Open Local Knowledge Graph Construction from Academic Papers Using Generative Large Language Models

Haoting Chen
chen.haoting@outlook.com
School of Computing, The Australian National University
Acton, ACT, Australia

Sergio Rodríguez Méndez
Sergio.RodriguezMendez@anu.edu.au
School of Computing, The Australian National University
Acton, ACT, Australia

Pouya Ghiasnezhad Omran
P.G.Omran@anu.edu.au
School of Computing, The Australian National University
Acton, ACT, Australia

Abstract

This manuscript introduces paper2lkg, a novel Local Knowledge Graph Construction (KGC) pipeline designed to transform individual academic papers into their structured local Knowledge Graph (KG) representations. The pipeline leverages Large Language Models (LLMs), particularly generative LLMs, to automate key Natural Language Processing (NLP) tasks in KGC. It aims to help enrich an existing academic KG that lacks detailed local representations of individual papers or construct new academic KGs from local KGs through further Knowledge Graph Alignment (KGA).

CCS Concepts

• **Information Systems**; • **Information Retrieval**; • **Document Representation**;

Keywords

Knowledge Graph Construction, Natural Language Processing, Large Language Model

ACM Reference Format:
Haoting Chen, Sergio Rodríguez Méndez, and Pouya Ghiasnezhad Omran. 2025. Open Local Knowledge Graph Construction from Academic Papers Using Generative Large Language Models. In *Proceedings of 4th International Workshop on Natural Language Processing for Knowledge Graph Construction (4th NLP4KGC Workshop).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Since Google introduced modern Knowledge Graphs (KGs) in 2012, KGs have gained widespread adoption, particularly in Information Retrieval (IR) [6, 15]. Compared to raw text documents, KGs are more structured and machine-readable [6, 15]. For example, a paper introducing Model A, in its KG representation, may have a concrete node of Model A linked to all of its dependencies, algorithms, and performance metrics. Therefore, querying KGs enables more efficient access to essential information than directly parsing unstructured text documents.

4th NLP4KGC Workshop, April, 2025, Sydney, Australia

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

In recent years, several academic KGs have been developed to assist researchers in retrieving and organising scholarly information, which includes Open Research Knowledge Graph (ORKG) [8], Microsoft Academic Graph (MAG) [5], and AMiner [23]. These KGs often focus on integrating knowledge across extensive academic corpora, aligning concepts, citations, and authorship to form a unified view of research domains [5, 8, 23]. While global knowledge is crucial, preserving document-specific knowledge can help users who need to find the exact documents where a concept is mentioned or explore how a concept is discussed differently across papers.

Hence, this research introduces paper2lkg, a pipeline for constructing *local* KGs from academic papers. Each local KG captures the content of a single paper only but in greater detail. The constructed local KGs can then be used to enrich an existing academic KG or form a more global KG on their own through Knowledge Graph Alignment (KGA). Such a global KG may also provide a unified view of a topic while clearly maintaining the differences in discussions between papers.

The academic KG that paper2lkg is designed for and tested on is the ANU Academic Scholarly Knowledge Graph (ASKG), an experimental academic KG that stores research papers from The Australian National University (ANU) [9, 30]. Like other academic KGs, ASKG focuses on globally linking and organising metadata across various sources [9, 30]. However, it lacks a deep semantic representation of the internal structure and meaning of each paper [9, 30], i.e., a local sub-KG that captures the entities and relationships.

This manuscript is structured as follows. Section 2 provides an additional background of local Knowledge Graph Construction (KGC) and KGC using Large Language Models (LLMs). Section 3 reviews related work on KGC with generative LLMs and outlines the expected contributions in addressing the research gaps. Section 4 provides an overview of the pipeline. Sections 5 and 6 present the evaluation methods and results, respectively. Section 7 concludes the paper and discusses potential directions for future research.

2 Background

The process of local Knowledge Graph Construction (KGC) from documents involves three main steps: Named Entities Recognition (NER), Entity Linking (EL), and Relation Extraction (RE) [6, 32]. NER identifies Named Entity Mentions within a document [6]. EL links and clusters Mentions referring to the same Entity into a single node in the output, which further requires Entity Disambiguation and Coreference Resolution to determine any identical Mentions referring to different Entities or different Mentions referring to the same Entity [6]. Finally, RE connects Entities Nodes with Predicate edges extracted from the original document [6].

Over time, approaches to local KGC have evolved significantly. Early methods mainly relied on rule-based techniques, such as syntactic parsing [6]. Later, statistical machine learning enables more diverse feature extraction, allowing classifiers to be trained to perform tasks like detecting Named Entity tokens or Relations between Entities based on language features [6, 15]. With the advance of the early pre-trained Large Language Models (LLMs) like BERT, the process of crafting feature extractors becomes automatic [15]. More recently, autoregressive generative LLMs, such as GPT and LLaMA, have further eliminated the step of training a classifier, as they can directly process instructions written in natural language to perform NER, EL, and RE [4, 17].

KGC based on generative LLMs is expected to be more flexible and *open* but less robust and efficient than the more traditional LLM encoders + classifiers approach. For example, during NER, the way of using generative LLMs can be as simple as directly asking an LLM chatbot to obtain an answer about a list of Named Entities the LLM thinks the input text has [1, 26]. Because of its generative nature, the LLM can detect Named Entities that do not fall into some common predefined types [26]. In contrast, NER with LLM encoders + classifiers requires the classifiers to be trained or fine-tuned on some samples to become functional [29]. However, samples are inevitably limited and usually cover a finite number of Entity Types. If the classifier is trained to detect the PERSON and LOCATION entities, it may fail to detect the ALGORITHM entities. A similar observation applies to RE tasks. Even so, generative LLMs can be unstable because every token output in its answer is based on how likely it can follow the previous token, and the LLMs do not inherently know and are able to validate facts [15]. Therefore, it can sometimes be overconfident and tend to produce an answer even when there is no answer, which is also known as the Hallucination issue [15]. Generative LLMs like GPT and LLaMA work autoregressive [15, 18]. That is, to generate one new token in the response, the prompt + the response generated so far must be processed through the whole LLM again, whereas the extraction of features or embeddings from a text using an LLM encoder only goes through the LLM once [18]. Therefore, KGC using generative LLMs is expected to be of high complexity [18].

The local KGC pipeline presented in this research is primarily based on generative LLMs. This choice is driven not only by the fact that the use of generative LLMs in KGC remains relatively recent but also by the nature of academic papers, which frequently introduce new terms, such as Named Entity Types or Predicates, that fall outside predefined types. Therefore, using generative LLMs for open KGC helps capture more comprehensive and dynamic knowledge from papers.

3 Related Work and Research Goals

By the time this research was initiated in early 2024, there had been a few studies related to *local KGC using generative LLMs*. However, these studies tend to mainly focus on the NER step of KGC instead of presenting a full KGC pipeline [1, 25, 28]. Additionally, some studies choose to make generative LLMs behave in the same way as traditional LLM encoders + classifiers, i.e., forcing the generative LLM to generate Entities or Relations that fall within a predefined set of types. While this approach ensures alignment with existing

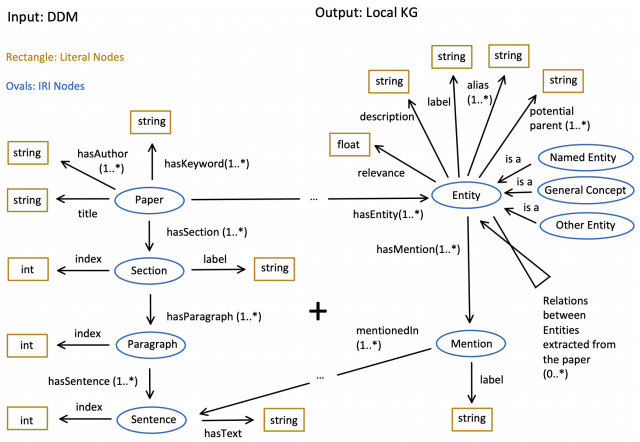

Figure 1: The Pipeline IO Interface

taxonomies, it limits the full potential of generative LLMs. Their prompts can be generalised into styles like "Please return the Named Entities found in the text belonging to PERSON.", where the specific types to retrieve are explicitly mentioned in the prompt. Despite these limitations, these early works pioneered the integration of generative LLMs into KGC and laid the foundation for future research. For example, the works of Ashok et al., Wang et al., and Wang et al. [1, 25, 28] all present NER systems leveraging generative LLMs such as ChatGPT [12]. They have utilised a number of Prompt Engineering techniques, such as Chain-of-Thought and Few-Shot learning, to achieve high performance in some datasets [1, 25, 28], demonstrating the possibility of using generative LLMs in KGC.

Throughout 2024, researchers tend to shift focus to the concept of Open Information Extraction [4, 17, 26], e.g., the prompt style has become "Please return the Named Entities found in the text.", without specifying the types or is able to detect types on-the-fly. This enables a more comprehensive construction of KG. These examples include ChatIE [26], a NER + RE + Event Extraction system that can dynamically detect the types or predicates a sentence has and then extract Entities or Relations falling into the previously detected Entity types or predicates, and works of Papaluca et al., Edge et al., and Zhang et al. [4, 17, 31], which step further to extract entities and relations without type restrictions directly. Papaluca et al. and Zhang et al. [17, 31] creatively combine NER and RE into a single Triple Extraction step, e.g., "Extract any (Subject, Predicate, Object) triples from the given text." However, most research targets generic documents rather than academic papers. Not all of these models include the EL process, and the complexity of using generative LLMs is rarely measured and discussed.

This research aims to make the following contributions to the field of KGC. (1) It presents a functional open local KGC pipeline prototype that specialises in academic papers and covers all main tasks required in KGC. (2) The output local KGs of the pipeline prototype may help enrich and construct academic KGs, e.g., ASKG. (3) The evaluation of the prototype can briefly explore efficacy and efficiency using generative LLMs in KGC, especially the complexity.

4 Pipeline Overview

The paper2lkg pipeline artifact introduced in this research is available at https://w3id.org/kgcp/paper2lkg.

The input to the paper2lkg is any academic paper that has been preprocessed from a raw document file, such as a PDF, into a semi-structured representation called the Deep Document Model (DDM). A portion of the DDM ontology of a paper, as well as the specific pat used by paper2lkg as its input, is shown on the right of Figure 1. The input DDM is based on the RDF-graph structure and has organised the document into a hierarchy of sections, paragraphs, and sentences, along with its metadata. This enables paper2lkg's algorithms to efficiently traverse the document or access specific parts for NLP tasks. More details on DDM can be found at https://w3id.org/kgcp/DDM.

The output of paper2lkg is the DDM plus the local KG representation of the paper, as shown in the right part of Figure 1. It consists of Entities and Relations extracted from the original paper. By definition, an **Entity** is anything that exists separately from other things and has its own identity [14], which can be represented as an Internationalized Resource Identifier (IRI) node in an RDF graph [22]. Each Entity has one or more occurrences, called **Mentions**, linked to a specific Sentence in the Paper. There are several attributes stored in an Entity, including its label, aliases, description, and relevance to the paper. To enhance semantic organization, **Entities** are classified into only one of the three *disjoint* categories. (1) **Named Entities** involve any unique, specific real-world object, such as "ANU". (2) **General Concepts** involve any fundamental or widely recognised concepts, such as "University", which can typically be found in a dictionary. (3) **Other Entities** are anything that falls outside the two above, which can include but is not limited to: (a) composite or non-general concept, such as a "tool that maps JSON into its RDF Graph representation.", a concept without possibly a standardised term; (b) any pronoun or referential expression, e.g., "She" and "It" that is ready to be resolved into a Named Entities or General Concept.

The pipeline involves *five* stages, which slightly extend the regular three stages: NER, EL, and RE. These include: (1) Mention Extraction, in replace of NER; (2) Entity Linking; (3) Local Relation Extraction (4) Global Relation Extraction (5) Taxonomy Generation and Predicate Resolution. The following subsection provides an overview of all stages of the pipeline in terms of functionalities, algorithms, and LLM prompts. For brevity, the full algorithms, prompts, and full discussion about functionalities, design choices, and limitations are available under the `/documentation/pipeline-overview` directory of the repository of paper2lkg.

4.1 Stage 1: Mention Extraction

Algorithm 1 (High-Level)

```
Def classify_mentions:
    Let the Named Entities, Entities, and Mentions
    extract from a section/paragraph/sentence as S1,
    S2, and S3, respectively.

    Named Entities = S1
    General Concept = S2 - S1
    Others = S3 - (S2 - S1) - S1

Def get_reference:
    Find the specific sentence where a Mention is
    extracted through string matching.

For {target} in
  [Named Entity, Named Entity + General Concept, Entity]:
    For each section:
        For each paragraph:
            For each sentence:
                Call the LLM using Prompt 1.
                Call classify_mentions
        Call the LLM using Prompt 1.
        Call classify_mentions
        Call get_reference for each Mention
    Call the LLM using Prompt 1.
    Call classify_mentions
    Gall get_reference for each Mention
```

Prompt 1 (High-Level)

```
## Task Definition
Given a text, extract all {target} from it and
assign them one or more potential parent types...
## Output Format
Please write your answer in JSON, as shown below...
## Examples...
## Input Text...
```

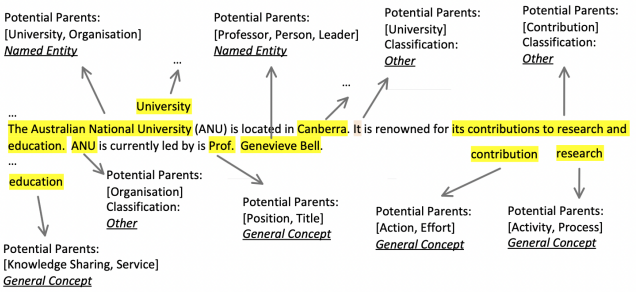

Figure 2: Stage One Example

This stage has been designed to extract not only the Mentions of Named Entities (e.g., "Wikidata") but also General Concepts (e.g., "Knowledge Graph") and anything else as long as they form some relations. This is because Knowledge in the real world is not only about the relations between two Named Entities but also between Named Entities and General Concepts (e.g., Java enables Software Development) or between two general terms (e.g., Cats eat Fish). This ensures information preservation to a larger extent, hence, a more holistic resultant KG from the text document.

To ensure successful extraction and categorisation of Mentions by LLM, the system follows a Narrow-to-Broad approach, i.e., first prompting the LL to extract only the Mentions of a Named Entity, then Named Entities plus General Concepts, and finally Entities (Everything). Set subtractions are applied to the three initial Entity Mention sets to obtain the three classes of Entity Mentions

The Narrow-to-Broad approach is applied instead of directly extracting these three classes of Entities separately. During development, it was observed that generative LLMs prefer and perform well on prompts with as few restrictions as possible, e.g., not limiting the extraction of general concepts only but all entities.

As shown in Algorithm 1, the extraction is performed at all section, paragraph, and sentence levels. During the development, it was found that this can minimise entity misses because the LLMs may not notice some entities unless a wider context is given, but a small context allows the LLMs to concentrate better. If inconsistent results are detected, e.g., an Entity is extracted as an Other Entity at the sentence level but is extracted as a Named Entity at the section level, the algorithm will be biased toward the result from a larger context.

The prompt, as shown in Prompt 1, is written in Markdown syntax because research indicates that LLMs can understand structure syntax better than plain text [24]. It follows the "Task Definition", "Examples", and "Input" style as many other related works [1, 25, 28]. Its response format is set to JSON because JSON can be easily parsed into Python Dictionary and processed by the pipeline program.

What is extracted along with the Mentions of Entities are the potential parent types of those Entities. These potential parents are helpful in generating Entities' descriptions in Stage 2 and taxonomic relations in Stage 5.

The complexity is in terms of the number of times a generative LLM is called and is measured by $O(9L)$, where L is the length of the document. This is because the algorithm loops through the document at three different levels for three different targets. The reason that only the number of times a generative LLM is called is measured by the complexity is that it is extremely time-consuming and dominant compared to calling an LLM encoder or all other logical code.

A simple example of how the Mention Extraction is performed shown in Figure 2.

4.2 Stage 2: Entity Linking

Algorithm 2 (High-Level)

```
For each mention extracted,
    Call the LLM using Prompt 2.1. Obtain Familiarity.
For each mention extracted,
    If Familiar,
        Call the LLM using Prompt 2.2.1
    Else,
        Call the LLm using Prompt 2.2.2
    Obtain Description.
For each mention,
    Call the LLM encoder.
    Obtain the Embedding of the Description
For each mention,
    For each mention,
      If the Cosine Similarity of their Embeddings > 0.95,
            Link two mentions temporarily.
Find a partition of Cliques that are maximum-sized and
disjoint from the graph Mentions and temporary links. For
each Clique, group them into an Entity Node.
```

Prompt 2.1 (High-Level)

```
... Given a mention, without seeing the context, can you
tell me what the mentions refer to....
```

Prompt 2.2.1 (High-Level)

```
...Given a mention, its types, the abstract of the paper
(paper-level context), the specific section where the
mention appears (section-level context), and the specific
sentence the mention locates (sentence-level context),
write a single-sentence description...
```

Prompt 2.2.1 (High-Level)

```
...Given a mention, its types, and the specific sentence
the mention locates (sentence-level context), write a
single-sentence description...
```

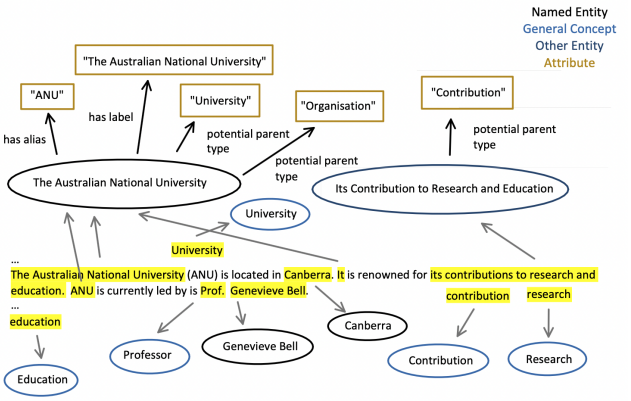

Figure 3: Stage Two Example

Entity Linking involves grouping Mentions that refer to the same thing as Entity nodes in the final KG. Mentions in a group can be anywhere in the original document in different forms, such as a "Jack" in the first sentence and "He" in the last sentence. Therefore, the idea behind this stage is to first convert each Mention into a clear Description using the Mention itself and its context. A Description should clearly represent the Entity a Mention refers to. For example, a "Jack" and "He" should result in a very similar Description. Each Description is further converted into Embedding so that Cosine Similarity can be used to check whether two Descriptions are similar and hence the similarity in two Mentions so that two Mentions can link to with each other.

The reason for using Embeddings + Cosine Similarity to determine the similarity of two mentions instead of directly asking a generative LLM whether two descriptions are the same is that the latter can result in quadratic complexity.

How much information is used to generate a description is based on whether the LLM know the Mention based on its built-in Knowledge and whether the Mention can easily become ambiguous, e.g., "He". This ensures a more accurate description generation for any new terms introduced by academic papers, and if a mention is well-known and general, avoid its Description being too specific to the document.

Within the graph of mentions with some temporary links, instead of grouping Mentions as long as they form a connected subgraph, only Mentions belonging to the same Cliques, i.e., a complete subgraph, are grouped into the same Entity. This ensures the transitivity of Mentions, i.e., Mention A = Mention B and Mention B = Mention C implies Mention A = Mention C. In order to group the maximum number of Mentions while ensuring each Mention only belongs to one Entitiy, the algorithm tries to create a maximum-sized clique partition of the graph. This is done by greedily applying the Bron–Kerbosch algorithm to obtain a single maximum clique, removing all nodes in that clique, and repeating.

When an Entity has more than one Mention, the label, i.e., the most representative name of the Entity, is chosen by the label of the Mention first appears in the original document. This is based on the assumption that academic papers usually use the full name of a term at the beginning but its abbreviation in the rest of the text. The alias and potential parents of an Entity are the union of the alias and potential parents of all its Mentions.

The complexity of this stage is $O(2M)$, where M is the number of Mentions, which can be easily derived from Algorithm 2. Figure 3 is an intuitive example of how Entity Linking is performed.

4.3 Stage 3: Local Relation Extraction

Algorithm 3 (High-Level)

```
For each section:
    for each paragraph:
        for each sentence:
            Fetch all Entities it has.
            Call the LLM using Prompt 3.
        Fetch all Entities it has.
        Call the LLM using Prompt 3.
    Fetch all Entities it has.
    Call the LLM using Prompt 3.
```

Prompt 3 (High-Level)

```
...Given a section/paragraph/sentence and a list of
Entities in it, extract all the relations between the
Entities and return in the form of
(Subject, Predicate, Object) triples...
```

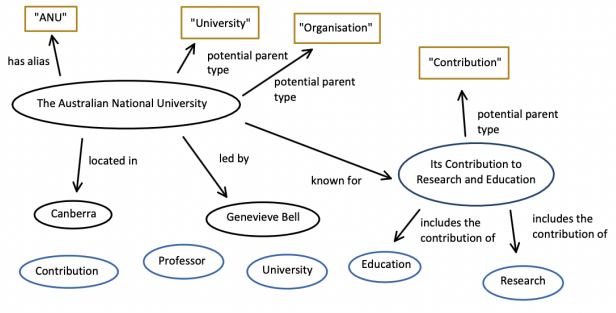

Figure 4: Stage Three Example

Stage Three focuses on extraction relations between Entities within a limited context, i.e., a specific Section, Paragraph, and Sentence. After Stage Three, some Predicate edges are drawn to relate some Entity nodes based on the original text, as shown in Figure 4. Its complexity can be modelled as $O(3L)$ its algorithm loops through the whole document three times at three different levels, as shown in Algorithm 3.

4.4 Stage 4: Global Relation Extraction

Algorithm 4 (High-Level)

```
Shrink the document by first shrinking its innermost
Section and then the outer Sections using Prompt 4.1 until
the whole document fits the context limit of the
generative LLM.

Embed the shortened document and Embed all the Entities
using their potential parent types and descriptions.

Compute a relevancy score for each entity.

For all of the 10% most relevant entities of the first
20 relevant entities, depending on which one is smaller,
    Pairwisely, call the LLM using Prompt 4.2
```

Prompt 4.1 (High-Level)

```
...Given a section, summarise it into half of its original
length while retaining key information and ideas...
```

Prompt 4.2 (High-Level)

```
...Given the title, authors, keywords, and content of
the paper, as well as the types and the description of
each of the two entities, extract the **most important**
predicate that connects them, if it exists, return in
the form of (Subject, Predicate, Object) triple...
```

In order for the output KG to be holistic, the KGC pipeline also considers relations that span across the whole paper in addition to those existing within a section, paragraph, or sentence, e.g., (“This paper”, “Concludes”, “XXX”). However, a paper-level context is required for this task, and it can exceed the context limit of some LLMs. Therefore, this stage starts with shrinking the paper carefully by assuming that the inner sections are less important than the outer sections, such as the Introduction and Conclusion, as shown in Algorithm 4. This is based on the common sense of how a paper is usually structured. Since it can be computationally expensive to check whether the relations exist between any two Entities at the paper level, the Global Relation Extraction is only performed within the most relevant entities to the paper. For each pair of the most relevant entities, the maximum number of relations the LLM can extract is limited to one, as shown in Prompt 4.2. This is because, during development, it was found that the LLM can be overconfident and extract too many irrelevant relations between entity pairs.

The complexity can be modelled as $O(L + Max(20^2, (0.1 \cdot E)^2)$ where L is the length of the paper and E is the number of entities.

The effect of Stage 4 on the example shown in Figure 4 would be more edges coming in or out of the Entities like "The Australian National University" from elsewhere in the document.

4.5 Stage 5: Taxonomy Generation and Predicate Resolution

Algorithm 5 (High-Level)

```
For each Entity as E1,
    For each Potential Parent Type of the Entity as T,
        Call Prompt 5.1.
        Obtain a Description, denoted as Descr(T)
        For each Entity as E2,
           If Embed(Descr(T) similar to Embed(Descr(E2))
             Add (E1, has a broader term, E2) into the KG

For each (S, P, O) triple in the KG,
    Call Prompt 5.2
    Obtain the Description of P, denoted as Descr(P)
For each P as P1,
    For each P as P2,
        If Embed(Descr(P1) similar to Embed(Descr(P2)),
            Link two Ps temporarily.
Find a maximum-sized partition of cliques from the graph of
Predicates with temporary links. For each clique, group
Predicates together.
```

Prompt 5.1 (High-Level)

```
...Given an Entity Type, by using your background knowledge,
write a single-sentence description for it...
```

Prompt 5.2 (High-Level)

```
...Given a Predicate in a (S, P, O) triple, write a
description for the Predicate. The description should not
specifically refer to the Subject or Object in the given
triple, but tell how the Predicate is used to relate any
Subjects and Objects of the same type as the Subject or
Object in the given triple.
```

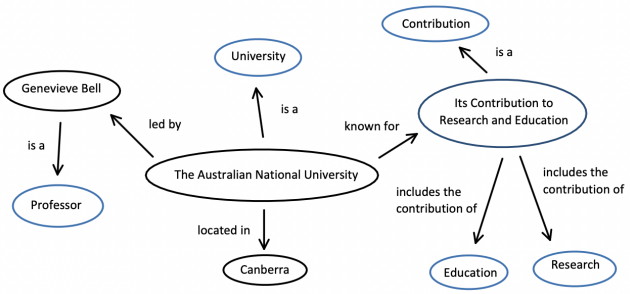

Figure 5: Stage Five Demo

The Taxonomy Generation focuses on extracting only the taxonomic relations between Entities within the document. Although Stages 3 and 4 should have extracted some explicit taxonomic relations from the text, such as a sentence in the form of "XX is a XX.", there are still many hidden relations that can be semantically derived from the original document, e.g., the name of ANU itself suggesting that it is a university. This step creates more complete taxonomic relations between Entities and their parents so that the resultant KG can be more expressive.

This step is performed by generating descriptions for all Entities and their potential parent types found in Stage One. If the description of a potential parent of Entity A is found to be highly similar to the description of Entity B, then Entity B is a parent of Entity A. However, since the Relations are indirectly derived without referring back to the original document, this step may introduce some non-existing relations, especially due to the Hallucination issue of generative LLMs.

The process for Predicate Resolution is similar to Entity Linking, where similar predicates are grouped together, e.g., "founded in" and "established in". This increases the conciseness and enhances the semantics of the resultant KG.

The complexity can be approximately modelled as $O(E * P + R)$, where E is the number of Entities, R is the number of triples, and the constant P is the average number of potential parent Entities an Entity has.

5 Experiment Setup

The experiment starts by using the pipeline to generate local KGs for some academic papers. These documents include four full papers from ASKG and one external paper [7, 16, 19–21], denoted as The Five-Paper dataset. Paper in this dataset has lengths ranging from 1156 to 5183 tokens. In addition, 100 abstract-only papers from the SciERC dataset [10] are used with an average length of 134 ± 51 tokens. The pipeline is run by a desktop computer with AMD Ryzen 5 5600X 6-Core Processor, 16 GB RAM, and GeForce RTX 3080 12GB on the platform Pop! OS 22.04 LTS. The generative LLM used throughout the pipeline is Meta-Llama-3-8B-Instruct.Q4_0 (Context Limit: 8192) [11] or gpt-4o-mini (Context Limit: 128k) [12]. The LLM encoder used for generating embeddings is BAAI–bge-m3 (Context Limit: 8192) [27] for Stage Five and BAAI–bge-base-en-v1.5 (Context Limit: 512) [3] for all other stages. The only two *independent variables* in the experiment are the input papers, and the generative LLM used (LlaMA or GPT). The pipeline's performance is estimated using the following three approaches.

5.1 General Evaluation

General Evaluation measures the number of Entities, Mentions, and Relations of the local KGs created by the pipeline against papers with different lengths. It also measures the wall-clock runtime (for LlaMA only) or bill (for GPT only) to compare them with the complexity modelled in Section 4. The reason for measuring expense in generating KGs instead of runtime is that the Internet speed can be unstable, but the bill is directly proportional to the usage.

5.2 Evaluation via Reverse Engineering

The Five-Paper dataset is unlabelled, i.e., there does not exist a ground true local KG for the pipeline to compare with. Even though the papers in SciERC do come with ground true KGs [10], it can be hard to tell how well the resultant KGs are by directly compare with their ground true KGs because different KGs may have different

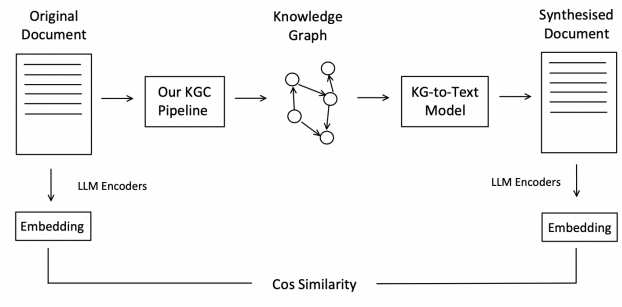

Figure 6: The Process of Evaluation via Reverse Engineering

ontology or schema, especially for the KG created through Open Information Extraction, e.g., the types of Entities and Predicates the resultant KG has is created on-the-fly instead of pre-defined. Therefore, this research uses two indirect ways to evaluate the resultant KGs and, hence, the pipeline, which will be discussed in the current 5.2 and the next subsection 5.3.

The idea behind Evaluation via Reverse Engineering is to test the reversibility of a local KG when converting it back to a text document, i.e., how much information in the original document can be captured by the local KG. This method is inspired by how an autoencoder is trained and tested [2]. A simple KG-to-text system was also created for testing purposes in this research. This system can loop through all the triples in a local KG in batch from the most relevant ones to the least relevant ones and convert them into a text segment for each batch, also by calling generative LLMs. The relevancy of a triple is calculated by the sum of the relevancy of its entities obtained in Stage Four. The text segments are further concatenated to form a synthesised document containing the most important information appearing at the beginning. The embeddings of the original document and the synthesised document are then compared and produce a similarity score.

This approach can reflect both the precision and recall of a constructed KG because a reversible KG indicate that it should capture enough and correct information from the original document. However, this method cannot evaluate the conciseness of a KG, i.e., a KG without a proper Entity Linking process and Predicate Resolution can still achieve a very high result because these two processes do not extract new information but to make a KG tiny and more semantical. Another drawback is that it has to test both the KGC and KG-to-text systems at the same time and hence, the result may not precisely reflect either system.

5.3 Evaluation via Application

The idea behind this evaluation method is to test how well a local KG can be used in practice. This method is inspired by the evaluation method of Edge et al.'s work [4]. A more SOTA and large-scale generative LLM, GPT-O1 [13], with a context limit larger than any of the documents in the test set, is used to read directly the whole document and generate 10 question-answer pairs as the ground true, which serves as the role of "Teacher". The test document is also transformed into local KG as if learning by a student in their mind. A Q&A chatbot receives the ten questions from the "Teacher"

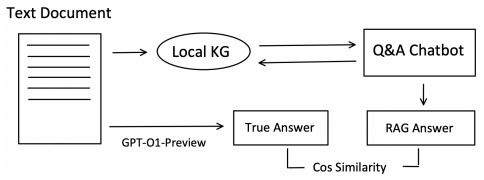

Figure 7: The Process of Evaluation via Application

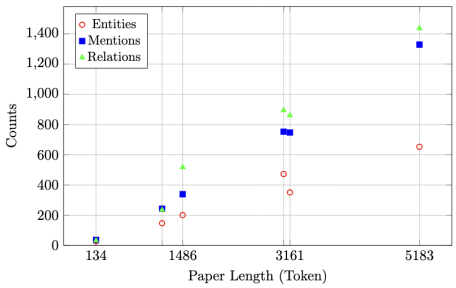

Figure 8: Statistics of the KGs Constructed from SciERC and The Five Papers Dataset using LlaMA

to perform graph-based Retrieval Augmented Generation (RAG) to try to answer the questions by only using the local KG generated. The true answers and the RAG answers are then compared, and similarity scores are obtained and serve as the performance metrics.

A simple Q&A chatbot utilising graph-based RAG has also been created for testing purposes. It works by first extracting entities in the questions using generative LLMs (LlaMA or GPT), querying the KG for any one-hop triples that match the extracted entities, and finally using generative LLMs again to answer the question assisted by the retrieved triples.

6 Experiment Results

For brevity, this section only presents the key findings of each evaluation appraoch. The complete experiment results, as well as the guide for reproducing the results, are available under the `/documentation/evaluation/` directory of the repository of paper2lkg.

6.1 General Evaluation

Figure 8 shows the number of Entities, Mentions and Relations of the local KG constructed from the 100 abstract-only papers from SciERC and the Five Papers dataset using LlaMA. It was found that the Mentions number is approximately linearly proportional to the paper length, which is as expected because the longer the paper is, the more terms it has. The relation of Entities and Paper length is sublinear, which is also as expected. This is because every paper should have its central theme, i.e., it does not keep introducing new things or topics all the time but rather discusses a limited number of things throughout the document. This shows that the Entity Linking part is functional, i.e., it is able to merge terms referring to the same thing into some limited number of Entities to avoid the number of Entities growing linearly with the paper length. The relationship

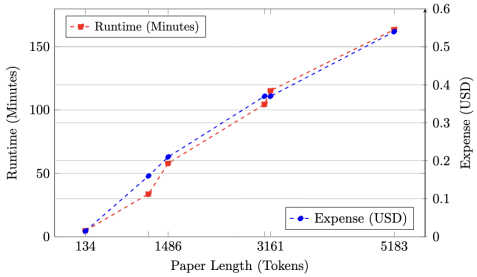

Figure 9: Wall-Clock Runtime (LlaMA) and Expense (GPT) to Construct KGs from SciERC and The Five-Paper Dataset

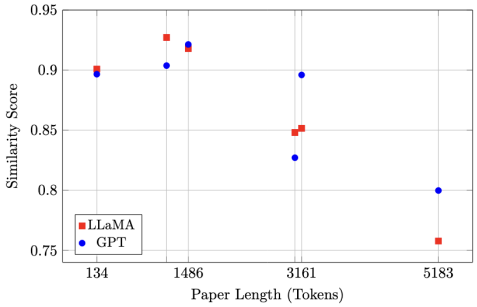

Figure 10: Similarity Scores of KGs Generated by LlaMA or GPT from Papers in Different Lengths

between Entities and Relations is expected to be quadratic because a graph with more nodes is expected to have a quadratically greater potential edge between them. Since Entities vs. Paper Length is sublinear. Then, the Relations vs. Paper Length turns back to linear, which is also demonstrated in the graph. Similar observations apply to KGs generated by GPT.

Figure 9 shows the wall-clock runtime and expense with respect to papers with different lengths. Both time and expense are bounded linearly as the paper length increases. This aligns with the complexity estimated in Section 4, where the complexity for all stages does not contain quadratic terms but is all linearly proportional to either the paper length L, mentions M, entities E, and relations R and all M, E, and R are not more than linearly proportional to L as demonstrated before. Even though the pipeline has achieved a linear complexity, generally speaking, spending up to 150 minutes on generating a local KG of 5000-token paper is still considered inefficient. The nature of using generative LLM makes the proportional constant high. Therefore, there is still room for improvement when the pipeline is used in practice.

6.2 Reverse Engineering and RAG Test

Figure 10 shows the similarity scores between the original documents and synthesised documents when using different models and papers of different lengths to perform KGC and KG-Text. It was found that the similarity scores for the current datasets and models are all relatively high, at about 0.9. However, there is also a trace that the scores slightly decrease as the paper length increases, suggesting that the model may find it challenging to perform KGC for long papers.

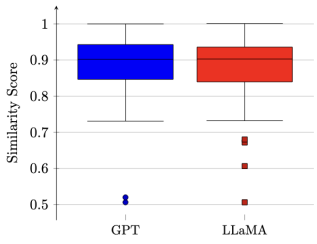

Figure 11: Boxplots of the Results of RAG Tests for KGs Generated from the Five Paper Dataset Using LLaMA and GPT

Only the Five Paper dataset is used in the RAG test. A boxplot showing the distribution of similarity scores between the RAG answers and the true answers (5 * 10 samples in total) for each LLM models are shown in Figure 11. Although the overall score is close to the high 0.9 standards, it is worth noticing in Figure 11 that there exist some outliers, meaning that the RAG system fails to answer some questions using the local KG, which further shows that the resultant KG has some information loss.

7 Conclusion and Future Work

This research has introduced a pipeline capable of converting academic papers into their local KG representations using LLMs. Evaluation indicates that the KGs generated by the pipeline can retain information to a relatively high extent and can potentially be used by a graph-based RAG system to answer users' questions. However, the high computational cost of generative LLM calls remains a challenge.

In relation to the Research Goals in Section 3. Goal 1 is considered achieved as a functional early-stage pipeline prototype has been developed, covering all key tasks in KGC. Regarding Goal 2, while paper2lkg is technically ready for deployment in ASKG, its output KG quality still requires refinement since academic content needs to be strictly accurate. The evaluation of paper2lkg has demonstrated the efficacy and efficiency of generative LLMs in KGC on a limited dataset. However, this only serves as a preliminary reference.

Future work can focus on two directions: "scaling up" and "scaling out". Scaling up means improving the performance of the pipeline itself, e.g., by providing better prompt designs to increase the quality of the resultant KGs and lower complexity. "Scaliing out" means providing a complete ecosystem for paper2lkg. This includes a Knowledge Graph Alignment system to interconnect multiple local KGs into a larger academic KG based on entity similarities. A dedicated graph-based RAG system, with distributed agents processing individual local KGs and a central master agent integrating their insights to generate comprehensive answers to user queries. Future work can also focus on the ethical part, e.g., the high complexity of generative LLMs in KGC and, hence, potentially high carbon emissions, as well as the privacy risk in KG, which increases the accessibility of personal information while enabling more efficient IR.

References

[1] Dhananjay Ashok and Zachary C. Lipton. 2023. PromptNER: Prompting For Named Entity Recognition. arXiv:2305.15444 [cs.CL] https://arxiv.org/abs/2305.15444

[2] Dor Bank, Noam Koenigstein, and Raja Giryes. 2021. Autoencoders. *Deep Learning in Science* (2021). https://api.semanticscholar.org/CorpusID:212717965

[3] Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In *Findings of the Association for Computational Linguistics ACL 2024*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand and virtual meeting, 2318–2335. https://doi.org/10.18653/v1/2024.findings-acl.137

[4] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. *arXiv e-prints*, Article arXiv:2404.16130 (April 2024), arXiv:2404.16130 pages. https://doi.org/10.48550/arXiv.2404.16130 arXiv:2404.16130 [cs.CL]

[5] Drahomira Herrmannova and Petr Knoth. 2016. An Analysis of the Microsoft Academic Graph. *D Lib Mag.* 22 (2016). https://api.semanticscholar.org/CorpusID:26876682

[6] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. 2021. Knowledge Graphs. *ACM Comput. Surv.* 54, 4, Article 71 (July 2021), 37 pages. https://doi.org/10.1145/3447772

[7] M. Numan Ince, Joseph Ledet, and Melih Gunay. 2019. Building An Open Source Linux Computing System On RISC-V. In *2019 1st International Informatics and Software Engineering Conference (UBMYK)*. 1–4. https://doi.org/10.1109/UBMYK48245.2019.8965559

[8] Mohamad Yaser Jaradeh, Allard Oelen, Manuel Prinz, Markus Stocker, and S. Auer. 2019. Open Research Knowledge Graph: A System Walkthrough. In *International Conference on Theory and Practice of Digital Libraries*. https://api.semanticscholar.org/CorpusID:202550338

[9] Runsong Jia, Bowen Zhang, Sergio J. Rodríguez Méndez, and Pouya G. Omran. 2024. Leveraging Large Language Models for Semantic Query Processing in a Scholarly Knowledge Graph. arXiv:2405.15374 [cs.IR] https://arxiv.org/abs/2405.15374

[10] Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-Task Identification of Entities, Relations, and Coreference for Scientific Knowledge Graph Construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, Brussels, Belgium, 3219–3232. https://doi.org/10.18653/v1/D18-1360

[11] Meta. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

[12] OpenAI. 2024. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL] https://arxiv.org/abs/2303.08774

[13] OpenAI. 2024. Learning to Reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/

[14] Oxford University Press. 2024. Oxford Learners Dictionaries. https://www.oxfordlearnersdictionaries.com/definition/english/entity

[15] Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying Large Language Models and Knowledge Graphs: A Roadmap. *IEEE Transactions on Knowledge and Data Engineering* 36, 7 (2024), 3580–3599. https://doi.org/10.1109/TKDE.2024.3352100

[16] Shuaiqun Pan, Sergio J. Rodríguez Méndez, and Kerry Taylor. 2022. A Pipeline for Analysing Grant Applications. arXiv:2210.16843 [cs.LG] https://arxiv.org/abs/2210.16843

[17] Andrea Papaluca, Daniel Krefl, Sergio Rodríguez Méndez, Artem Lensky, and Hanna Suominen. 2024. Zero- and Few-Shots Knowledge Graph Triplet Extraction with Large Language Models. , 12–23 pages. https://doi.org/10.18653/v1/2024.kallm-1.2

[18] Jesse Roberts. 2024. How Powerful are Decoder-Only Transformer Neural Models?. In *2024 International Joint Conference on Neural Networks (IJCNN)*, Vol. 1. IEEE, 1–8. https://doi.org/10.1109/ijcnn60899.2024.10651286

[19] Sergio J. Rodríguez Méndez. 2018. Modeling actuations in BCI-O: A Context-based Integration of SOSA and IoT-O. In *Proceedings of the 8th International Conference on the Internet of Things, IoT 2018 (ACM International Conference Proceeding Series)*. Association for Computing Machinery. https://doi.org/10.1145/3277593.3277914 Publisher Copyright: © 2018 Copyright is held by the owner/author(s). Publication rights licensed to ACM.; 8th International Conference on the Internet of Things, IoT 2018 ; Conference date: 15-10-2018 Through 18-10-2018.

[20] Sergio J. Rodríguez Méndez, Pouya G. Omran, Armin Haller, and Kerry Taylor. 2021. MEL: Metadata Extractor & Loader. Publisher Copyright: © 2021 CEUR-WS. All rights reserved.; 2021 International Semantic Web Conference Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice, ISWC-Posters-Demos-Industry 2021 ; Conference date: 24-10-2021 Through 28-10-2021.

[21] Sandaru Seneviratne, Sergio J. Rodríguez Méndez, Xuecheng Zhang, Pouya G. Omran, Kerry Taylor, and Armin Haller. 2021. TNNT: The Named Entity Recognition Toolkit. In *Proceedings of the 11th Knowledge Capture Conference* (Virtual Event, USA) *(K-CAP '21)*. Association for Computing Machinery, New York, NY, USA, 249–252. https://doi.org/10.1145/3460210.3493550

[22] W3C. 2014. RDF 1.1 Concepts and Abstract Syntax. https://www.w3.org/TR/rdf11-concepts/

[23] Huaiyu Wan, Yutao Zhang, Jing Zhang, and Jie Tang. 2019. AMiner: Search and Mining of Academic Social Networks. *Data Intelligence* 1 (2019), 58–76. https://api.semanticscholar.org/CorpusID:85527423

[24] Ming Wang, Yuanzhong Liu, Xiaoyu Liang, Songlian Li, Yijie Huang, Xiaoming Zhang, Sijia Shen, Chaofeng Guan, Daling Wang, Shi Feng, Huaiwen Zhang, Yifei Zhang, Minghui Zheng, and Chi Zhang. 2024. LangGPT: Rethinking Structured Reusable Prompt Design Framework for LLMs from the Programming Language. arXiv:2402.16929 [cs.SE] https://arxiv.org/abs/2402.16929

[25] Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. 2023. GPT-NER: Named Entity Recognition via Large Language Models. arXiv:2304.10428 [cs.CL] https://arxiv.org/abs/2304.10428

[26] Xiang Wei, Xingyu Cui, Ning Cheng, Xiaobin Wang, Xin Zhang, Shen Huang, Pengjun Xie, Jinan Xu, Yufeng Chen, Meishan Zhang, Yong Jiang, and Wenjuan Han. 2024. ChatIE: Zero-Shot Information Extraction via Chatting with ChatGPT. arXiv:2302.10205 [cs.CL] https://arxiv.org/abs/2302.10205

[27] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-Pack: Packed Resources For General Chinese Embeddings. , 9 pages. https://doi.org/10.1145/3626772.3657878

[28] Tingyu Xie, Qi Li, Jian Zhang, Yan Zhang, Zuozhu Liu, and Hongwei Wang. 2023. Empirical Study of Zero-Shot NER with ChatGPT. , 7935–7956 pages. https://doi.org/10.18653/v1/2023.emnlp-main.493

[29] Yongxiu Xu, Heyan Huang, Chong Feng, and Yue Hu. 2021. A Supervised Multi-Head Self-Attention Network for Nested Named Entity Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 16 (May 2021), 14185–14193. https://doi.org/10.1609/aaai.v35i16.17669

[30] Bowen Zhang, Sergio J. Rodríguez-Méndez, and Pouya Ghiasnezhad Omran. 2023. ASKG: An Approach to Enrich Scholarly Knowledge Graphs through Paper Decomposition with Deep Learning. *CEUR Workshop Proceedings* 3632 (2023). Publisher Copyright: © 2023 Copyright © 2023 for this paper by its authors.; 22nd International Semantic Web Conference on Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice, ISWC-Posters-Demos-Industry 2023 ; Conference date: 06-11-2023 Through 10-11-2023.

[31] Bowen Zhang and Harold Soh. 2024. Extract, Define, Canonicalize: An LLM-based Framework for Knowledge Graph Construction. , 9820–9836 pages. https://doi.org/10.18653/v1/2024.emnlp-main.548

[32] Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2023. A Comprehensive Survey on Automatic Knowledge Graph Construction. *ACM Comput. Surv.* 56, 4, Article 94 (Nov. 2023), 62 pages. https://doi.org/10.1145/3618295

Received 1 January 2025; revised 7 February 2025; accepted xx xx 2025