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

Haoting Chen
u7227871@anu.edu.au
School of Computing, Australian National University
Canberra, ACT, Australia

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

Pouya Ghiasnezhad Omran
P.G.Omran@anu.edu.au
School of Computing, Australian National University
Canberra, 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 harnesses Large Language Models (LLMs), particularly generative LLMs, to automate key Natural Language Processing (NLP) tasks in KGC. The constructed local KGs can potentially be used to enrich an existing academic KG that lacks detailed local representations of individual papers or further integrated into new academic KGs through Knowledge Graph Alignment (KGA).

CCS Concepts

• **Computing methodologies** → **Information extraction**; **Semantic networks**; • **Information systems** → *Document representation.*

Keywords

Knowledge Graph Construction, Natural Language Processing, Large Language Model

ACM Reference Format:

Haoting Chen, Sergio José Rodríguez Méndez, and Pouya Ghiasnezhad Omran. 2025. Open Local Knowledge Graph Construction from Academic Papers Using Generative Large Language Models. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3701716.3717820

1 Introduction

Since Google introduced modern Knowledge Graphs (KGs) in 2012, KGs have gained widespread adoption, particularly in Information Retrieval (IR) [10, 21]. Compared to raw text documents, KGs are more structured and machine-readable [10, 21]. For instance, a paper introducing Model A, in its KG representation, would have a concrete node of Model A linked to all of its dependencies, structure, and performance metrics, which enables more efficient retrieval of related information compared to searching within the original document.

In recent years, several prominent academic KGs have been developed to assist researchers in retrieving and organising scholarly data, which includes Open Research Knowledge Graph [12], Microsoft Academic Graph [9], and AMiner [31]. These KGs often focus on integrating knowledge across extensive academic corpora, aligning concepts, citations, and authorship to form a unified view of research domains [9, 12, 31]. While global knowledge is crucial, preserving document-specific knowledge can help researchers locate 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. These local KGs can then be used to enrich an existing academic KG or serve as building blocks for a broader academic KG 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.

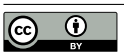
The primary academic KG for which paper2lkg is designed and tested is the ANU Academic Scholarly Knowledge Graph (ASKG), an experimental academic KG that stores research papers from The Australian National University (ANU) [13, 38]. Like other academic KGs, ASKG focuses on globally linking and organising metadata across various sources [13, 38]. However, it lacks a deep semantic representation of each paper [13, 38], i.e., a local sub-KG that captures the Entities and Relations within.

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) [10, 40]. NER identifies Named Entity Mentions within a document [10]. EL links and clusters Mentions that refer to the same Entity into a single Entity node, which requires Entity Disambiguation and Coreference Resolution to determine whether identical Mentions refer to different Entities or whether different Mentions refer to the same Entity [10]. Finally, RE connects Entity nodes with Predicate edges derived from the semantics of the original document. [10].

Over time, approaches to local KGC have evolved significantly. Early methods relied mainly on rule-based techniques, such as



WWW Companion '25, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1331-6/2025/04
https://doi.org/10.1145/3701716.3717820

syntactic parsing [10]. Later, statistical machine learning enables more diverse feature extraction, allowing classifiers to be trained to detect Named Entity tokens, Entities' similarity, or Relations between Entities based on language features. [10, 21]. With the advancement of the early pre-trained Large Language Model (LLM) encoders like BERT [6], the process of crafting feature extractors becomes automated [21]. More recently, autoregressive generative LLMs, such as GPT and LLaMA, have further eliminated the need for training classifiers, as they can directly process natural language instructions to perform all NER, EL, and RE [2, 28, 35].

KGC based on generative LLMs is expected to be more flexible and *open* but less robust and efficient than traditional LLM encoder-classifier approaches. For example, in NER, using generative LLMs can be as simple as asking a chatbot for a list of Named Entities from a given text [2, 35]. Becuase it is a generative task, the LLM is free to extract as many Entities as it identifies [35]. In contrast, traditional LLM approaches often treat NER as a classification task, where an LLM encoder is fine-tuned alongside a classifier. The encoder extracts text features and outputs embeddings while the classifier maps these embeddings into labels, determining whether a token at a certain position is a Named Entity and its corresponding type. However, fine-tuning usually relies on a limited set of training samples, restricting the number of Entity Types the classifier can recognise. For example, a classifier trained to detect PERSON and LOCATION Entities may fail to detect ALGORITHM Entities. A similar challenge exists in RE tasks.

Even so, generative LLMs can be unstable because each token they generate is probabilistically determined based on preceding tokens, and they typically lack inherent fact-checking abilities [21]. It can also sometimes be overconfident and generate answers even when none exist, which is known as Hallucination [21]. Additionally, generative LLMs like GPT and LLaMA operate autoregressive [21, 25]. That is, to generate each new token, the model must process the entire prompt and previously generated response again, leading to higher computational complexity [25]. In contrast, traditional encoder-classifier approaches typically generate outputs in a single forward pass, making them more efficient [14].

Despite the potential drawbacks, 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 topic of using generative LLMs in KGC remains relatively recent but also by the nature of academic papers, which frequently introduce new Named Entity and Predicates that may not be recognised by traditional encoder-classifier approaches. Thus, using generative LLMs allows local KGC to be more adaptive and comprehensive.

3 Related Work and Research Goals

By early 2024, a few studies had partially explored KGC using generative LLMs. These studies focus on a specific step of KGC, mainly NER, rather than developing a full KGC pipeline [2, 34, 37]. Additionally, these studies typically constrained generative LLMs to behave like traditional LLM encoder-classifier models, e.g., limiting the Entity types it can output in NER or Relations it can detect in RE, which may not fully leverage the advantage of generative LLMs. Their prompts follow structures such as: "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, early research laid the groundwork for integrating generative LLMs into KGC. For example, the works of Ashok et al., Wang et al., and Wang et al. [2, 34, 37] all present their NER systems leveraging generative LLMs such as ChatGPT [19], but with different combinations of techniques, such as Zero-Shot, Few-Shot, Chain-of-Thought, and Syntactic Augmentation, which demonstrates the possibility of using generative LLMs in KGC and the efficacy of some LLM techniques [2, 34, 37].

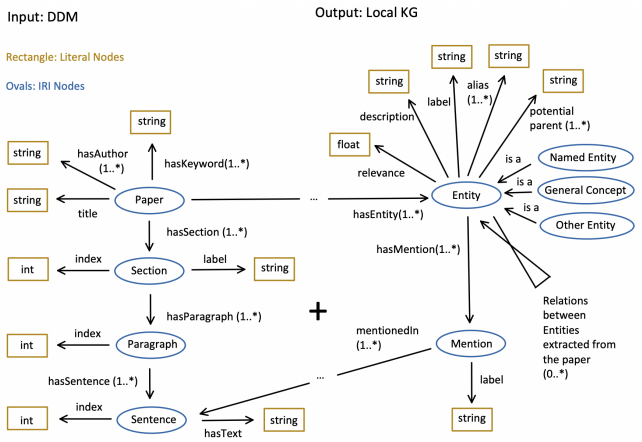


Figure 1: The Pipeline IO Interface

Throughout 2024, researchers shift focus towards the concept of Open Information Extraction [7, 23, 35]. For instance, a NER prompt may become "Please return the Named Entities found in the text." without specifying types. Examples include ChatIE [35], a system equipped with NER, RE, and Event Extraction and all Entity types, Predicates, and Event types are detected on the fly. Additionally, the systems developed by Papaluca et al. and Zhang et al. [23, 39] step forward to merge NER and RE into a single Triple Extraction step to reduce LLM calls and minimise error propogation. However, these researches mostly target generic documents rather than academic papers. Furthermore, many models lack an EL process, and the complexity of using generative LLMs in KGC is rarely measured and discussed.

This research aims to contribute to the field of KGC by achieving the following: (1) developing a functional open local KGC pipeline prototype tailored for academic papers, covering all key KGC tasks; (2) generating local KGs that can enrich existing academic KGs (e.g., ASKG) or support academic KG construction; (3) briefly exploring the efficacy and efficiency of generative LLMs in KGC through the evaluation of the KGC pipeline prototype built.

4 Pipeline Overview

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

The input to 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 Paper that is used by paper2lkg as its input is shown on the right of Figure 1. The input DDM

is based on the RDF-graph structure. It organises a paper into a hierarchy of Sections, Paragraphs, and Sentences, along with the paper's metadata. This enables paper2lkg's algorithms to efficiently traverse the paper 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 a set of Entities along with some attributes, such as labels, aliases, and descriptions. An **Entity** is anything that exists separately from other things and has its own identity [20]. In this case, the identity is an Internationalized Resource Identifier (IRI) [30]. Each Entity is linked to zero or more other Entities through some Predicates. Every Entity has one or more occurrences, called **Mentions**, linked to a specific Sentence in the Paper. To enhance semantic organisation, **Entities** are categorised into three mutually exclusive classes. (1) **Named Entities** refer to unique, specific real-world objects [10], such as "ANU," and are often leaves in a taxonomic graph. (2) **General Concepts** represent any abstract and widely recognised things, such as "University," and are often non-leaf nodes in a taxonomic graph. (3) **Other Entities** are anything that falls outside the two above, including but not limited to: (a) composite or non-general concept, such as a "tool that maps JSON into its RDF Graph representation.", which may not have a standardised term; (b) any pronoun or referential expression, e.g., "She" and "It" that is yet to be resolved into Named Entities or General Concepts.

The pipeline involves *five* stages, which slightly extend beyond the standard three stages NER, EL, and RE, while the separation between NER and RE remains becuase EL is involved. These include (1) Mention Extraction, replacing 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 pipeline stages, covering their functionalities, algorithms, and LLM prompts. For brevity, complete algorithms, prompts, and a more detailed discussion of the design choices and limitations for each stage can be found in 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
    extracted from a Section/Paragraph/Sentence be 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} and assign
them one or more potential parent types...
## Output Format
Please write your answer in JSON, as shown below...
## Examples...
## Input Text...
```

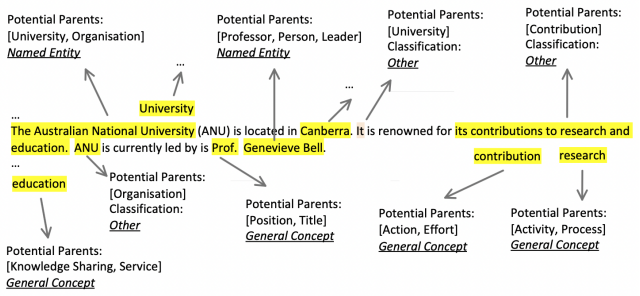


Figure 2: Stage 1 Example

This stage is 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 involves not only Relations between two Named Entities but also between Named Entities and General Concepts (e.g., Java enables Software Development) or between General Concepts (e.g., Cats eat Fish). This preserves information to a larger extent when transforming papers into local KGs.

To ensure successful extraction and categorisation of Mentions by LLM, the system follows a Narrow-to-Broad approach, as shown in Algorithm 1, i.e., first prompting the LLM to extract only the Mentions of Named Entities, then Named Entities plus General Concepts, and finally all Entities. Set subtractions are applied to the three initial Entity Mention sets to derive the three distinct classes of Entity Mentions

While extracting these three classes of Entities separately seems more intuitive, development observations showed that generative LLMs perform better when given fewer restrictions, e.g., not limiting extraction to General Concepts but including all Entities.

The extraction is also performed at all Section, Paragraph, and Sentence levels, as shown in Algorithm 1. During 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 a General Concept at the Sentence level but is extracted as a Named Entity at the Section level, the algorithm prioritises the result from a broader context.

Extracted along with the Entity Mentions are the potential parent types of the Entity that the Mention refers to. These potential parents are helpful when generating Entities' descriptions in Stage 2 and taxonomic Relations in Stage 5.

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 [33]. It follows the "Task Definition", "Examples", and "Input" style as other related works [2, 34, 37]. Its response format is set to JSON because a JSON output can be easily parsed into a Python Dictionary and processed by the pipeline algorithms.

The complexity is measured by the number of generative LLM calls. Only the number of generative LLM calls is measured because each call is computationally expensive, making the time spent on LLM encoders and logical code negligible in comparison. For Stage 1, it is $O(9L)$, where L is the length of the document. This is becuase the algorithm iterates through the document at three different context levels to extract three different scopes of targets.

A simple example of how the Mention Extraction is performed is 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 a 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 with 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 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 where the Mention appears (Sentence-level context),
write a single-Sentence description...
```

Prompt 2.2.1 (High-Level)

```
...Given a Mention, its types, and the specific Sentence
where the Mention appears (Sentence-level context), write
a single-Sentence description...
```

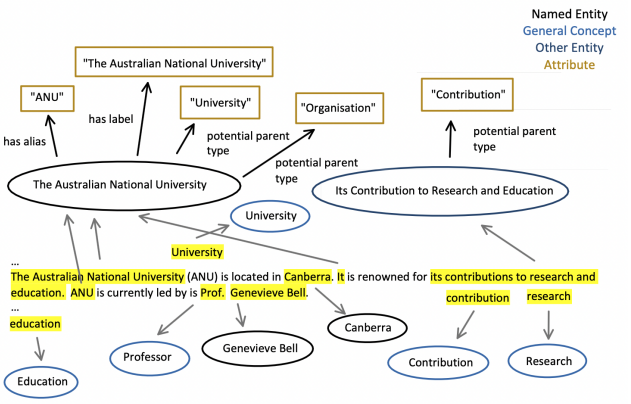


Figure 3: Stage 2 Example

Entity Linking (EL) involves grouping Mentions that refer to the same thing as Entity nodes in the final KG. Mentions within a group may appear in different forms across the document, such as "Jack" in one Sentence and "He" in another Sentence. To achieve EL, this stage first converts each Mention into a clear description leveraging both the Mention itself and its context. A description should accurately represent the Entity to which the Mention refers. For instance, "Jack" and "He" should generate highly similar descriptions. Each description is further converted into embedding, allowing Cosine Similarity to detect similar Mentions for grouping.

How much information is used to generate a description depends on whether the LLM knows the Mention based on its built-in Knowledge and whether the Mention is potentially ambiguous. For example, the Mention "He" can be ambiguous and requires a larger context to determine its meaning. This approach ensures a more accurate description generation for any new or ambiguous terms. Conversely, if a Mention is already well-known and general, it encourages the LLM to produce a more general description by reducing its exposure to the specific paper content.

Once similar Mentions are identified, a temporary link is drawn between each similar pair, forming a graph of Mentions. However, instead of grouping all connected Mentions into an Entity, only Mentions forming a complete subgraph (clique) are grouped together. This guarantees transitivity, meaning that if Mention A = Mention B and Mention B = Mention C, then Mention A = Mention C. To maximise the number of grouped Mentions while ensuring each Mention belongs to only one Entity, the algorithm tries to

find a maximum-sized clique partition of the graph. This is done by greedily applying the Bron-Kerbosch algorithm to obtain the current largest clique, removing all nodes in that clique, and repeating.

When grouping multiple Mentions into an Entity, the label, i.e., the most representative name of the Entity, is assigned based on the label of the Mention that first appears in the original document. This follows the assumption that academic papers typically introduce a term in full before using its abbreviation. The alias and potential parents of an Entity are derived from the union of the alias and potential parents of all its Mentions.

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

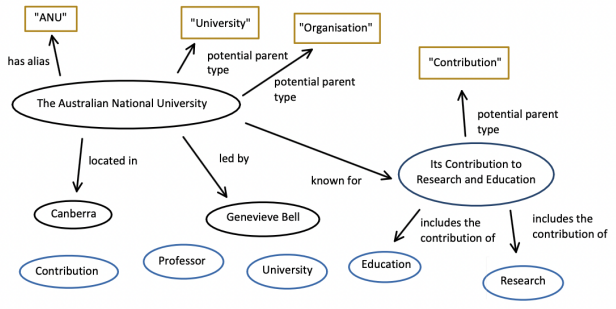


Figure 4: Stage 3 Example

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

4.4 Stage 4: Global Relation Extraction

Algorithm 4 (High-Level)

```
Shrink the document by first reducing its innermost
Section, then progressively shrinking the outer Sections
using Prompt 4.1, until the whole document fits within
the generative LLM's context limit

Embed the shortened document. Embed all the Entities
along with their descriptions.

Compute a relevancy score for each Entity.

For all of the 10% most relevant Entities or the first
20 relevant Entities, whichever 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
the two Entities, extract the **most important**
predicate that relates them, if it exists, and return
in the form of (Subject, Predicate, Object) triple...
```

This stage focuses on extracting Relations that potentially span across the entire paper, e.g., ("This paper", "Concludes", "XXX"). Since an academic paper may exceed the context limit of an LLM, the document is strategically compressed, starting with the innermost Sections and progressing outward until it fits within the LLM's context limit, as shown in Algorithm 4. This is based on the assumption that outer Sections, such as the Introduction, Evaluation, and Conclusion are more important than inner Sections.

Since it can lead to quadratic complexity in extracting the Relations between all Entity pairs, the 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, which aims to prevent overgeneration and reduce irrelevant Relations as LLMs can sometimes be overconfident.

The complexity can be modelled as $O(L + Min(20^2, (0.1 \cdot E)^2)$, where L is the length of the paper and E the number of Entities. The changes in Figure 4 after Stage 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.
```

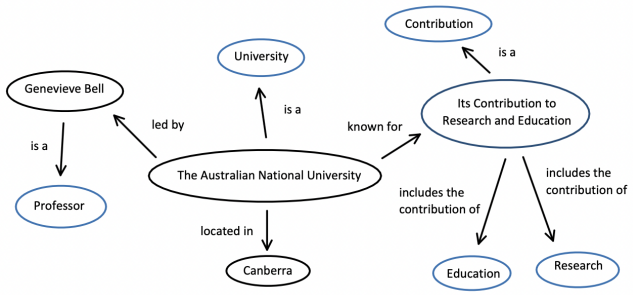


Figure 5: Stage 5 Example

The Taxonomy Generation aim to extract 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 "XX is a YY.", 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 constructs a more complete taxonomic structure, enhancing the expressiveness of the final KG.

This process starts by generating descriptions for all Entities and for their potential parent types found in Stage 1. 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 LLM hallucination.

The process for Predicate Resolution is similar to EL, 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 is approximated as $O(E * P + R)$, where E is the number of Entities, R is the number of triples, and the constant P is the maximum number of potential parent Entities an Entity has.

5 Experiment Setup

The experiment begins by using the pipeline to generate local KGs for a selection of academic papers. These include nine full papers from ASKG [8, 16–18, 22, 24, 26, 27, 29] and one external paper [11], collectively referred to as The Ten-Paper Dataset. Papers in this dataset range in length from 1,156 to 12,113 tokens. Additionally, 100 abstract-only papers from the SciERC dataset [15] are included, 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) [1] or gpt-4o-mini (Context Limit: 128k) [19]. The LLM encoder used for generating embeddings is BAAI–bge-m3 (Context Limit: 8192) [36] for Stage 5 and BAAI–bge-base-en-v1.5 (Context Limit: 512) [4] for all other stages. The only two *independent variables* in the experiment are the input paper and the generative LLM used (LLaMA or GPT). The pipeline's performance is evaluated 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 with respect to papers of different lengths. It also measures the wall-clock runtime (for LLaMA only) or expense in USD (for GPT only) to compare them against the complexity modelled in Section 4. The reason for measuring expense instead of runtime when using GPT through its API is that Internet speed can fluctuate, whereas the expense is approximately proportional to its usage [1].

5.2 Evaluation via Reverse Engineering

The Ten-Paper dataset is unlabelled, meaning that no ground-truth local KG exists for direct comparison. Therefore, the approaches introduced in Section 5.2 and Section 5.3 are designed specifically for unlabelled data.

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 [3].

As shown in Figure 6, a simple KG-to-text system was also implemented for testing purposes in this research. The system processes triples from the local KG in batches, sorted from most relevant to least relevant, and converts them into text segments. Relevancy

[1] https://openai.com/api/pricing/

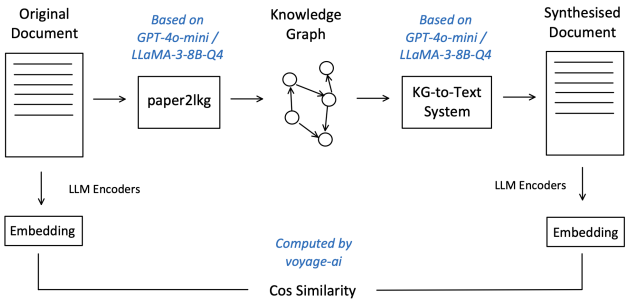


Figure 6: The Process of Evaluation via Reverse Engineering

is calculated as the sum of the relevancies of the two Entities in a triple, where the relevancies of Entities are from Stage 4. The same generative LLM used in the KGC process is used to perform the reverse transformation. The resulting text segments are concatenated into a synthesised document, with more important information appearing earlier. The original document and the synthesised document are then embedded by the LLM encoder, voyage-ai (Context Limit: 30000) [32] and a similarity score is calculated.

This approach reflects both the precision and recall of a constructed KG, as a reversible KG indicates that it captures sufficient and correct information from the original document. However, this method does not assess KG conciseness. A KG without proper EL and Predicate Resolution can still achieve a very high result because these processes primarily organise and refine existing information rather than extract new information. Another drawback is that it evaluates both the KGC and KG-to-text systems together, making it difficult to isolate the performance of each system individually. While embeddings serve as effective representations of documents, they are inherently lossy, which may impact the accuracy of the similarity scores used in the evaluation.

5.3 Evaluation via Application

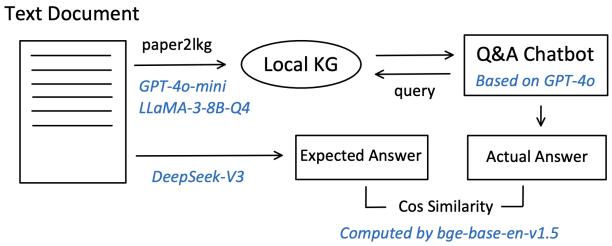


Figure 7: The Process of Evaluation via Application

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. [7]. The generative LLM, DeepSeek-V3 [5], with a context limit larger than any of the documents in the test set, is used to read the whole document directly and generate 10 question-answer pairs as the ground truth, which serves as the role of "Teacher". When generating these question-answer pairs, the LLM is explicitly instructed to create questions that have objective answers and cannot be answered by common knowledge or

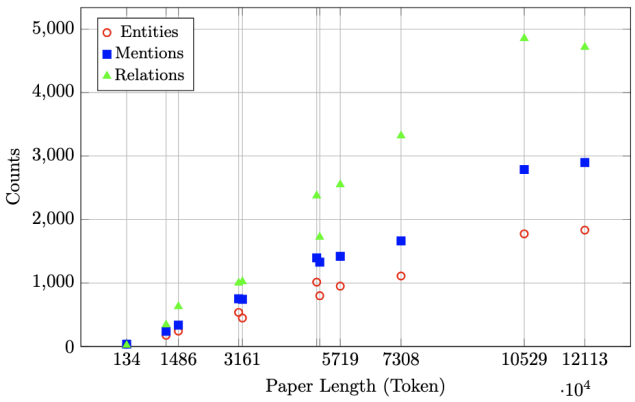


Figure 8: General Metrics of the KGs Constructed from SciERC and The Ten-Papers Dataset using LLaMA

without direct reference to the original document. Meanwhile, the test document is transformed into local KG, mimicking a student learning process. A Q&A chatbot then receives the ten questions from the "Teacher" and attempts to answer them solely using the local KG, leveraging graph-based Retrieval Augmented Generation (RAG). Finally, the RAG-generated answers are compared with the ground-truth answers, and similarity scores are computed to measure the KG's effectiveness in preserving key information.

A simple Q&A chatbot utilising graph-based RAG was developed for this evaluation. It works by first extracting Entities in the questions through prompting the generative LLMs, gpt-4o [19], then querying the KG for any one-hop triples that match the extracted Entities, and finally using gpt-4o again to generate an answer with the assistance of the retrieved triples. Different LLMs are deliberately involved in the "Teacher" and "Student" sides to ensure that generated questions are not tailored to be easily answerable by the same model.

6 Experiment Results

For brevity, this section only presents the key findings of each evaluation approach. The complete experiment results, as well as a 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 general metrics of the local KGs constructed from the 100 abstract-only papers in SciERC and 10 full papers in the Ten-Paper dataset, using LLaMA. For SciERC, the average result for each metric is represented in Figure 8 as a single data point rather than individual sample results.

From Figure 8, it was observed that the number of Mentions increases at a faster rate than the number of Entities as paper length grows. This indicates that multiple Mentions are, to some extent, grouped into the same Entity during the EL stage. Additionally, the number of Relations exhibits the most rapid growth, as the maximum possible connections between Entities scale quadratically with the number of Entities.

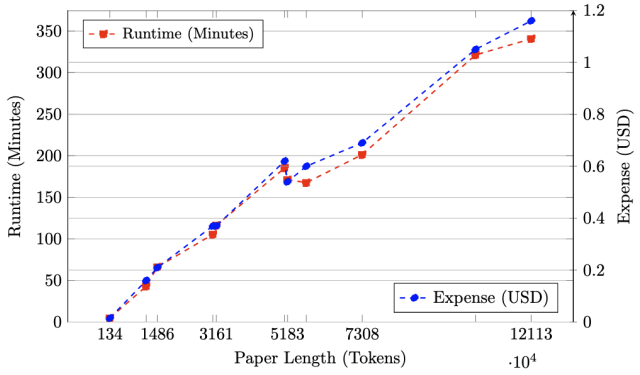


Figure 9: Wall-Clock Runtime (LLaMA) and Expense (GPT) to Construct KGs from SciERC and The Ten-Paper Dataset

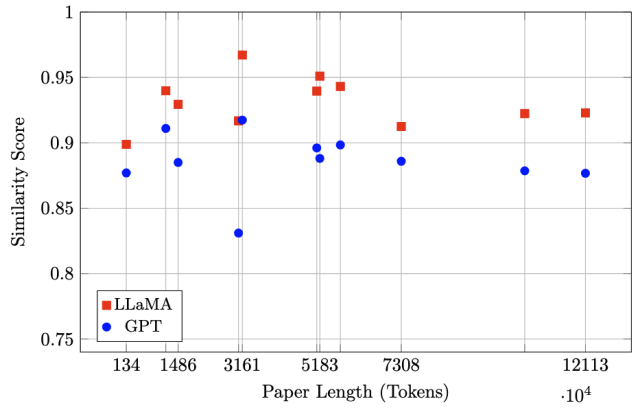


Figure 10: Reverse Engineering Test Scores vs. Papers of Different Lengths Generated by Different LLM Models

Figure 9 shows the wall-clock runtime and expense for processing papers of different lengths. Both runtime and cost are bounded linearly as paper length grows, aligning with the complexity estimates in Section 4, where all stages exhibit linear complexity with respect to either paper length (L), Mentions (M), Entities (E), or Relations (R). And from Figure 8, all M, E, and R appear not to exceed linear growth with paper length, or at most, deviate slightly within the limited dataset.

Despite achieving linear complexity, the pipeline remains computationally expensive. Processing a 5000-token paper can take up to 200 minutes, which is still inefficient in practical scenarios. The high proportional constant is mainly due to the use of generative LLMs, which are intrinsically resource-intensive.

6.2 Reverse Engineering and RAG Test

Figure 10 presents the similarity scores between the original and synthesised documents when using different models and papers of different lengths. The results show that the overall similarity score across the current datasets and models is relatively high, around 0.9. Additionally, the graph does not exhibit a noticeable decline

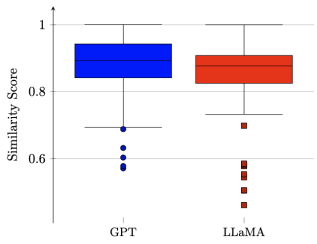


Figure 11: Boxplots of RAG Test Scores for KGs Generated from the Ten-Paper Dataset Using LLaMA and GPT

in similarity scores as paper length increases, suggesting that the pipeline may generalise well to longer papers.

Only the Ten-Paper dataset is used in the RAG test. A boxplot showing the distribution of similarity scores between the RAG answers and the true answers (10 * 10 samples in total) for KGs generated by each LLM model is shown in Figure 11. The average RAG score is 0.88 for KGs generated by GPT and 0.85 for those generated by LLaMA. Although most RAG answers are similar to the expected answers, the outliers present in Figure 11 indicate that the RAG system has failed to answer a few questions using the local KGs, which suggests that the output KG has some information loss.

7 Conclusion and Future Work

This research has introduced a pipeline capable of converting academic papers into local KG representations using LLMs. Evaluation results suggest that the generated KGs can retain information to a relatively high extent and may be useful for a graph-based RAG system in answering users' queries. However, the high computational cost of generative LLM calls remains a challenge when integrating generative LLMs in KGC.

In relation to the Research Goals outlined in Section 3, Goal 1 has been 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, the quality of the generated KGs requires further refinement, given the need for high accuracy in academic content. The evaluation results demonstrate the efficacy and efficiency of generative LLMs in KGC, though these findings are based on a limited dataset and should be considered preliminary.

Future work can be divided into two directions: 'scaling up' and 'scaling out'. Scaling up means improving the performance and efficiency of the pipeline itself, such as enhancing prompt designs and algorithms. "Scaling 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. Additionally, a dedicated graph-based RAG system could be implemented, where distributed agents process individual local KGs while a central master agent integrates insights to generate more 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] Meta AI. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI] https://arxiv.org/abs/2407.21783

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

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

[4] 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. doi:10.18653/v1/2024.findings-acl.137

[5] DeepSeek-AI. 2025. DeepSeek-V3 Technical Report. arXiv:2412.19437 [cs.CL] https://arxiv.org/abs/2412.19437

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL] https://arxiv.org/abs/1810.04805

[7] 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. doi:10.48550/arXiv.2404.16130 arXiv:2404.16130 [cs.CL]

[8] Armin Haller, Axel Polleres, Daniil Dobriy, Nicolas Ferranti, and Sergio José Rodríguez Méndez. 2022. An Analysis of Links in Wikidata. In *Extended Semantic Web Conference*. https://api.semanticscholar.org/CorpusID:249318527

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

[10] 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. doi:10.1145/3447772

[11] 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. doi:10.1109/UBMYK48245.2019.8965559

[12] 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

[13] Runsong Jia, Bowen Zhang, Sergio José 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

[14] Fei Li, Zhichao Lin, Meishan Zhang, and Donghong Ji. 2021. A Span-Based Model for Joint Overlapped and Discontinuous Named Entity Recognition. arXiv:2106.14373 [cs.CL] https://arxiv.org/abs/2106.14373

[15] 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. doi:10.18653/v1/D18-1360

[16] Sergio José Rodríguez Méndez and John Kar-Kin Zao. 2018. BCI Ontology: A Context-based Sense and Actuation Model for Brain-Computer Interactions. In *SSN@ISWC*. https://api.semanticscholar.org/CorpusID:52303174

[17] Pouya Ghiasnezhad Omran, Kerry Taylor, Sergio José Rodríguez Méndez, and Armin Haller. 2022. Learning SHACL shapes from knowledge graphs. *Semantic Web* 14, 1 (26 Sept. 2022), 101–121. doi:10.3233/SW-223063 Publisher Copyright: © 2023-The authors. Published by IOS Press..

[18] Pouya Ghiasnezhad Omran, Kerry Taylor, Sergio José Rodríguez Méndez, and Armin Haller. 2022. Active Knowledge Graph Completion. *Inf. Sci.* 604, C (Aug. 2022), 267–279. doi:10.1016/j.ins.2022.05.027

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

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

[21] 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. doi:10.1109/TKDE.2024.3352100

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

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

[24] Madhawa Perera, Armin Haller, Sergio José Rodríguez Méndez, and Matt Adcock. 2020. HDGI: A Human Device Gesture Interaction Ontology for the Internet of Things. In *The Semantic Web – ISWC 2020: 19th International Semantic Web Conference, Athens, Greece, November 2–6, 2020, Proceedings, Part II* (Athens, Greece). Springer-Verlag, Berlin, Heidelberg, 111–126. doi:10.1007/978-3-030-62466-8_8

[25] 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. doi:10.1109/ijcnn60899.2024.10651286

[26] Sergio José 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. doi: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.

[27] Sergio José 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.

[28] Muhammad Salman, Haoting Chen, Sergio José Rodríguez Méndez, and Armin Haller. 2024. An LLM-SPARQL Hybrid Framework for Named Entity Linking and Disambiguation to Wikidata. In *the 18th China Conference on Knowledge Graph and Semantic Computing and the 13th International Joint Conference on Knowledge Graphs (CCKS-IJCKG) 2024*.

[29] Sandaru Seneviratne, Sergio José 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. doi:10.1145/3460210.3493550

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

[31] 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

[32] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An Open-Ended Embodied Agent with Large Language Models. arXiv:2305.16291 [cs.AI] https://arxiv.org/abs/2305.16291

[33] 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

[34] 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

[35] 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

[36] 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. doi:10.1145/3626772.3657878

[37] 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. doi:10.18653/v1/2023.emnlp-main.493

[38] Bowen Zhang, Sergio José 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.

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

[40] 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. doi:10.1145/3618295