is lower, and the standard deviation is higher. However, given the limited samples, we cannot claim that one is statistically different from the other because their p-value (0.333) is not small enough.

Therefore, this further shows that removing the double-checking mechanism may not significantly affect the performance estimated through the Evaluation via Reverse Engineering approach.

Table 11.19 and Figure 11.20 show the similarity scores between the target answers

Table 11.19: Comparison of the overall **Similarity Scores** between the target answers and the answer generated by the RAG system using the KGs generated by the **original pipeline** and the **reduced pipeline** based on **LLaMA**.

	Original Pipeline	Reduced Pipeline
Mean/Std	0.8842 ± 0.09523	0.8704 ± 0.09832

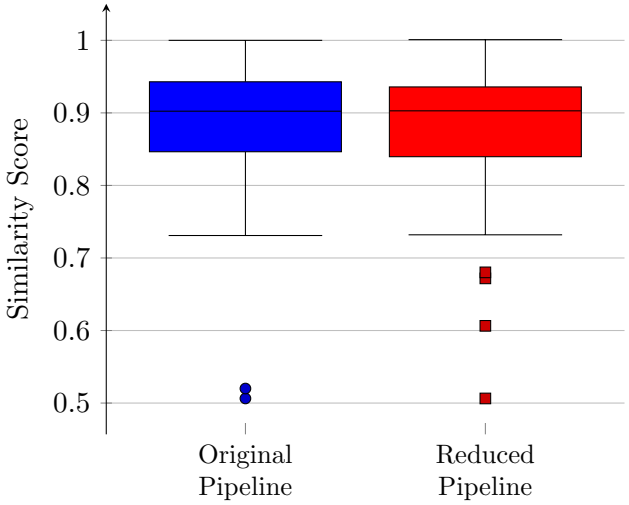

Figure 11.20: Boxplot of the **Similarity Scores** between the target answers and the answer generated by the RAG system using the KGs generated by the **original pipeline** and the **reduced pipeline** based on **LLaMA**.

and the answer generated by the RAG system using the KGs generated by the original pipeline and the reduced pipeline. Both pipelines are based on LLaMA, and both KGs are constructed from the Five-Full-Paper dataset. It was found that the overall results are quite similar. However, as shown in the boxplot, for the reduced pipeline side, there are more outliers. An outlier, especially for those with low similarity, potentially means the chatbot almost fails to answer the question, which is further due to the incorrectness or incompleteness of the KGs. Therefore, removing the double-checking mechanism may slightly reduce the quality of the RAG and thus further affect the performance of the RAG system. However, the effect is not statistically significant, given the limited samples.

Overall, the ablation study shows that removing the double-checking component in Stage 2, Coreference Resolution and Entity Disambiguation, does not significantly affect the performance of the pipeline based on the current dataset. However, a slight decrease in performance has been detected for both evaluation methods.

11.7 Summary of Limitations

The major limitation during the evaluation process is that the datasets used are too small, both for the sizes and numbers of samples. Only five full papers are used. The rest of the 100 papers are abstract only.

- This is mainly due to the fact that the current complexity of the pipeline is still high, and it can take a lot of time to build even a single local KG for a paper.
- The second reason is that by the start of our evaluation process, we were only given ten semi-structured papers from the ASKG. Since our pipeline does not start from raw text documents, it is hard for us to have more samples. Within these ten documents, one is used as a validation set, and 4 of them are preprocessed and used in our evaluation. The remaining six are still under preprocessing as we have found some major issues in these semi-structured documents, such as the incorrectness of indices of sentences and paragraphs, which can significantly affect later KGC.

The second limitation is the embeddings of documents themselves can be lossy. Both Evaluation via Reverse Engineering and Evaluation via Application heavily rely on the similarity score between the embedding of the predicated results and the ground truth results. However, even though their embeddings are found similar, the predicated results and the ground truth result may not be completely similar. Therefore, the results are only the estimation of their similarity.

The third limitation is that both Evaluation via Reverse Engineering and Evaluation via Application evaluate not only the KGC pipeline but also other components together. Therefore, the results may not clearly show the actual performance of the pipeline.

The fourth limitation is the narrow author group and topic of these papers. In the Five-Full-Paper dataset, all papers are in the field of computer science. Additionally, four of them are, in fact, the papers written by the supervisors of this project. Given the limited author group and topic, the linguistic patterns of these documents can also be limited. Different linguistic patterns can lead to different performances. Some may even cause the pipeline to fail if they are unhandled edge cases. Therefore, the evaluation may not truly affect the performance of the KGC pipeline of a variety of paper documents.