ITERATIVE ZERO-SHOT LLM PROMPTING FOR KNOWLEDGE GRAPH CONSTRUCTION

Salvatore Carta
Department of Mathematics
and Computer Science
University of Cagliari
Palazzo delle Scienze, Via Ospedale,
72, 09124, Cagliari, Italy
salvatore@unica.it

Alessandro Giuliani,
Department of Mathematics
and Computer Science
University of Cagliari
Palazzo delle Scienze, Via Ospedale,
72, 09124, Cagliari, Italy
alessandro.giuliani@unica.it

Leonardo Piano
Department of Mathematics
and Computer Science
University of Cagliari
Palazzo delle Scienze, Via Ospedale,
72, 09124, Cagliari, Italy
leonardo.piano@unica.it

Alessandro Sebastian Podda
Department of Mathematics
and Computer Science
University of Cagliari
Palazzo delle Scienze, Via Ospedale,
72, 09124, Cagliari, Italy
sebastianpodda@unica.it

Livio Pompianu
Department of Mathematics
and Computer Science
University of Cagliari
Palazzo delle Scienze, Via Ospedale,
72, 09124, Cagliari, Italy
livio.pompianu@unica.it

Sandro Gabriele Tiddia
Department of Mathematics
and Computer Science
University of Cagliari
Palazzo delle Scienze, Via Ospedale,
72, 09124, Cagliari, Italy
sandrog.tiddia@unica.it

ABSTRACT

In the current digitalization era, capturing and effectively representing knowledge is crucial in most real-world scenarios. In this context, knowledge graphs represent a potent tool for retrieving and organizing a vast amount of information in a properly interconnected and interpretable structure. However, their generation is still challenging and often requires considerable human effort and domain expertise, hampering the scalability and flexibility across different application fields. This paper proposes an innovative knowledge graph generation approach that leverages the potential of the latest generative large language models, such as GPT-3.5, that can address all the main critical issues in knowledge graph building. The approach is conveyed in a pipeline that comprises novel iterative zero-shot and external knowledge-agnostic strategies in the main stages of the generation process. Our unique manifold approach may encompass significant benefits to the scientific community. In particular, the main contribution can be summarized by: (i) an innovative strategy for iteratively prompting large language models to extract relevant components of the final graph; (ii) a zero-shot strategy for each prompt, meaning that there is no need for providing examples for "guiding" the prompt result; (iii) a scalable solution, as the adoption of LLMs avoids the need for any external resources or human expertise. To assess the effectiveness of our proposed model, we performed experiments on a dataset that covered a specific domain. We claim that our proposal is a suitable solution for scalable and versatile knowledge graph construction and may be applied to different and novel contexts.

1 Introduction

Nowadays, the world is experiencing an unprecedented transformation since the development of recent generative large language models (LLMs). The capability of such AI systems, like GPT-3.5, in analyzing, processing, and comprehending a considerable amount of human language leads to rapid and strong improvements in several application fields. LLMs are changing the way of accessing, ingesting, and processing information. Indeed, they can enable humans to interact easily with machines, which can receive and understand the users' queries expressed in natural language and generate coherent, reliable, and contextually relevant responses represented in a proper natural language format. The impact of LLMs transcends pure language processing. Indeed, they may lay the foundations for a not-distant future where the interaction and cooperation between humans and machines represent the primary strategy for innovation and progress. In this scenario, the potential of LLMs may revolutionize all real-world domains, e.g., technological applications, healthcare, or creative industries like music composition or even art. Like classical Machine Learning algorithms, which are trained on large datasets to provide predictions given a specific input [21], LLMs are trained on vast written language datasets to predict natural language elements following a given input textual query (i.e., a *prompt*), such as a question, instruction or statement. LLMs can generate a remarkable variety of outputs depending on the given prompt. Therefore, LLM prompting assumes a central role, and researchers are ever more focused on exploiting their potential to devise innovative algorithms and tools and improve knowledge in various domains. LLM prompting can be leveraged to enhance various Machine Learning tasks, as the principal strength is the ability to generate high-quality synthetic data reducing the manual effort in collecting and annotating data. Indeed, by composing well-formulated prompts, LLMs can be highly valuable in supporting the devising of models in many scenarios, e.g., in generating creative content (like stories or poems) [20], Information Retrieval (where users can request information on specific topics) [5], problem-solving [28], or text summarization [29].

Notwithstanding the general encouraging performance of LLMs, their usefulness is still not optimal or under investigation in several areas. In this context, this paper aims to provide an innovative approach to knowledge representation, which plays an essential role in many real-world scenarios, transforming how the information is collected, organized, processed, and used. In particular, we focus on a novel method for creating suitable Knowledge Graphs (KGs), that have been demonstrated to be extremely valuable across several industries and domains. KGs organize information in a proper graph structure, where nodes represent entities and edges represent relations between entities. Each node and edge may be associated with properties or attributes, providing further details or metadata. KGs are powerful tools for capturing and representing knowledge appropriately that hold considerable advantages:

- they can infer and integrate information from heterogeneous sources, e.g., structured databases, unstructured text, or web resources;
- they can capture both explicit and implicit knowledge. Explicit information is directly represented in the KG (e.g., "Joe Biden is the president of the USA"), whereas the implicit knowledge can be inferred by exploring the graph and finding relevant patterns and paths, allowing to discover new insights;
- they allow a straightforward and effective query and navigation of information.

KGs are widely applied in many domains, e.g., in recommender systems [8], semantic search [27], healthcare and clinical tools [1], or finance [12].

Although significant improvements have been made in graph construction, creating a complete and comprehensive KG in an open-domain setting involves several challenges and limitations. For example, the lack of established large annotated datasets for relation extraction, which arises from the absence of a rigorous definition of what constitutes a valid open-domain relational tuple, leads to the development of relation extraction tools using heterogeneous datasets, which tend to work well on their trained dataset but are prone to produce incorrect results when used on different genres of text [19]. Moreover, the open-domain implementations of the complementary tasks of Named Entity Recognition and Entity Resolution are understandably less performing than their closed-domain counterparts, and they also suffer from the lack of well-established toolkits and comprehensive knowledge bases.

To overcome the aforementioned limitations, our insight is to rely on LLMs to support the construction of KGs. Indeed, their ability to analyze and generate human-like text at a large scale can be instrumental in knowledge representation. We deem that LLMs can enhance the KG generation in the main stages of the process, e.g., in extracting entities and relations, disambiguation, or textual inference. In particular, relying on LLM prompting may be a key point of KG generation. By composing an appropriate prompt, LLMs can efficiently process structured and unstructured text and transform it into KG components. A well-formed prompt can lead to extract relevant entities, relationships, and types.

Summarizing, we present a novel approach for KG construction based on extracting knowledge with the support of an iterative zero-shot (i.e., without the need for any examples and fine-tuning) LLM prompting. In particular, a sequence of appropriate prompts is used iteratively on a given set of input documents to extract relevant triplets and their attributes

for composing a KG. Subsequently, supported by a further prompting strategy, we defined a proper entity/predicate resolution method for resolving the entity/relations co-references. Finally, a different prompting approach is applied to develop an inference-based technique for defining a meaningful schema. To our knowledge, although using LLMs is a hot research topic, this is the first attempt at using them to develop a suitable approach for creating a complete KG.

The rest of the paper is organized as follows: Section 2 reports the Knowledge Graph generation state-of-the-art. Section 3 summarizes our claim and research goals, whereas the proposed methodology is described in Section 4. Section 5 is aimed at reporting all the experimental results and the related discussion. Section 6 ends the paper with the conclusions.

2 Related Work

Automatic knowledge graph construction aims to create a structured knowledge representation from different data sources without manual intervention. Knowledge Graph Construction (KGC) pipelines typically employ Named Entity Recognition [7], Relation Extraction [18], and Entity Resolution [9] techniques, in order to transform unstructured text into a structured representation that captures the entities, their relationships, and associated attributes. The pipeline of [10] combined co-reference resolution, named entities recognition, and Semantic Role Labeling to build a financial news Knowledge Graph. Luan et al. [16] developed a multi-task model for identifying entities, relations, and coreference clusters in scientific articles able to support the creation of Scientific Knowledge Graphs. Mehta et al. [17] presented an end-to-end KG construction system and a novel Deep Learning-based predicate mapping model. Their system identifies and extracts entities and relationships from text and maps them to the DBpedia namespace. Although these pipelines can achieve satisfactory results and produce high-quality Knowledge Graphs, their methods are often limited to a predefined set of entities and relationships or dependent on a specific ontology. Our proposal addresses these limitations, as we do not rely on predefined sets or external ontologies. Additionally, almost all methodologies exploit a supervised approach, requiring extensive human manual annotation. To address this issue, since the introduction of the first Pre-Trained Language Models, the question has been: *can we use the knowledge stored in pre-trained LMs to construct KGs?* Wang et al. have been among the first to raise and address such a question. They designed an unsupervised approach called *MAMA* that constructs Knowledge Graphs with a single forward pass of the pre-trained LMs over the corpora without any fine-tuning [24]. Hao et al. [11] proposed a novel method for extracting vast Knowledge Graphs of arbitrary relations from any pre-trained LMS by using minimal user input. Given the minimal definition of the input relation as an initial prompt and some examples of entity pairs, their method generates a set of new prompts that can express the target relation in a diverse way. The prompts are then weighted with confidence scores, and the LM is used to search a large collection of candidate entity pairs, followed by a ranking that yields the top entity pairs as the output knowledge. Recent technological and scientific advancements accompanied by increasing data availability have led to a severe escalation in developing Large Language Models. New LLMs such as GPT-3.5 have shown remarkable zero and few-shot Information Extraction capabilities, as demonstrated by Agrawal et al. [2] and Wei et al. [25]. Li et al. [15] systemically assessed ChatGPT performance in various IE tasks. They pointed out that ChatGPT performs very well on Open Information Extraction and other simple IE tasks (e.g., entity typing) but struggles with more complex and challenging tasks such as Relation Extraction and Event Extraction. Wan et al.have resolved this issue [23], identifying an in-Context Learning strategy to bridge the gap between LLMs and fully-supervised baselines reaching SOTA results in diverse literature RE datasets. Ashok and Lipton exploited LLM to provide an alternate way to approach the few-shot NER making it easily adjustable across multiple domains [3]. Their method, PromptNER, prompts an LLM to generate a comprehensive list of potential entities, accompanied by explanations that support their compatibility with the given definitions for entity types. Similarly, Ji introduced VicunaNER [13], a zero/few-shot NER framework based on the open-source LLM, Vicuna. Trajanoska et al. proposed an improved KGC pipeline supported by LLM to organize and connect concepts [22], prompting ChatGPT to extract entities and relationships jointly and then preprocessing entities with an entity-linking strategy, where the entities having the same DBpedia URI are considered conceptually equal. The authors evaluated their system on a sustainability-related text use case. Nevertheless, their method has a crucial weakness, i.e., as it relies on an external knowledge base, many relevant entities not included in DBpedia are missed. Our approach, compliant with their method in some steps, aims to overcome such drawbacks. Finally, Bi et al. proposed an innovative LLM-powered KGC method [4]. As LLMs have demonstrated excellent abilities in structure predictions, they leveraged GPT-3.5 code generation to create a Knowledge Graph by converting natural language into code formats. Converting natural language to code formats could make the model better capture structural and semantic information. Using structural code prompts and encoding schema information, they aim to enhance the modeling of predefined entity and relation schemas in knowledge graphs. In accordance with this work, our approach proposes an innovative LLM-based schema generation, as described in the following Sections.

3 Research Aims and Motivations

In this Section, we report the motivations and the research aims, starting by describing the open challenges in the scenario of KG generation.

3.1 Problem statement

The current solutions in constructing KGs, either domain-specific or general-purpose, address the task from many perspectives. This is also due to the need to discover the right tradeoff between providing high data quality, scalability, and automation [26]. The variety of state-of-the-art approaches is a clear indicator of the intrinsic complexity of the problem. Therefore, many open challenges are currently leading to a demand for more research efforts. In particular, several factors affect the process of building a KG:

- *Data availabity and acquisition*. Accessing and adopting relevant and suitable data sources is still challenging. One of the main goals of a KG is to capture an exhaustive range of knowledge from various domains and sources, either structured or unstructured. However, the availability of data is typically hampered by several factors. For example, a common scenario is when data sources are restricted or inaccessible due to policies or privacy concerns, or data may be strewed across different repositories, platforms, formats, or even languages, making their integration very challenging. Furthermore, data is highly dynamic nowadays, with the risk of existing data becoming obsolete. Therefore, the current challenge is to put extensive effort into data acquisition, monitoring, and updating.
- *Data quality*. The acquired data should be accurate, complete, consistent, and reliable to build and maintain a KG properly. Indeed, the quality of such data highly affects the effectiveness and reliability of KGs. Addressing data issues like incorrect or outdated information, insufficient or missing data, unreliable sources, or contradictory data is crucial in KG generation. To ensure richness and accuracy, a classical approach is to involve humans in annotating and labeling data. Leveraging human knowledge allows for a better understanding and context-specific interpretation of the input data. However, this process may not be affordable as it heavily depends on the availability of knowledgeable annotators and their time and effort. Therefore, guaranteeing reliable data and limiting human efforts, especially in the case of unstructured text data, is crucial.
- *Scalability*. Providing and maintaining a high-quality KG, also in the case of automatic data acquisition and integration, is a complex goal due to the data dimensionality and heterogeneity of data sources. Indeed, scalability is crucial in a real-world scenario where an enormous amount of data must be analyzed. In general, the more the graph grows, the more the computation requires resources and time consumption.
- *Subjectivity and contextual knowledge*. Although KGs mainly focus on factual information, representing subjective or context-dependent knowledge is extremely challenging and often requires further contextual information or external resources;
- *Semantic disambiguation*. Natural language is intrinsically ambiguous. Semantic disambiguation, i.e., disambiguating words, resolving synonyms, handling polysemous terms, and capturing fine-grained unlikenesses in meaning, are current research challenges.
- *Domain-specific expertise*. Different domains may retain complex and specialized knowledge structures, demanding domain expertise and more specialized strategies for knowledge inference. Domain experts may ensure quality and effectiveness but usually require a significant human effort, as already pointed out.
- *External resources*. The current methods of KG generation often rely on *entity linking*, i.e., resolving entities and relations to external knowledge base (KB) concepts. The main weakness of such methods is the high risk of filtering out many relevant entities or concepts not included in the reference KB. On the other hand, a common way to extract triplets without relying on external knowledge is the adoption of Open Information Extraction (OpenIE), which discovers a wide range of triplets without prior knowledge. However, OpenIE methods usually generate a high number of incorrect or incomplete triplets; this behavior leads to including incorrect and misleading information in the KG.
- *Evaluation*. Evaluating a KG is crucial to assess whether the generated KG properly represents the knowledge of the underlying scenario. However, evaluation is a challenging task due to several issues. First, there are often no ground truths or golden standards to adopt for comparisons. Furthermore, there are no standard metrics, and evaluating general-purpose methods across different use cases or application fields may be complex.
- *Pipeline definition*. While, typically, most parts of a specific KG generation pipeline are well-known methods, algorithms, or models, with exhaustive previous research, the interaction and the integration of all tasks is a current challenge.

3.2 Research questions

According to the aforementioned challenges and limitations, this work proposes an approach to address various issues in KG construction. In particular, we intend to answer to the following research question:

- How can we effectively extract, analyze and enhance information from multiple textual data sources?
- How can we improve the quality of the extracted information avoiding the need for human effort, also in case of generating KGs that usually require specific human expertise?
- How can we generate relevant and proper triplets without relying on external KBs or OpenIE methods?
- Regarding scalability, which strategies can we define for dealing with large-scale datasets for generating KGs with millions or even billions of entities and relationships?
- What methods can be devised to adequately perform disambiguation, entity resolution, and linking to properly represent the knowledge?
- How can we properly evaluate the generated KG also without priorly having a gold standard or a specific ground truth?

3.3 Main contribution

The previous research questions have guided the investigation of innovative methods for proposing proper strategies to address the main problems and open challenges. Let us note that, in this work, we focus on extracting information from heterogeneous unstructured textual documents. All our solutions have been incorporated into a novel pipeline for generating KGs, described in Section 4, that (i) relies on adopting an LLM that is iteratively queried with a sequence of adequately well-formed prompts to perform the main stages of the KG generation and (ii) can effectively perform an automated entity/predicate resolution without the need of external knowledge-bases or any human effort. To our knowledge, at the state-of-the-art, no other KG generation models rely on using our LLM prompting strategy in combination with an automated and knowledge-base agnostic entity resolution for a complete KG generation pipeline. Let us point out that in this preliminary work, we have assumed GPT-3.5 as LLM.

The main contribution of the paper is summarized in the following:

- we propose an iterative LLM prompting-based pipeline for automatically generating KGs. Let us remark that there is no need for any human effort;
- we propose a sequence of proper, well-formed LLM prompts for each stage of the process. The devised prompts are able to:
  - identify relevant entities and extract their descriptions and type;
  - identify meaningful relationships and their descriptions;
  - given the previous components, identify relevant triplets;
  - provide domain-specific triplets, even if no information about the domain is given as input;
  - resolving entities and predicates in an automated and reliable way, without relying on any third-party resources.
- we propose a "zero-shot" approach, as all the devised prompts do not need any examples or external KBs for inferring the related information.
- our proposal may deal with large-scale data, as no human effort and examples documents are required;
- we exploit the outcome of several prompts for manually building a ground truth, useful to apply additional evaluation metrics.

4 Methodology

This Section describes our methodology, starting from the general perspective, introducing the main key points and functionalities, and, subsequently, detailing the pipeline step by step.

4.1 LLM Prompting

The main focus of our innovative approach is the integration of task-specific prompting of an LLM in each step of the KG construction. To this end, we first investigate, in this work, the capability to analyze, elaborate, and generate

human-like text of a trendy, well-known LLM, i.e., GPT-3.5, in particular the GPT-3.5-turbo-0301 release[1]. We access the GPT-3.5 capabilities using the official chat completion API[2]. As described in the API references, we interact with the model submitting the chat as a list of messages, each with a specified role, and we get a response coherent with the conversation thread. There are four supported roles, but we are only focused on the following:

- *system*: the role which aims to tell the model how to behave in its responses;
- *user*: the role of the user messages/requests;
- *assistant*: the role of the model responses.

In our case, we pass the detailed task instructions as a system prompt and the data to operate on as an appropriately formatted user prompt. We receive the task results in a message with the assistant role. Furthermore, we aim to obtain a deterministic behavior of the model (i.e., to get the same message in response to the same input prompts) by setting the API *temperature* parameter to zero.

4.2 Methodology Overview

The main process of the proposed approach can be represented by the high-level architecture in Figure 1, which describes the main stages of the process, and embodies three main tasks: *Candidate Triplet Extraction* (1), *Entity/Predicate Resolution* (2), and *Schema Inference* (3). Such tasks are performed sequentially, and their main functionalities are described in the following sections.

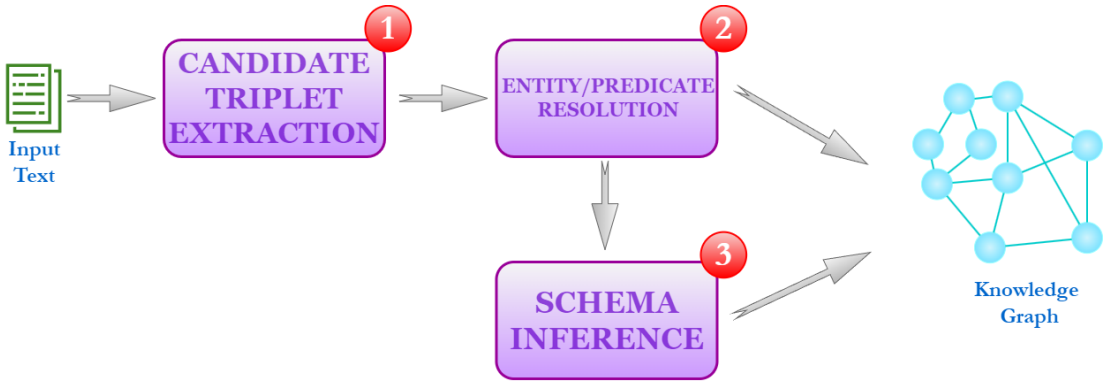

Figure 1: High-level architecture

4.2.1 Candidate Triplet Extraction

As already remarked, LLM prompting can leverage the capability of a model to process a massive amount of information. In this module, the challenge is defining well-formed prompts to effectively generate proper candidate triplets for being included in the final graph. In particular, we first established the following goals:

- **A proper entity characterization**, aiming to explore further the simple identification of text spans representing an entity *mention*, enriching a potential mention with:
  - a representative *entity label*, not necessarily corresponding to an exact text span;
  - a proper entity *description*;
  - a list of *types* or *hypernyms* that denote the entity beyond the mention.
- **An appropriate characterization of triplets and predicates**, representing a relation between two entities (i.e., the *subjet* and the *object*) with a relevant predicate defined by:
  - a suitable label, not necessarily corresponding to an exact text span;
  - a general description of the relationship existing between subject and object.

[1] `https://platform.openai.com/docs/models/gpt-3-5`
[2] `https://platform.openai.com/docs/api-reference/chat/create`

The requirement of an extended entity/predicate description comes out from providing a more informative and exhaustive representation of the extracted concepts, enriching the pure extraction approaches, like classical OpenIE methods, that are typically based on identifying and extracting entities or predicates corresponding to exact text segments included in the given input document. We deem that the description generated by our approach, together with the entity type list, provides a detailed context for each mention, leading us to an actual *semantification*.

4.2.2 Entity/Predicate Resolution

On the one hand, the goal of the entity resolution is the identification and merging of mentions which refers to the same underlying entity. On the other hand, predicate resolution refers to identifying and resolving different expressions or textual forms that convey the identical relation between entities. As the data sources are mainly unstructured nowadays, one of the main challenges in this task is addressing the ambiguity and heterogeneity of natural language expressions without relying on external resources or domain expertise. Furthermore, capturing the complete semantic knowledge of entities or predicates is still complicated, particularly when dealing with domain-specific or complex concepts. To this end, we devised a novel resolution model which combines a semantic aggregation of concepts with a subsequent prompting stage. The former aims to identify and aggregate similar semantic concepts, and the latter aims to identify, in such groups, the concepts having the same meaning and labeling with a unique textual representation. A preliminary clustering is necessary as the current LLMs are more efficient with limited information. Hence, sending all the extracted entities or relations may be less reliable. The method is detailed in Section 4.3.2.

4.2.3 Schema Inference

A KG schema describes the design, organization, and semantics of a given KG. In detail, it defines the types of entities, including their attributes and relationships, acting as a blueprint or a conceptual model for a KG. Providing a KG with an appropriate schema facilitates the reuse, sharing, and comprehension of the graph for humans and machines. Nevertheless, building a schema from scratch is challenging, as, typically, it is often an interactive task that requires domain expertise and feedback by human knowledge. We aim to infer a schema in an automated way without relying on any human support. To this end, we deem that LLM prompting can be helpful also in this stage.

4.3 Knowledge Graph Construction Pipeline

The devised KG generation pipeline is detailed in Figure 2. The Figure provides an in-depth summary of all modules mentioned in the previous Section, described in the following.

4.3.1 Candidate Triplet Extraction

Defining the LLM-prompting approach for extracting well-characterized entities and triplets requires decomposing the problem into simpler tasks resolvable by GPT-3.5 and studying the most effective prompts to solve them. To this end, we performed an exploratory stage to find an optimal prompting strategy: the task decomposition and prompt definition activities have been iterated in a trial-and-error method until each task performed with adequate reliability and accuracy. We started by using the official guidelines[3] as a reference and gradually leveraged the experience gained during the various trials to finally define the current candidate triplet extraction phase.

The devised triplet extraction approach begins with a pre-processing stage (*text split*) designed to reduce overly long texts into smaller chunks to process for the extraction by GPT-3.5. Inspired by what might be an intuitive and systematic way of approaching the same extraction task by a human annotator, the subsequent extraction stage from the text chunks decomposes into two phases, in which we first identify the entities in a text chunk (*entity extraction*) and then check how they relate to each other to find the actual triplets (*iterative triplet extraction*). We deem that this is a more natural and straightforward process than directly starting with the triplet extraction, in which the search for entities is implicit and adds to the complexity of identifying the relationships, also simplifying the overall task and dealing with the difficulty of GPT-3.5 in reliably performing long and complex operations. The iterative strategy addresses the additional struggle of GPT-3.5 in respecting some constraints, which in this case means explicitly referencing the original list of entities when searching for triplets.

We cover the details of the presented problem decomposition in the following sections.

[3] `https://platform.openai.com/docs/guides/gpt-best-practices`

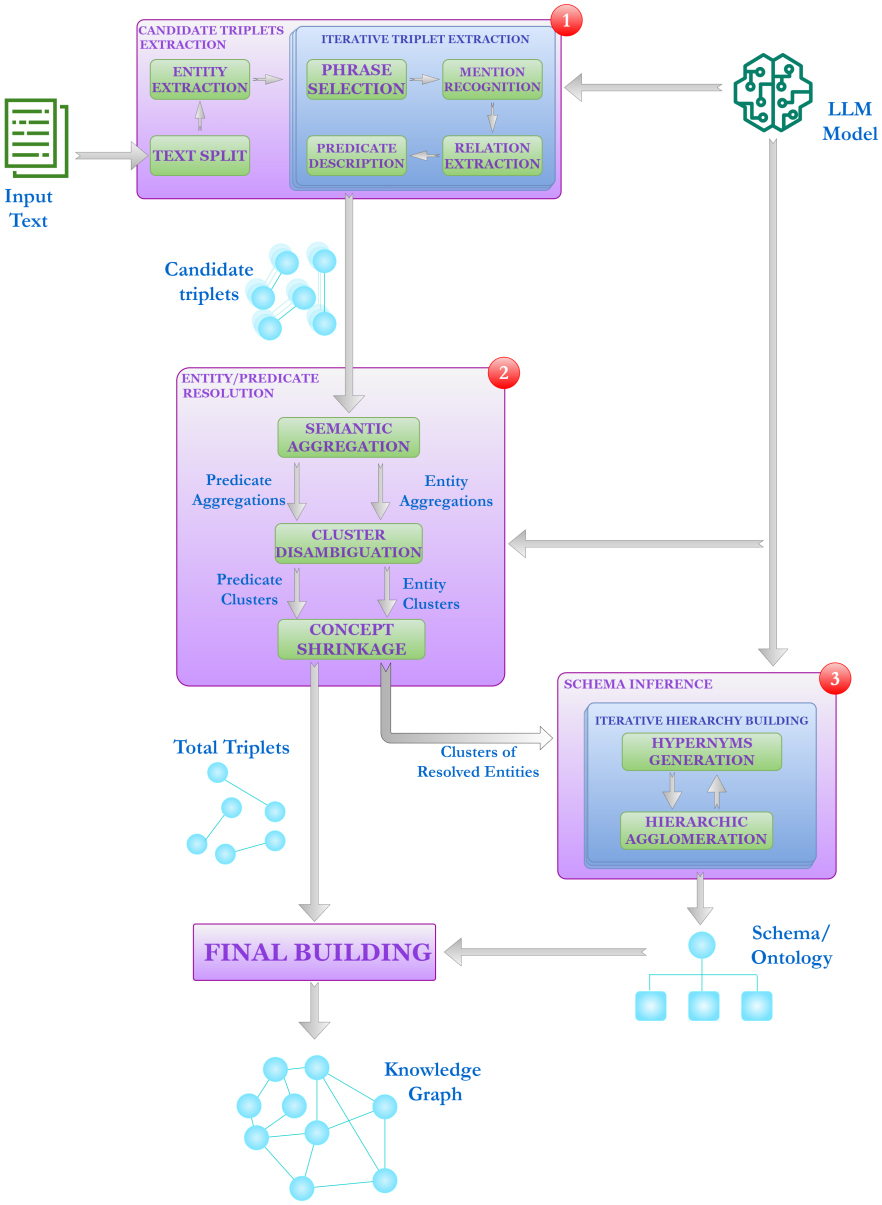

Figure 2: KG generation pipeline

Text Split

Generative LLMs have a limit on the number of tokens they can process due to architectural and computational constraints. The specific GPT-3.5 has a limit of 4,096 tokens (defined by the cl100k tokenizer[4]), shared by both the received input and the generated output. Because of this technical limit, entity, and triplet extraction from long texts[5] must be performed in several steps to ensure the token limit is never reached.

Splitting the text into several independent chunks is a non-trivial necessity that brings to addressing two major problems:

[4] `https://github.com/openai/openai-cookbook/blob/main/examples/How_to_count_tokens_with_tiktoken.ipynb`

[5] We refer to long texts that are not decomposable into several stand-alone text excerpts but consist of a single unique narrative thread.

- The **omission of the global context**, since each independent chunk use terms and expressions that may lose their meaning when extrapolated from the overall narrative of the full text.
- The **separation of related entities**, as two entities that are members of a triplet may fall into two separate chunks, making it impossible to recover their relationship.

We tackled the two problems by designing a splitting technique that aims to:

- **Avoid the loss of context** by providing each chunk with a summary of the preceding chunks to serve as a global context. Since the summarization task also faces the token limits, the summary for $chunk_i$ (i.e the summary of all chunks from 1 to $i-1$ and that we call $summary_i$) is generated using $chunk_{i-1}$ and $summary_{i-1}$ as input, as formally stated in the formula:

$$summary_i = summarization\_task(summary_{i-1}, chunk_{i-1})$$

  Since we process the chunks in order when we generate $summary_i$ we already have both of the necessary inputs as they have been computed with the extraction on $chunk_{i-1}$.
- **Reduce the probability of separating two related entities** by defining chunks as partially overlapping sliding windows of the input text: the larger the chunk size and overlap, the lower the probability of separating the two related entities.

Entity Extraction

The extraction task begins by identifying entity mentions in a text chunk. On the one hand, we guided GPT-3.5 using a detailed system prompt that respects this overall specifics:

S1) labelitem:An explanation of what is meant by an entity;

S2) An explicit request to retrieve the entity mentions in the user's text, specifying the additional requirement of providing a description and a list of types for each;

S3) The details for a properly formatted output.

The text from which to extract the entities is integrated within the user's prompt.

While S2 explains what task will be performed and S3 establishes a standard format for the output, the actual role and functionality of S1 are less immediate. S1 is not mandatory for the actual execution of the task since GPT-3.5 already "knows" what an entity is in the context of knowledge graphs and can find entity mentions even without our specifics. However, there is a notable difference in the amount and significance of the results extracted when we explicitly provide GPT-3.5 with our description of what we mean by an entity. Including S1 decreases the number of entities retrieved but leads to a greater focus on concrete nouns and named entities better detailed in the text chunk, leaving general and abstract nouns to serve mainly as types. Moreover, when we exclude S1 from the prompt, we observe less detailed entity descriptions, which we can especially appreciate by comparing the entities commonly found with and without S1.

For example, we refer to a few entities extracted from the use case texts introduced in Section 5. From the textual content of "Cagliari" webpage, we extract 27 entities with respecting the S1 constraint and 62 when excluding it. However, the extra entities are mostly general concepts such as "History and art", "Sea and parks", "Sea view", "Shopping streets", and many others, which are not well covered in the text. Generating these entities has a negative effect on the descriptions themselves, which cannot be detailed. Furthermore, this negative impact also affects the entities that are properly described: thus, the description of "Cagliari" passes from "The capital city of Sardinia, offering history, art, seashores, parks, and fine cuisine" with S1 to "The capital of Sardinia"" without S1; the description of "Marina District" changes from "A quarter in Cagliari, featuring lovely buildings and the porticos of Via Roma, including the Palazzo Civico" to "The waterfront district of Cagliari".

Because of this observed behavior, we evaluated the contribution of S1 as positive and maintained it in our implementation and experiments. We denote as E the set of entities extracted at this phase.

Iterative Triplet Extraction. In this phase, we use the set of entities E as a constraint to extract the relations between the entities from the text chunk content T in the form of triplets. Extracting triplets that explicitly refer to the set of entities E poses a challenging requirement for GPT-3.5. When set E is highly populated, GPT-3.5 fails at sticking to the complete set and starts generating triplets involving entities not in E. Even if some of these triplets can be considered an expression of correct statements, they are certainly not very interesting since they include entities for which we don't have a description or a list of types, which are an essential requirement of our approach.

To extract triplets that refer only to entities in E, we designed an iterative process in which we extract triplets by focusing on a different entity $e_i \in E$ at each iteration i. The core concept of each iteration is to simplify the task

complexity at every step, making it easier to select only entities in E. We discuss the steps in each iteration in the following paragraphs.

Phrase Selection

To focus on e_i and avoid using the whole text T, which might contain unrelated statements only about other entities, we aim to extract a target text excerpt T_i^G for the triplet extraction that summarizes the information about e_i present in T. T_i^G defines a smaller context than the complete T, thus reducing the complexity of the next extraction steps and increasing the reliability of GPT-3.5.

The extraction of T_i^G may be compliant with classical *Query-Focused Abstractive Summarization* or *Open-Domain Question Answering* tasks that can be performed again with GPT-3.5. We designed a suitable prompt focusing on the abstraction of simple declarative sentences that provide information about the entity e_i, which we can concatenate to obtain T_i^G. We also identified an alternative solution in which we bypass the explicit extraction of T_i^G and directly select the very same description of e_i already generated with the approach described in Section *Entity Extraction*. The latter solution provides a more concise (but still meaningful) T_i^G than the one obtainable with the former solution but significantly reduces the computation time and cost due to the omitted API calls. The experiment in Section 5 uses the latter solution.

Mention Recognition

The new T_i^G text is a summary focusing on e_i, in which we expect to find mentions of the other entities $e_j \in E$ that are related to e_i and contribute to shaping its identity. We want to detect which entities e_j are mentioned in the generated summary, thus defining the subset E_i of entities mentioned in T_i^G. We expect E_i to have significantly fewer entities than E. To solve this task, we instruct GPT-3.5 with a system prompt requesting to recognize the mentions of the listed entities in the specific text, all provided by the user. We ask GPT-3.5 to rewrite the same list of entities, appending a "yes/no" answer at the end of each entry. The user prompt includes the numbered list of all the entities in E and the generated text T_i^G, appropriately delimited and marked.

Although we demand to refer to the complete set of entities E, we enforce this requirement on a simple task to perform on a small reduced-context T_i^G, which allows GPT-3.5 to do it reliably. Moreover, using a numbered list instead of a simple bulleted list introduces the necessity to preserve the association entity ID/label, which seems to support GPT-3.5 in a correct execution even more. We retrieve the mentioned entities e_j that define E_i by finding the entries marked with a "yes" in the answer.

Relation Extraction

We can perform the relation extraction using the narrowed context constructed in the previous two steps. We query GPT-3.5 with the system prompt for identifying the relations between the listed entities within the text, both supplied by the user. We ask to express the entity relations in the form of RDF triplets, using subjects and objects selected from the list of entities (reporting both their name and ID) and by choosing an expressive predicate. The user prompt provides the text T_i^G and the numbered list of entities in E_i, appropriately delimited and tagged. GPT-3.5 answers with the identified triplets R_i.

The system prompt includes an essential clarification of what we mean by an expressive predicate, improving the overall quality of the extracted triplets. Our explanation points GPT-3.5 into generating predicates that correctly represent the relationship between the two entities without being too specific, as it would make the predicate hardly reusable and observable in other triplets, aiming for a sort of predicate *canonicalization*.

We refer to an extracted triplet to better explain the importance of our addition. Given the following text excerpt:

> ***Cagliari**: the capital of Sardinia is steeped in Mediterranean atmosphere and offers everything you could want from a vacation: history and art, seashores and parks, comfort and fine cuisine. Picturesque historical districts with sea views, elegant shopping streets and panoramic terraces, including the **Bastione di Santa Croce**, a great place for a romantic evening after a fiery sunset.*

Among the entities found with the entity extraction prompts we can find:

- **Cagliari** (City, Tourist Destination): The capital city of Sardinia, offering history, art, seashores, parks, and fine cuisine.
- **Bastione di Santa Croce** (Tourist Attraction, Landmark): A panoramic terrace in **Cagliari**, offering a romantic view of the sunset.

With which we extract the triplet:

- **Cagliari**; *has landmark*; **Bastione di Santa Croce**

According to the provided text, it could be extracted other overly specific predicates like "has panoramic terrace" or "has great place for romantic evening", or other excessively generic ones like "includes". We did not choose these predicates randomly but used explicit terms within the text as most OpenIE tools would do. We deem the choice of a predicate like "has landmark" to be much more meaningful, as it better expresses the inferable relationship between those two entities. We observed the same tendency in the choice of the predicate in the other extracted triplets.

Since both T_i^G and E_i are reasonably simple, GPT-3.5 has proven capable of reliably extracting meaningful triplets while respecting the given list of entities, unlike the naive case where we refer to the complete text T and the full set of entities E.

Predicate Description

To avoid adding complexity to the relation extraction step, we exclude the generation of the predicate description, which we set as one of our main goals, choosing to do it in a final independent step. We use a system prompt to tell GPT-3.5 to return the description of each unique predicate, referencing the text and list of RDF triplets provided by the user. The user prompt provides the text T_i^G and the list of triplets R_i, appropriately delimited and tagged. The model responds with a list of predicate and description pairs.

The complexity that prevented us from generating the description during the relation extraction arises from the need to have not just any description but one capable of capturing the generic nature of the relation expressed by the label of the predicate. We defined this necessity in our system prompt. We refer again to the extracted triplet used earlier as an example:

- **Cagliari**; *has landmark*; **Bastione di Santa Croce**

Simply asking for a predicate description leads to answers like "It expresses that the Bastione di Santa Croce is a landmark in Cagliari", which are tightly related to the specific instance of the relation in which we use that predicate. With our prompt instead, we can get a description like "Expresses a relationship between a place and a landmark located in it", which perfectly fits the predicate and our original intentions. We obtain the same behavior during the description of most of the predicates.

Response validation. Each system prompt specifies to GPT-3.5 the format to respect when generating the response, which is necessary to parse the response and retrieve the results. For the tasks where we expect a list of results, the format details are for a generic line of the list. For the other tasks involving numbered lists of entries to refer to, we designed the output format to enforce the model into reporting both the entry label and number, which seems to increase the reliability of the answers.

According to these prompt standards, we introduce two types of tests on the model answers:

- **Pattern matching** (always) uses regular expressions to check that the answer, or each of its lines, correctly conforms to the format described in the system prompt.
- **Consistency check** (only when we reference a numbered list) verifies that the model preserves the label/ID association, checking if that label corresponds to that numbered entry.

When a response (or one of its lines) doesn't pass all the required tests, we discard it as unparsable or likely to contain false information.

4.3.2 Entity/Predicate Resolution

As already mentioned, the module combines semantic aggregation with proper LLM prompting. Semantic aggregation aims to identify and aggregate related entities/relations in groups containing concepts having similar semantic meanings or referring to a higher-level concept (e.g., "car" and "motorcycle" are related to the concept of a "vehicle"). This preliminary aggregation is needed to split the information and send a sequence of prompts rather than a unique prompt. Indeed, as also reported in the GPT models guidelines[6], complex tasks tend to have higher error rates than simpler tasks. The entity/resolution task is detailed in the following:

Semantic aggregation

The first step is to aggregate all semantically similar entities and do the same with the relations. In detail, we devised a proper aggregation strategy, based on computing two specific similarity scores for entities (S_e) and relations (S_r), both

[6] `https://platform.openai.com/docs/guides/gpt-best-practices/six-strategies-for-getting-better-results`

based on analyzing the contributions of the KG components (entity/relation, description, and type). To this end, we first consider all the pairs combinations for both entities and relations. For each pair (i, j) of entities or relations, we compute the individual contributions as follows:

- *Label similarity*. We computed a similarity score for each entity or relation pair (i, j) as the Levenshtein distance [14] between the two entity labels. We denote as $e_{i,j}$ the similarity between two entities, and with $r_{i,j}$ the similarity between two relations. Then, $e_{i,j}$ and $r_{i,j}$ are normalized in the range $[0, 1]$, where a similarity equal to 1 means the entities are identical.
- *Entity types similarity*. We adopt the same strategy of entity and relation similarity for computing the similarity $(t_{i,j})$ between each type pair (i, j). Let us remark that types are associated with entities only.
- *Description similarity*. As the description is a text segment more complex than a simple entity or type label, we project all entity and relation descriptions in an embedding space, and we compute the similarity between two descriptions i and j, relying on a classical cosine similarity metric. In detail, we adopt the Universal Sentence Encoder model [6] as the embedding model, and we denote as $ed_{i,j}$ the similarity between two entity descriptions, and with $rd_{i,j}$ the similarity between two relation descriptions.

The final similarities scores are, in both cases, a weighted combination of the previous contributions and are summarized by the following formulas:

$$S_e(i, j) = \alpha \cdot e_{i,j} + \beta \cdot ed_{i,j} \tag{1}$$

$$S_r(i, j) = \gamma \cdot e_{i,j} + \delta \cdot rd_{i,j} \tag{2}$$

We empirically fixed the coefficient values, choosing $\alpha = 0.35$, $\beta = 0.65$, $\gamma = 0.25$, and $\delta = 0.75$.

The final goal of this task is to aggregate similar entities/relations. In so doing, we adopted distinct empiric strategies for entities and predicates.

Similar entities: two entities i and j are considered similar if:

- $S_e(i, j) \geq 0.9$ or
- $0.7 < S_e(i, j) < 0.9$ and $t_{i,j} > 0.25$.

The latter condition is to give importance to a pair if they are less similar but have somewhat related types.

Similar predicates: two relations i and j are considered similar if:

- $S_r(i, j) \geq 0.8$

The module outputs a set of aggregations for both entities and relations. Each aggregation is a group of semantically similar entities or relations. In other words, each group represents a *cluster* of entities/relations. Each cluster is integrated into a proper prompt sent to the LLM. As mentioned above, splitting the elements into clusters is needed to improve the performance of LLM prompting.

Cluster disambiguation

Each entity/relation cluster contains semantically similar elements, but typically they are not all related to only one concept; it is actually what we expect from semantic aggregation. As an example, let us suppose a cluster composed of the entities *car*, *automobile*, *motorcycle*, *motorbike*, *bicycle*, and *bike*. They are similar, as all entities are means of transport but do not represent the same entities. The goal of this task is to identify which subsets refer to the same entity; in the previous example, *car* and *automobile* refer to the same entity, like *motorcycle* and *motorbike*, and *bicycle* and *bike*.

To this end, we prompt the GPT-3.5 model with another well-formed prompt, asking to return the subsets of semantically equal entities or relations. Iterating across all the clusters, the outcome of this task will be a set of semantically identical entities or relations.

Concept shrinkage

Let us remark that entity resolution aims to identify entity mentions that refer to the same concept, providing a unique entity identifier for the aforementioned mentions. Likewise, predicate resolution recognizes relations having a different textual representation but conveying an identical relationship, denoting such relations with a unique relation identifier.

To this end, each group of equal entities or relations returned by the *Cluster Disambiguation* module is used to compose a further prompt aimed at asking for a unique label for representing the underlying group. Such a label is used as the final unique representation of the given entity/relation.

4.3.3 Schema Inference

A schema is an underlying structure of a KG, usually a taxonomy or an ontology that accurately reflects the KG content. The schema contains the entity types, usually organized in a taxonomic or ontological structure, connected by proper relations. A suitable schema may be helpful in discovering implicit knowledge and supporting the exploitation of KGs in many tasks, e.g., in question answering or recommendation. Typically, a schema is manually defined by human experts, requiring considerable effort and time. To overcome this limitation, we aim to develop an iteratively bottom-up LLM-based schema inference.

The proposed method is performed by module 3 (Schema Inference), depicted in Figure 2. The module receives the set of clusters composed of resolved entities from which the corresponding types are considered. The schema inference is an iterative process that involves two main steps at each iteration:

Hypernym Generation

For each cluster, after removing the possible duplicates (as, obviously, different entities may belong to the same type), the types are embedded in an appropriate prompt sent to GPT-3.5 to find a common hypernym for the entire cluster and relation that links such hypernym to the entity types, or, depending on the cluster size and semantic similarities among types, finding a set of appropriate hypernyms, each one being related to a distinct cluster subset. For example, for the types *legumes*, *green vegetables*, *poultry*, *pork*, *fish*, and *crustacean*, the most suitable hypernyms may be *vegetables* (connected to the types *legumes* and *green vegetables*), *meat* (for *pork* and *poultry*), and *seafood* (for *fish* and *crustacean*). In all three cases, each type may be linked with the related hypernym with the relation *is type of*.

Hierarchical Agglomeration

Subsequently, all generated hypernyms and relations are merged across all clusters to remove redundancies. Let us point out that the initial entity types will represent the lower level of the schema, whereas the hypernyms will represent the upper level of the taxonomy. Afterward, for the upper level, we apply the same *Semantic Aggregation* technique described in Section 4.3.2 for finding a new set of clusters. We then applied the **Hypernym Generation** and the **Hierarchical Agglomeration** iteratively for constructing the upper levels of the taxonomy until we reach the scenario in which, at this stage, only one cluster and one hypernym are generated.

5 Experiments

This section describes the experiments we performed to validate our pipeline and the extraction capabilities of GPT-3.5. To test its validity, we identified a set of texts of interest from which to extract the triplets, along with the evaluation method and metrics that could quantify the quality of the extraction results in all their aspects (labels, descriptions, and types).

5.1 Dataset

As our input data, we gathered a set of informative texts from the English version of the SardegnaTurismo website[7], which is a portal describing tourist destinations and points of interest in Sardinia. We targeted the city of Cagliari (the capital of Sardinia), selecting 44 pages describing the city and nearby locations and providing a set of independent texts on the same topic in which recurrent mentions of the same entities are inevitably present.

The selected texts have an average of ∼660 tokens, with a peak of ∼1100 tokens (referring to GPT-3.5's tokenizer: cl100k[8]). Since each independent text afforded us to stay very far from the model's token limit, we did not undergo the need to split the text before processing, which was therefore not tested in this experiments.

5.2 Evaluation

The evaluation of a KG is a challenging key topic. Assessing the quality of the defined entities and relationships is crucial for ensuring the relevance, reliability, and suitability of the information held in the graph. Furthermore, another

[7] `https://www.sardegnaturismo.it/en/`

[8] `https://github.com/openai/openai-cookbook/blob/main/examples/How_to_count_tokens_with_tiktoken.ipynb`

topic is to guarantee the inclusion of all relevant information. We evaluate the quality of the generated entities and triplets, relying on a manual annotation, where human assessors judge each component of the resulting KG. From the annotations, we compute several metrics described in the following Sections.

5.2.1 Human assessment

A classical approach for evaluating the quality of information held by an AI model involves human assessors in judging and annotating the output of a system. In our study, human expertise may be highly effective for estimating the quality and the correctness of a generated KG. Furthermore, they can also support us in estimating completeness, e.g., identifying missing entities that should be included in the graph.

KG components annotation. We asked several assessors to judge as correct or incorrect the following components:

- *Entity*, which we judge as correct depending on two factors: (i) the input text should mention the entity, i.e. there should be an actual reference to an entity corresponding to that label and description, and (ii) the entity should be intuitively significant and relevant to the overall textual context. To illustrate these criteria, we report two entities that were evaluated as incorrect::
  - *label*: Santo Stefano del Lazzaretto
    description: A location near the Tower of Prezzemolo.
    evaluation: incorrect. The entity violates (i) because although there is a Santo Stefano del Lazzaretto in the text, it is not a location near the Tower of Prezzemolo but another name for the same Tower of Prezzemolo.
  - *label*: Nature
    description: The natural environment in Monte Claro park, including trees, bushes, and plants.
    evaluation: incorrect. The entity violates (ii) because, although the label/description could be a suitable pair to refer to the natural amenities of the park, at least in the narrow context of the text from which it comes, the entity is overly abstract, and the text barely addresses this concept.
- *Entity type*, which is considered correct if it captures the actual class and context of the entity. We provide an example:
  - *label*: Gaetano Cima
    description: Famous architect who designed the facade of San Giacomo church.
    types: [Architect, Neoclassical architecture]
    evaluation: [(Architect, correct), (Neoclassical architecture, incorrect)]
- *Triplet*, labeled as correct depending on two factors: (i) the linked entities should be correct, and (ii) it should represent a true and reasonable relation accurately expressed by the predicate label and description. We include some clarifying examples:
  - *triplet*: Nature; is part of; Monte Claro
    predicate description: Expresses a relationship of inclusion or belonging between two entities.
    evaluation: incorrect. The entity violates (i) because Nature is incorrect, as we said in the previous example.
  - *triplet*: Artistic exhibition; is showcased in; Museo del Tesoro
    predicate description: None.
    evaluation: incorrect. The entity violates (ii) because the predicate description is missing, so the predicate cannot represent any relation.
  - *triplet*: Casa Spadaccino; has garden; Outdoor activities
    predicate description: Expresses the presence of a garden in a location.
    evaluation: incorrect. The entity violates (ii) as Outdoor activities is not a garden located in Casa Spadaccino and, indeed, it is not even a garden.

Inferred components annotation. Furthermore, both correct entities and triplets are labeled with an additional annotation to evaluate whether GPT-3.5 retrieves the information from the text or draws on its knowledge. For the entities, we check whether the description contains information from the given text or additional information generated by GPT-3.5. For the triplets, we check whether the relation between the two entities that the predicate expresses can be inferred from the text or is just known a priori by the model.

As an example, the following entity is evaluated as having a GPT-3.5 generated description because there is no mention in the text of the global extent of the event or the start and end years of the war.

- *label*: WWII
  description: An abbreviation for World War II, a global war that lasted from 1939 to 1945.

Instead, we found no triplets with a relation recognized by the model but missing from the text, and perhaps we cannot provide an example.

Ground truth annotation. The assessors provided a further annotation, identifying, for each document, a list of "missed" entities, i.e., relevant entities included in the input text but not retrieved by the model. Indeed, a challenge in listing the omitted entities in an open-domain setting is the ambiguity in defining which concepts should be of interest and thus extracted since there is no topic reference. Practically, anything could be considered an entity. Therefore, we decided to evaluate the coherence of the GPT-3.5 model in the entity extraction, using the types assigned to the entities to obtain a reference schema according to which it is possible to define which entities are missing. In detail, the assessors considered all entity types automatically extracted with at least two associated entities. For each type, they identified all entities mentioned in the input text that should have been labeled with the underlying type. Aggregating the missed and correct entities composes a suitable ground truth, useful for further evaluation of the final graph.

5.2.2 Evaluation metrics

We rely on well-known metrics for assessing the extracted triplets. First, we adopt classical confusion matrix entries for assessing the performance in generating each component. In detail, each correct component is a *true positive* (TP), an incorrect item is a *false positive* (FP), and a missed entity is a *false negative* (FN). Such entries permit us to compute the *Precision* (P), i.e., the fraction of correct elements among all retrieved elements (see Eq. 3), and the *recall* (R), i.e., the fraction of correct retrieved elements among all correct elements (see Eq. 4). In other words, precision estimates the ability to generate correct components, whereas recall evaluates the ability to identify all the relevant knowledge from documents. Furthermore, a useful metric that combines precision and recall is the *F-score* (F_1), i.e., the harmonic mean of P and R (see Eq. 5).

$$P = \frac{TP}{TP + FP} \tag{3}$$

$$R = \frac{TP}{TP + FN} \tag{4}$$

$$F_1 = \frac{2 \cdot P \cdot R}{P + R} \tag{5}$$

Let us note that the recall R and F_1 can be computed only for entities, as, in this preliminary work, we asked assessors to annotate only the missing entities. Indeed, identifying information on missing relations is more challenging and requires more human effort. In detail, considering all generated components, we can compute the following metrics:

- P^E: precision of the entity generation;
- R^E: recall of the entity generation;
- F_1^E: F-score of the entity generation;
- P^T: precision of entity typing;
- P^R: precision of the relation extraction.

Furthermore, we can also estimate the ability of the model to infer additional information from its knowledge by taking into account, as already pointed out in the previous Section 5.2.1, the correct entity descriptions and the relations which are not extracted from the input document. To this end, we define a score σ corresponding to the percentage of truthful information returned by the GPT-3.5 model that comes from its internal knowledge (I) among all the returned truthful information (D):

$$\sigma = \frac{I}{D} \tag{6}$$

Therefore, we can compute the sigma score with Eq. 6 in these two variations:

- σ^E: sigma-score of the entities description generation;
- σ^R: sigma-score of the relation extraction.

5.3 Results

We applied our approach to the dataset described in Section 5.1, where each page represents an input document of the pipeline depicted in Figure 2. The system generated a basic knowledge graph comprising 761 entities and 616 triplets. Furthermore, the generated schema contains about ≈ 500 nodes and ≈ 600 edges. According to the assessment process described in the previous Section 5.2, we manually annotated all the extracted graph components with a binary label to compute the evaluation metrics. We report the final results in Table 1.

Metric	P^E	R^E	F_1^E	P^T	P^R	σ^E	σ^R
Score (%)	98.82	93.18	95.92	85.71	75.31	9.20	0.00

Table 1: Evaluation metrics scores.

Let us point out that we have conducted experiments assessing only our approach since we cannot perform comparisons with state-of-the-art tools. The main reason is that there are yet no proven state-of-the-art tools to assume as a baseline, especially considering the peculiar kind of extraction performed by our method, which, to the best of our knowledge, is the only one that invests in detailing the entities with a description and a set of types and in providing a canonical label and extended description for the predicates used in the triplets. Furthermore, using custom corpora from a use case of our particular interest, chosen because of the general lack of datasets with sufficiently elaborate text adequate for testing our approach, prevented the possibility of retrieving the already available results from any other different tool that used the same dataset. We analyzed the results objectively, knowing that a comparison with the results from other toolkits will be required for fairness in the future, at least for the specific task we perform for which other toolkits are also available.

The results, considering the early stage of our study, are encouraging, as our approach and prompts guided GPT-3.5 in extracting correct and valid entities 98.82% of the time (P^E). Furthermore, the system retrieved most of the entities within the same types, with a recall of 93.18% (R^E) and an overall F-score of 95.92% (F_1^E).

We also obtained a precision of 85.71% in the typing task (P^T) and of 75.31% in the triplet extraction (P^R), which we deem to be a very positive result given the open-domain setting, the zero-shot performance of the LLM model, and the complexity of the input text, as many of the correctly individuated relations are not blatantly explicit and involve entities that occur in distant clauses, as can be seen in the example shown in the *Relation Extraction* paragraph of Section 4.3.1.

As expected, we obtained a higher precision in the task we considered easier and a slight precision decrease in the more challenging tasks of typing and relation extraction. The σ-scores tell us how often GPT-3.5 added its own knowledge when generating entity descriptions and relations. We observed the addition of new details in entity descriptions in 9.20% of the cases (σ^E). However, since we extracted the triplets using the entity descriptions as contexts (see *Phrase Selection* in Section 4.3.1), and we found no triplets describing relations that were not in the text (σ^R), this indicates that the details added in the entity descriptions are mostly marginal and do not constitute the whole sentence. The σ-scores also indicate another positive result, as we push the model to rely on what is in the text, helping to contain possible hallucination problems.

6 Conclusion

In this paper, we proposed a novel iterative approach to open-domain knowledge graph construction that leverages the zero-shot generative and comprehension capabilities of the trending GPT-3.5 model to address the typical challenges of this research topic caused by the lack of well-established datasets and highly reliable methods.

Our approach starts with the extraction of entities and triplets, providing the entities with multiple types and an extended description of their identity and the triplets with a proper explanation of the relation represented by the predicate and subsisting between the linked entities, going through an actual "semantification" of the extracted elements. The candidate triplets are then refined in the following stages, exploiting the additional semantics offered by the types and descriptions both to formulate a suitable method to perform the entity and predicate resolution without relying on an already existing knowledge base and to infer a schema that defines the structure of the KG and simplifies its reuse and sharing.

Our experiments focused on evaluating the entities and triplets extraction capabilities of GPT-3.5 using web corpora with non-trivial content. The results show that GPT-3.5 is extremely good at entity extraction and performs very well in the typing and triplet extraction task, despite the open domain and zero-shot settings.

Although the lack of commonly available datasets makes it challengi-ng, we plan to conduct more in-depth experimentation of the proposed methods and properly compare our approach with other implementations on the same tasks. We also plan to explore the feasibility of introducing additional instructions in the prompts to define a scope, thus shifting the approach towards a closed-domain setting and focusing on the entities and triplets of particular interest. Finally, we will investigate the potential contribution of GPT-3.5's alternative generative LLMs, which are already emerging and will become increasingly popular in the future.

Acknowledgments

We acknowledge financial support under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.5 - Call for tender No.3277 published on December 30, 2021 by the Italian Ministry of University and Research (MUR) funded by the European Union – NextGenerationEU. Project Code ECS0000038 – Project Title eINS Ecosystem of Innovation for Next Generation Sardinia – CUP F53C22000430001- Grant Assignment Decree No. 1056 adopted on June 23, 2022 by the Italian Ministry of University and Research (MUR).

Furthermore, the authors thank Dr. Marco Manolo Manca, Diego Argiolas, and Gabriele Varchetta for their support in several activities of this work.

References

[1] B. Abu-Salih, M. AL-Qurishi, M. Alweshah, M. AL-Smadi, R. Alfayez, and H. Saadeh. Healthcare knowledge graph construction: A systematic review of the state-of-the-art, open issues, and opportunities. Journal of Big Data, 10(1), 2023. doi: 10.1186/s40537-023-00774-9. Cited by: 0; All Open Access, Gold Open Access.

[2] M. Agrawal, S. Hegselmann, H. Lang, Y. Kim, and D. Sontag. Large language models are few-shot clinical information extractors. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 1998–2022, 2022.

[3] D. Ashok and Z. C. Lipton. Promptner: Prompting for named entity recognition. arXiv preprint arXiv:2305.15444, 2023.

[4] Z. Bi, J. Chen, Y. Jiang, F. Xiong, W. Guo, H. Chen, and N. Zhang. Codekgc: Code language model for generative knowledge graph construction. arXiv preprint arXiv:2304.09048, 2023.

[5] L. Bonifacio, H. Abonizio, M. Fadaee, and R. Nogueira. Inpars: Data augmentation for information retrieval using large language models, 2022.

[6] D. Cer, Y. Yang, S. yi Kong, N. Hua, N. L. U. Limtiaco, R. S. John, N. Constant, M. Guajardo-Céspedes, S. Yuan, C. Tar, Y. hsuan Sung, B. Strope, and R. Kurzweil. Universal sentence encoder. In In submission to: EMNLP demonstration, Brussels, Belgium, 2018. URL `https://arxiv.org/abs/1803.11175`. In submission.

[7] J. P. Chiu and E. Nichols. Named Entity Recognition with Bidirectional LSTM-CNNs. Transactions of the Association for Computational Linguistics, 4:357–370, 07 2016. ISSN 2307-387X. doi: 10.1162/tacl_a_00104. URL `https://doi.org/10.1162/tacl_a_00104`.

[8] H. Du, Y. Tang, and Z. Cheng. An efficient joint framework for interacting knowledge graph and item recommendation. Knowledge and Information Systems, 65(4):1685 – 1712, 2023. doi: 10.1007/s10115-022-01808-z. Cited by: 0.

[9] M. Ebraheem, S. Thirumuruganathan, S. R. Joty, M. Ouzzani, and N. Tang. Deeper - deep entity resolution. ArXiv, abs/1710.00597, 2017.

[10] S. Elhammadi, L. V. Lakshmanan, R. Ng, M. Simpson, B. Huai, Z. Wang, and L. Wang. A high precision pipeline for financial knowledge graph construction. In Proceedings of the 28th international conference on computational linguistics, pages 967–977, 2020.

[11] S. Hao, B. Tan, K. Tang, H. Zhang, E. P. Xing, and Z. Hu. Bertnet: Harvesting knowledge graphs from pretrained language models. arXiv preprint arXiv:2206.14268, 2022.

[12] X. Hou. Design and application of intelligent financial accounting model based on knowledge graph. Mobile Information Systems, 2022, 2022. doi: 10.1155/2022/8353937. Cited by: 1; All Open Access, Gold Open Access, Green Open Access.

[13] B. Ji. Vicunaner: Zero/few-shot named entity recognition using vicuna. arXiv preprint arXiv:2305.03253, 2023.

[14] V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady, 10:707, 1966.

[15] B. Li, G. Fang, Y. Yang, Q. Wang, W. Ye, W. Zhao, and S. Zhang. Evaluating chatgpt's information extraction capabilities: An assessment of performance, explainability, calibration, and faithfulness. ArXiv, abs/2304.11633, 2023.

[16] Y. Luan, L. He, M. Ostendorf, and H. Hajishirzi. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. arXiv preprint arXiv:1808.09602, 2018.

[17] A. Mehta, A. Singhal, and K. Karlapalem. Scalable knowledge graph construction over text using deep learning based predicate mapping. In Companion Proceedings of The 2019 World Wide Web Conference, pages 705–713, 2019.

[18] T. H. Nguyen and R. Grishman. Relation extraction: Perspective from convolutional neural networks. In VS@HLT-NAACL, 2015.

[19] C. Niklaus, M. Cetto, A. Freitas, and S. Handschuh. A survey on open information extraction. In International Conference on Computational Linguistics, 2018.

[20] N. Peng. Controllable text generation for open-domain creativity and fairness, 2022.

[21] I. H. Sarker. Machine learning: Algorithms, real-world applications and research directions. SN Comput. Sci., 2 (3), mar 2021. doi: 10.1007/s42979-021-00592-x. URL https://doi.org/10.1007/s42979-021-00592-x.

[22] M. Trajanoska, R. Stojanov, and D. Trajanov. Enhancing knowledge graph construction using large language models. arXiv preprint arXiv:2305.04676, 2023.

[23] Z. Wan, F. Cheng, Z. Mao, Q. Liu, H. Song, J. Li, and S. Kurohashi. Gpt-re: In-context learning for relation extraction using large language models. ArXiv, abs/2305.02105, 2023.

[24] C. Wang, X. Liu, and D. Song. Language models are open knowledge graphs. arXiv preprint arXiv:2010.11967, 2020.

[25] X. Wei, X. Cui, N. Cheng, X. Wang, X. Zhang, S. Huang, P. Xie, J. Xu, Y. Chen, M. Zhang, Y. Jiang, and W. Han. Zero-shot information extraction via chatting with chatgpt. ArXiv, abs/2302.10205, 2023.

[26] G. Weikum, X. L. Dong, S. Razniewski, and F. Suchanek. Machine knowledge: Creation and curation of comprehensive knowledge bases. Found. Trends Databases, 10(2–4):108–490, jul 2021. ISSN 1931-7883. doi: 10.1561/1900000064. URL https://doi.org/10.1561/1900000064.

[27] Q. Wu, D. Fu, B. Shen, and Y. Chen. Semantic service search in it crowdsourcing platform: A knowledge graph-based approach. International Journal of Software Engineering and Knowledge Engineering, 30(6):765 – 783, 2020. doi: 10.1142/S0218194020400069. Cited by: 6.

[28] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023.

[29] T. Zhang, F. Ladhak, E. Durmus, P. Liang, K. McKeown, and T. B. Hashimoto. Benchmarking large language models for news summarization, 2023.