2023 IEEE International Conference on Big Data (BigData) | 979-8-3503-2445-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/BigData59044.2023.10386454

AutoKG: Efficient Automated Knowledge Graph Generation for Language Models

Bohan Chen* and Andrea L. Bertozzi
Department of Mathematics, University of California, Los Angeles
520 Portola Plaza
Los Angeles, CA, 90095, USA

***Abstract*—Traditional methods of linking large language models (LLMs) to knowledge bases via the semantic similarity search often fall short of capturing complex relational dynamics. To address these limitations, we introduce AutoKG, a lightweight and efficient approach for automated knowledge graph (KG) construction. For a given knowledge base consisting of text blocks, AutoKG first extracts keywords using a LLM and then evaluates the relationship weight between each pair of keywords using graph Laplace learning. We employ a hybrid search scheme combining vector similarity and graph-based associations to enrich LLM responses. Preliminary experiments demonstrate that AutoKG offers a more comprehensive and interconnected knowledge retrieval mechanism compared to the semantic similarity search, thereby enhancing the capabilities of LLMs in generating more insightful and relevant outputs.**

***Index Terms*—Language model, Knowledge Graph, Graph Learning, Retrieval-augmented Generation**

I. INTRODUCTION

Large Language Models (LLMs) such as BERT [1], RoBERTA [2], T5 [3], and PaLM [4], are intricately designed architectures equipped with an extensive number of parameters. These models have been rigorously pre-trained on vast and diverse corpora, thereby enabling them to excel in a wide array of Natural Language Processing (NLP) tasks, from language understanding to both conditional and unconditional text generation [5], [6]. These advancements have been heralded as a step toward higher-bandwidth human-computer interactions. However, their deployment faces significant challenges. On one hand, LLMs exhibit a tendency for 'hallucinations' [7], [8], providing plausible yet nonfactual predictions. On the other hand, the black-box nature of LLMs compromises both interpretability and factual accuracy, often resulting in erroneous statements despite memorizing facts during training [9], [10].

Knowledge in natural language can be externally sourced from a retrievable database, reducing hallucinations and enhancing the interpretability of LLMs [11]. Utilizing dense neural retrievers, which employ dense query and document vectors generated by a neural network [12], the system can evaluate the semantic similarity to an information-seeking query by calculating the embedding vector similarity across related concepts [13], [14].

To go beyond mere semantic similarity in information retrieval and augment the reasoning capabilities of LLMs, two advanced methodologies are particularly transformative: prompt engineering like the Chain-of-thought prompting, and the incorporation of Knowledge Graphs (KGs) [15]. The former, chain-of-thought prompting, provides a framework for advanced reasoning by generating paths of explanations and predictions that are cross-verified through knowledge retrieval [16], [17]. While this method offers significant benefits, it is not the primary focus of this study. As for the latter, KGs offer LLMs a structured and efficient way to address their limitations in factual accuracy and reasoning [15], [18]. KGs not only provide accurate and explicit knowledge crucial for various applications [19] but are also known for their symbolic reasoning capabilities to produce interpretable results [20]. These graphs are dynamic, continuously evolving with the addition of new knowledge [21], and can be specialized for domain-specific requirements [22].

In this study, our emphasis is on techniques of automated KG generation and incorporation with LLMs. Most of the works related to these two tasks rely intensively on the ongoing training of neural networks [15], [23], which is both difficult to employ and less flexible for on-the-fly updates. Traditional KG construction approach uses NLP techniques for entity recognition [24], [25], or keyword identification based on term frequency [26], [27], followed by determining relationship strength through word proximity [28]. Current automated techniques necessitate neural network training [29]–[31]. As for the interaction between KGs and LLMs, neural networks are trained to let LLMs understand the information retrieved from KGs [32], [33].

The recent advancements in LLMs make us think much more simply about the automatic generation of KGs and the integration of LLMs with KGs. State-of-the-art LLMs such as ChatGPT[1], BARD[2], and LLAMA [34] have demonstrated impressive reasoning capabilities [35], [36]. Given sufficient information, they can independently execute effective inference. This observation suggests an opportunity to simplify the KG structure: perhaps the intricate relational patterns found in traditional KGs could be simplified into basic strength indicators of association. Consequently, specific relationships

---

Bohan Chen is supported by the UC-National Lab In-Residence Graduate Fellowship Grant L21GF3606. This work is supported by NSF grants DMS-2027277 and DMS-2318817.

* Corresponding author (email: bhchenyz@ucla.edu).

[1] https://openai.com/blog/chatgpt

[2] https://blog.google/technology/ai/bard-google-ai-search-updates/

Authorized licensed use limited to: University Town Library of Shenzhen. Downloaded on July 30,2024 at 06:36:27 UTC from IEEE Xplore. Restrictions apply.

are implicitly conveyed to the model through corpus blocks associated with the KG. In addition, we can provide retrieved keywords and the related corpus directly in the prompt rather than training a network to let LLMs understand the retrieved subgraph structure.

Motivated by these ideas, this study makes the following contributions:

1) We introduce AutoKG, an innovative method for automated KG generation, based on a knowledge base comprised of text blocks. AutoKG circumvents the need for training or fine-tuning neural networks, employs pretrained LLMs for extracting keywords as nodes, and applies graph Laplace learning to evaluate the edge weights between these keywords. The output is a simplified KG, where edges lack attributes and directionality, possessing only a weight that signifies the relevance between nodes.
2) We present a hybrid search strategy in tandem with prompt engineering, which empowers large LLMs to effectively utilize information from the generated KGs. This approach simultaneously searches for semantically relevant corpora based on embedding vectors and the most pertinent adjacent information within the knowledge graphs.

The KG constructed here is a simplified version compared to traditional KGs, which are typically composed of relations in the form of triplets. Firstly, nodes in AutoKG are not entities in the usual sense; they are more abstract keywords. These keywords can represent entities, concepts, or any content that serves as a foundation for search. Additionally, instead of directed edges with specific semantic meanings found in traditional KGs, AutoKG utilizes undirected edges with a single weight value. The node keywords are extracted from the knowledge base with the aid of LLMs, while the graph structure is algorithmically derived. Such a KG can be efficiently stored with just a keyword list and a sparse adjacency matrix.

Section II explains the detailed process of automated KG generation, while Section III describes the hybrid search method. An essential highlight is that our proposed techniques require no neural network training or fine-tuning.

II. AUTOMATED KG GENERATION

In this section, we introduce our proposed approach, AutoKG, for automated KG generation. The training aspects of the LLM are not the focus of this article. We operate under the assumption that the LLM is already pre-trained and is accompanied by a corresponding vector embedding model. Specifically, we have employed OpenAI's *gpt-4* or *gpt-3.5-turbo-16k* as the LLM and the *text-embedding-ada-002* as the embedding model.

Consider a scenario involving an external knowledge base, comprised of discrete text blocks. AutoKG constructs a KG where the nodes represent keywords extracted from the external knowledge base. The edges between these nodes carry a single non-negative integer weight, signifying the strength of the association between the connected keywords. AutoKG encompasses two primary steps: the extraction of keywords, which correspond to the nodes in the graph, and the establishment of relationships between these keywords, represented by the edges in the graph. It is worth noting that the pretrained LLM is employed only in the keyword extraction step of the process. Figure 1 is the flowchart of the KG construction.

A. Keywords Extraction

Let the external knowledge base be denoted by $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, where each $\mathbf{x}_i$ is a block of text with the maximum length of T tokens, represented as a string. The corresponding embedding vectors for these text blocks are encapsulated in $\mathcal{V} = \{\mathbf{v}(\mathbf{x}_1), \mathbf{v}(\mathbf{x}_2), \dots, \mathbf{v}(\mathbf{x}_N)\} \subset \mathbb{R}^d$, where $\mathbf{v}$ is the embedding projection from string to $\mathbb{R}^d$. We extract keywords from the knowledge base $\mathcal{X}$ with unsupervised clustering algorithms and the assistance of LLMs.

Algorithm 1 outlines the keyword extraction process. The algorithm takes as input all text blocks and their corresponding embedding vectors $\mathcal{X}$ and $\mathcal{V}$, along with pre-defined parameters: n for the number of clusters, c for the number of text blocks to select, and l_1, l_2 as keyword extraction parameters. Additionally, the algorithm also utilizes a parameter m to specify the number of sampled previous keywords. Two unsupervised clustering algorithms, K-means clustering [37], [38] and spectral clustering [39], are applied to cluster the knowledge base. For each cluster identified, we sample $2c$ text blocks, with c closest to the cluster center and c randomly selected, to capture both the global and centered information. The LLM is used twice in this algorithm. First, it extracts keywords from a selection of $2c$ text blocks, guided by the parameters l_1 and l_2, while avoiding the sampled m previous keywords. Second, the same LLM is employed to filter and refine the extracted keywords.

The construction of the prompts for these applications strictly follows the format outlined in Table I. A specific prompt example for the keyword extraction is given in the Appendix. Specifically, each prompt is formed by concatenating the *Task Information*, *Input Information*, *Additional Requirements*, and *Outputs*. It is essential to note that within each task, the length of the prompt sections corresponding to *Task Information* and *Additional Requirements* is fixed.

For Task 1, which deals with keyword extraction, the maximum input length is set to $2cT + m(l_2 + 1)$, where T represents the token length of a single text block. Note that each keyword can have a length of up to $l_2 + 1$ tokens when accounting for potential separators such as commas. Similarly, the maximum output length is $l_1(l_2 + 1)$, where l_1 is the maximum number of keywords and l_2 is the maximum token length of each keyword. Since Task 1 is applied once for each of the n clusters generated by the two clustering methods, the total maximum token usage for Task 1 would be $2n(2cT + (m + l_1)(l_2 + 1))$. This process yields a maximum of $2nl_1$ extracted keywords. For Task 2, which involves filtering and refining the keywords, the maximum lengths for both the input and output are governed by the formula $2nl_1(l_2 + 1)$.

Authorized licensed use limited to: University Town Library of Shenzhen. Downloaded on July 30,2024 at 06:36:27 UTC from IEEE Xplore. Restrictions apply.

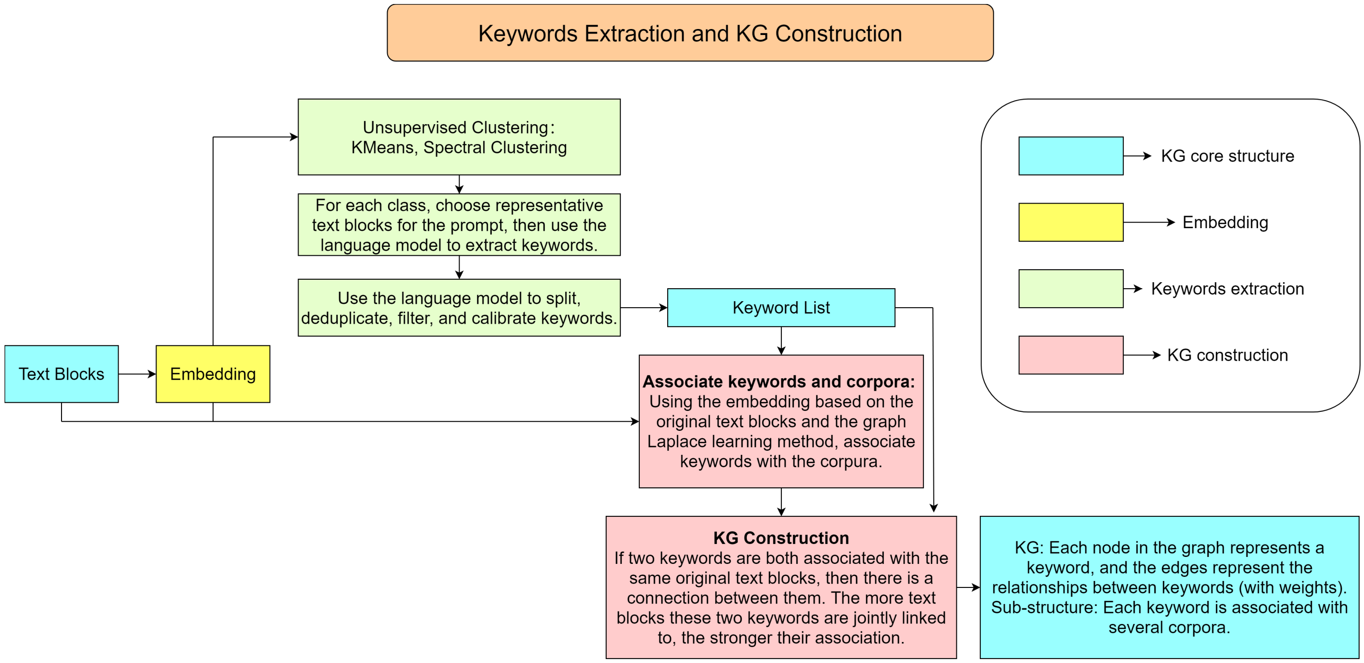
Keywords Extraction and KG Construction
Unsupervised Clustering: KMeans, Spectral Clustering
For each class, choose representative text blocks for the prompt, then use the language model to extract keywords.
Use the language model to split, deduplicate, filter, and calibrate keywords.
Keyword List
Text Blocks
Embedding
Associate keywords and corpora: Using the embedding based on the original text blocks and the graph Laplace learning method, associate keywords with the corpura.
KG Construction If two keywords are both associated with the same original text blocks, then there is a connection between them. The more text blocks these two keywords are jointly linked to, the stronger their association.
KG: Each node in the graph represents a keyword, and the edges represent the relationships between keywords (with weights). Sub-structure: Each keyword is associated with several corpora.
KG core structure
Embedding
Keywords extraction
KG construction

Fig. 1. Flowchart of the KG Construction Process. This figure illustrates the different steps involved in the construction of the KG. The blue blocks represent the core components of the KG, yellow blocks indicate the embedding process, green blocks focus on keyword extraction, and the red blocks correspond to the establishment of relationships between keywords and the corpus as well as among the keywords themselves.

In summary, the maximum usage of tokens $M_{\text{tokens KG}}$ for the keyword extraction process is

$$M_{\text{tokens KG}} = 2n(2cT + (m + 2l_1)(l_2 + 1)) + L_F, \qquad (1)$$

where L_F is the fixed total length of tokens of the task information and additional requirement parts.

B. Graph Structure Construction

In this section, we detail how to construct a KG based on the keywords extracted in Section II-A. Specifically, we establish whether there are edges between keywords and how to weight these edges. We propose a method based on label propagation on the graph, a step that does not require the involvement of any LLM.

Firstly, we create a graph $G^t = (\mathcal{X}, W^t)$ where $\mathcal{X}$ is the set of text blocks serving as the nodes of graph G^t, and W^t is the weight matrix for the edges. W^t_{ij} is determined by the similarity between the corresponding embedding vectors $\mathbf{v}_i$ and $\mathbf{v}_j$. Define the similarity function:

$$w(\mathbf{v}_i, \mathbf{v}_j) = \exp\left(-\frac{\angle(\mathbf{v}_i, \mathbf{v}_j)^2}{\sqrt{\tau_i \tau_j}}\right), \qquad (2)$$

where $\angle(\mathbf{v}_i, \mathbf{v}_j) = \arccos\left(\frac{\mathbf{v}_i^\top \mathbf{v}_j}{\|\mathbf{v}_i\|\|\mathbf{v}_j\|}\right)$ is the angle between feature vectors $\mathbf{v}_i$ and $\mathbf{v}_j$. The normalization constant τ_i is chosen according to the similarity to the Kth nearest neighbor of i (i.e., $\tau_i = \angle(\mathbf{v}_i, \mathbf{v}_{i_K})$, where $\mathbf{v}_{i_K}$ is the Kth nearest neighbor to $\mathbf{v}_i$).

For computational efficiency, we construct a sparse weight matrix W^t by considering only the K-nearest neighbors [40] for each vertex. Let $x_{i_k}, \ k = 1, 2, \ldots, K$ be the K-nearest neighbors (KNN) of x_i (including x_i itself) according to angle similarity. Define a sparse weight matrix by

$$\bar{W}^t_{ij} = \begin{cases} w(\mathbf{v}_i, \mathbf{v}_j), \ j = i_1, i_2, \ldots, i_K, \\ 0, \ \text{otherwise}. \end{cases} \qquad (3)$$

K is chosen to ensure that the corresponding graph G^t is connected, empirically $K = 30$. We symmetrize the sparse weight matrix to obtain our final weight matrix W^t by redefining $W^t_{ij} := (\bar{W}^t_{ij} + \bar{W}^t_{ji})/2$. Note that W^t is sparse, symmetric, and non-negative (i.e. $W^t_{ij} \geq 0$).

Next, we utilize the graph $G^t = (\mathcal{X}, W^t)$, constructed on text blocks, to establish a keyword KG $G^k = (\mathcal{K}, W^k)$. Here, $\mathcal{K}$ is the set of keywords, and W^k is the weight matrix for the edges. In this matrix, W^k_{ij} quantifies the strength of association between keywords $\mathbf{k}_i$ and $\mathbf{k}_j$. Importantly, this association is not semantic but is reflected across the entire corpus in the knowledge base. Specifically, W^k_{ij} corresponds to the count of text blocks that are simultaneously associated with both keywords $\mathbf{k}_i$ and $\mathbf{k}_j$.

Algorithm 2 establishes the relationship between a keyword and text blocks. The core idea is to select a subset of text blocks that are closest to the keyword as positive data, and another subset that is farthest as negative data. We then employ graph Laplace learning [41] based on the graph structure $G^t = (\mathcal{X}, W^t)$ that we have previously constructed for text blocks. The graph Laplace learning is a semi-supervised learning method on graphs, utilizing the harmonic property of the solution function $u : \mathcal{X} \to [0, 1]$ to diffuse the label values

Authorized licensed use limited to: University Town Library of Shenzhen. Downloaded on July 30,2024 at 06:36:27 UTC from IEEE Xplore. Restrictions apply.

TABLE I
PROMPT CONSTRUCTION FOR DIFFERENT TASKS USING LLM

ID	Task Information	Input Information	Additional Requirements	Outputs
1	Keywords Extraction	1.Sampled text blocks 2.Sampled previous keywords	1. Avoid previous keywords 2. Output up to l_1 keywords 3. Each output keyword is at most l_2 tokens	Extracted Keywords
2	Refining Keywords	Originally extracted keywords	Concentration, deduplication, splitting, and deletion	Refined Keywords
3	Response to the Query	1. Original query 2. Related text blocks 3. Related keywords	Indicate the method used to search for texts and keywords: Direct, via keywords, or KG adjacency search	Final Response

Algorithm 1 Algorithm for Keyword Extraction in AutoKG

```
Input: All text blocks and their corresponding embedding vectors $\mathcal{X}$ and $\mathcal{V}$, pre-defined parameters $n$ (number of clusters), $c$ (number of text blocks to select), $l_1, l_2$ (keyword extraction parameters), $m$ (number of sampled previous keywords)
Output: A set of extracted keywords $\mathcal{K} = \{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_M\}$
$\mathcal{K} = \emptyset$
for each clustering algorithm $P$ in {k-means, spectral clustering} do
    Cluster $\mathcal{V}$ into $n$ clusters $\mathcal{V}_i^P,\ i = 1, 2, \ldots, n$ using algorithm $P$
    for i = 1,2,...,n do
        Randomly select $c$ text blocks and $c$ nearest to the cluster center from cluster $\mathcal{V}_i^P$
        if $|\mathcal{K}| > m$ then
            Select a subset $\mathcal{K}_s \subset \mathcal{K}$ such that $|\mathcal{K}_s| = m$
        else
            $\mathcal{K}_s = \mathcal{K}$
        end if
        Include these $2c$ text blocks and previous keywords $\mathcal{K}_s$ in a prompt for keyword extraction
        Use LLM, extract up to $l_1$ keywords of maximum token length $l_2$, collected as $\mathcal{K}_i^P$
        Update $\mathcal{K} = \mathcal{K} \cup \mathcal{K}_i^P$
    end for
end for
Filter and refine $\mathcal{K}$ using a LLM to obtain the final keyword list
return $\mathcal{K}$
```

Algorithm 2 Identifying Keyword to Text Block Association

```
Input: Keyword $\mathbf{k}$, Set of text blocks $\mathcal{X}$, Forward relation parameter $n_1$, Backward relation parameter $n_2$
Output: $\mathcal{X}^{\mathbf{k}} \subset \mathcal{X}$, the subset of $\mathcal{X}$ associated with $\mathbf{k}$
Obtain the embedding vector $\mathbf{v}(\mathbf{k})$.
Find the $n_1$ nearest vectors in $\mathcal{X}$ to $\mathbf{v}(\mathbf{k})$ (label them as 1) and $n_2$ farthest vectors (label them as 0).
In the text-block graph $G^t = (\mathcal{X}, W^t)$, use the graph Laplace learning algorithm [41] to label the remaining nodes based on these $n_1 + n_2$ labeled nodes. Obtain a real-valued function $u : \mathcal{X} \to [0, 1]$ on the graph nodes.
Define $\mathcal{X}^{\mathbf{k}} = \{\mathbf{x}_i \in \mathcal{X} : u(\mathbf{x}_i) \geq 0.5\}$
return $\mathcal{X}^{\mathbf{k}}$
```

from a subset of labeled nodes to other unlabeled nodes in the graph. The text blocks that are classified towards the positive side (with a node function value $u \geq 0.5$) are considered to be associated with the keyword.

The association weight W_{ij}^k between $\mathbf{k}_i$ and $\mathbf{k}_j$ is defined as follows:

$$W_{ij}^k = W_{ji}^k = |\mathcal{X}^{\mathbf{k}_i} \cap \mathcal{X}^{\mathbf{k}_j}|. \tag{4}$$

With this, we complete the construction of the keyword-based KG G^k, which is built upon the text block graph G^t.

C. Time Complexity Analyzation

This section analyzes the efficiency of the AutoKG method. The token consumption required for KG construction in the AutoKG method has the upper bound according to Eq. 1. The efficiency of the algorithm is mainly influenced by three aspects:

1) **Constructing the similarity graph based on text blocks** $G^t = (\mathcal{X}, W^t)$: An approximate nearest neighbor search [40] is employed for KNN search, leading to a complexity of $O(N \log N)$.
2) **Clustering algorithm**: Since both K-means clustering [37], [38] and spectral clustering [39] are NP-hard, we bound the complexity by I_{max}, the preset maximum number of iterations. Spectral clustering is essentially the Kmeans method augmented with an eigendecomposition of the graph Laplacian. The time complexity here is mainly dominated by the Kmeans method and is $O(NndI_{\text{max}})$, where n is the number of clusters, and d is the vector dimension (1536 for OpenAI's embedding model).
3) **Graph Laplace learning**: Given that our graph Laplacian matrix is sparse, employing the conjugate gradient method to solve the graph Laplace learning problem results in a time complexity of $O(\hat{N}\sqrt{\kappa})$, where $\hat{N}$ represents the count of non-zero elements in the graph Laplacian matrix, and $\sqrt{\kappa}$ denotes the condition number. We have the upper bound for $\hat{N}$ as $2KN$, where K is the number of nearest neighbors.

Considering these factors, for large N and if preconditioning techniques can keep the condition number of the graph Laplacian matrix small, our automated KG construction algorithm

Authorized licensed use limited to: University Town Library of Shenzhen. Downloaded on July 30,2024 at 06:36:27 UTC from IEEE Xplore. Restrictions apply.

should operate with a time complexity of

$$O(N\log N + NndI_{\max} + 2KN\sqrt{\kappa}) = O(N\log N + Nn),$$

where the number of clusters practically depends on N.

D. Remarks

In the process of generating the entire KG, there are several points to be considered:

- Although the keywords are extracted from clusters of text blocks, we do not take into account the previous clustering results when establishing the relationship between keywords and text blocks. This is because the same keyword may be included in multiple clusters.
- When constructing the relationship between keywords, we did not incorporate the embedding vectors of the keywords into the graph for the graph Laplace learning process. There are two reasons for this decision: first, we do not need to update the graph structure when selecting different keywords; second, empirically speaking, the embedding vectors of the keywords tend to be quite distant from the embedding vectors of the text blocks. Therefore, including them in the initial label data for Laplace learning might be meaningless.

Our approach considerably outperforms these conventional methods in both keyword extraction and relationship construction. The primary shortcoming of traditional techniques is their reliance on a fixed set of words, leading to a significant loss of related information and often producing overly localized insights. In terms of keyword extraction, our method leverages the capabilities of LLMs, allowing for the refining of keywords that are more central to the topic at hand, rather than merely being high-frequency terms. When it comes to relationship construction, our strategy is grounded in a macroscopic algorithm on graphs of all text blocks. This approach encompasses the information from the entire knowledge base of text blocks, providing a more comprehensive perspective compared to relationships derived from local distances.

III. HYBRID SEARCH: INCORPORATING KG AND LLM

In this section, we propose a hybrid search approach, based on the KG generated according to Section II. For a given query, the search results using this hybrid search strategy include not only the text blocks that are semantically related to the query but also additional associative information sourced from the KG. This supplementary data serves to provide more detailed and in-depth reasoning for further analysis by the model. The incorporation of a KG allows us to capture complex relationships between different entities, thereby enriching the contextual understanding of the query.

In our proposed hybrid search approach, we have devised a multi-stage search process that incorporates both direct text block search as well as keyword-based searching guided by the KG. This process is detailed in Algorithm 3. Initially, we perform the initial search by computing the text blocks that are closest to the given query embedding vector. Then, we turn to the KG and identify the keywords that are closest to the query, along with text blocks associated with these keywords. Lastly, we identify additional keywords that have the strongest association with the previously identified ones, based on the weight matrix in the KG, and accordingly search for related text blocks. The algorithm returns not just a set of text blocks that are highly relevant to the query, but also a set of keywords that are closely connected to the query.

Algorithm 3 hybrid search Algorithm

```
Input: Query $\mathbf{q}$, embedding vector $\mathbf{v}(\mathbf{q})$, Parameters $(s_0^t, s_1^k, s_1^t, s_2^k, s_2^t)$
Output: Set $\mathcal{X}_{\text{final}}$ containing text blocks related to $\mathbf{q}$, and Set $\mathcal{K}_{\text{final}}$ containing keywords related to $\mathbf{q}$
Step 1: Vector Similarity Search
Find the closest $s_0^t$ text blocks in $\mathcal{X}$ to $\mathbf{v}(\mathbf{q})$
$\mathcal{X}_0 \leftarrow$ set of closest $s_0^t$ text blocks
Step 2: Similar Keyword Search
Find the closest $s_1^k$ keywords in $\mathcal{K}$ to $\mathbf{v}(\mathbf{q})$
$\mathcal{K}_1 \leftarrow$ set of closest $s_1^k$ keywords
For each $\mathbf{k}$ in $\mathcal{K}_1$, find the closest $s_1^t$ text blocks in $\mathcal{X}$
$\mathcal{X}_1 \leftarrow$ merged set of closest $s_1^t$ text blocks for each $k$ in $\mathcal{K}_1$
Step 3: Keyword Adjacency Search
For each $\mathbf{k}$ in $\mathcal{K}_1$, find $s_2^k$ strongest connected keywords according to $W^k$
$\mathcal{K}_2 \leftarrow$ merged set of $s_2^k$ strongest connected keywords for each $k$ in $\mathcal{K}_1$
For each $\mathbf{k}$ in $\mathcal{K}_2$, find the closest $s_2^t$ text blocks in $\mathcal{X}$
$\mathcal{X}_2 \leftarrow$ merged set of closest $s_2^t$ text blocks for each $k$ in $\mathcal{K}_2$
$\mathcal{X}_{\text{final}} \leftarrow \mathcal{X}_0 \cup \mathcal{X}_1 \cup \mathcal{X}_2$
$\mathcal{K}_{\text{final}} \leftarrow \mathcal{K}_1 \cup \mathcal{K}_2$
return $\mathcal{X}_{\text{final}}, \mathcal{K}_{\text{final}}$
```

To estimate the maximum number of tokens returned by the hybrid search, we consider the maximum number of tokens T for a single text block and l_2 for a single keyword. The total number of keywords retrieved will be $s_1^k + s_1^k \cdot s_2^k$, and the total number of text blocks will be $s_0^t + s_1^k \cdot s_1^t + s_1^k \cdot s_2^k \cdot s_2^t$. Therefore, the maximum number of tokens $M_{\text{tokens QA}}$ can be calculated as:

$$M_{\text{tokens QA}} = s_1^k \cdot l_2 \cdot (1+s_2^k) + T \cdot (s_0^t + s_1^k \cdot s_1^t + s_1^k \cdot s_2^k \cdot s_2^t). \tag{5}$$

In practical applications, the actual number of tokens obtained through the search often falls below the theoretical maximum. This is because there is substantial overlap between the text blocks and keywords discovered via different search methods. Subsequently, the retrieved information is incorporated into the prompt to enhance the LLM's response to the original query. For details on prompt construction, one may refer to Task 3 in Table I. A specific prompt example is provided in the Appendix. Importantly, an adaptive approach can be employed during the prompt construction to ensure that the maximum token limit for the LLM is not exceeded. Text blocks can be added sequentially until the token limit is reached.

Authorized licensed use limited to: University Town Library of Shenzhen. Downloaded on July 30,2024 at 06:36:27 UTC from IEEE Xplore. Restrictions apply.

IV. EXPERIMENTS AND RESULTS

In this section, our primary goal is to demonstrate through experiments that our proposed AutoKG approach provides significantly better responses while maintaining a comparable efficiency, compared with the retrieval-augmented generation (RAG) method based on semantic vector similarity [13], [14]. Our approach that combines AutoKG and hybrid search extracts more valuable information for the model than RAG which relies on semantic vector similarity search.

Unfortunately, we encountered challenges in identifying a suitable dataset to conduct these experiments. We attempted to utilize the WikiWhy dataset [42], which is designed to evaluate the reasoning capability of models. The dataset comprises approximately 9,000 entries. Each entry contains a paragraph of content, spanning between 100 to 200 words. Based on this content, every entry provides a "why" question along with its corresponding cause-effect relationship and explanation. When we employ the hybrid search based on AutoKG or the semantic vector similarity search of RAG, we can easily retrieve the content corresponding to the given question and instruct the model to answer based on that content. In both methods, the model's responses are almost identical. Since the 9,000 entries are relatively independent of each other, cross-entry data retrieval provided by our method doesn't significantly contribute to answering the questions.

As a consequence, we adopt qualitative approaches rather than employing numerical metrics to evaluate the experimental performance of our method. First, we provide a simple example to explain why our AutoKG with hybrid search approach has benefits compared to methods based on semantic vector similarity search. Next, we present a detailed example based on all 40 references of this article and the associated subgraph from the KG used during the query. Finally, we compare the efficiency of hybrid search and semantic vector similarity search from both theoretical and experimental perspectives.

A. A Simple Example: Why We Need KG?

Consider a simple knowledge base that contains text blocks detailing a day in the life of an individual named Alex, along with related information. The core narrative is that after leaving his home in the morning, Alex goes to Cafe A to buy a coffee and then takes a bus to Company B for work. Interspersed within the knowledge base are numerous pieces of granular information such as conversations Alex had with the barista at the cafe, the coffee order details, dialogues on the bus, as well as conversations at his workplace, and so forth.

The point of interest here is how a model would answer the question: *"Was it raining this morning when Alex left his home?"* under the assumption that there is no direct answer to this question and no content about the weather in the knowledge base. We aim to compare the responses given the support information retrieved using our method versus that retrieved through semantic similarity search. Within the knowledge base, there are two indirect pieces of information hinting at the weather conditions:

1) Related to Cafe A: "Many people were chatting and drinking coffee in the square outside Cafe A."
2) Related to Company B: "The car wash located downstairs of Company B was bustling with business today."

Both these snippets subtly suggest that it was not raining.

Given that the question is primarily about Alex and the weather, the information retrieved from the knowledge base through semantic similarity vector search would only be about Alex (as there is no direct information about the weather). The search results would primarily outline his movements throughout the day. Even with an increase in search entries, it would mostly retrieve additional miscellaneous details, like his coffee order and dialogues. Unfortunately, these details do not contain any hints to infer the day's weather.

On the other hand, employing AutoKG with a hybrid search approach yields different results. During the KG generation process, we extract keywords such as Alex, Cafe A, and Company B. With the hybrid search, the initial step uses the input question to retrieve the keyword Alex. Then, the adjacency search identifies Cafe A and Company B as related keywords. Subsequently, text blocks are sought based on these keywords, resulting in the identification of implicit weather-related information. This example illustrates the utility of the hybrid search. Semantic similarity alone can lack cross-topic connections. It tends to retrieve many minor details within the scope of a given question. When searching with the KG constructed using the AutoKG method, the breadth and diversity of the retrieved information is enhanced. Moreover, prior work has easily substantiated GPT-4's capability to reason effectively with provided clues [35], [36].

From the dialogue record with GPT-4 in the Appendix, it is evident that GPT-4 can accurately infer that it did not rain today when given clues about today's weather. However, when only provided with information about Alex from the semantic similarity vector search, it cannot make any predictions about today's weather.

B. An Example with Article References

We present a concrete example utilizing content from the 42 references cited in this paper. The resulting KG is interactively queried using the hybrid search method outlined above. Both the KG generation and subsequent querying processes were performed using the *gpt-3.5-turbo-16k* model, chosen to minimize cost. The 40 references, once segmented, comprise 5,261 text blocks, each less than 201 tokens in length. For the keyword extraction process, as per Algorithm 1, the parameters are: $n = 15, c = 15, l_1 = 10, l_2 = 3, m = 300$. For Algorithm 2, we use the parameters $n_1 = 5$ and $n_2 = 35$. The entire KG construction consumes 137,516 tokens, which is less than the theoretical maximum of 181,280 tokens given by Eq. 1. This calculation of the theoretical maximum does not account for the fixed total length of tokens pertaining to task information and additional requirement parts.

The constructed KG comprises 461 nodes (extracted keywords) with its adjacency matrix containing 40,458 non-zero elements. The node with the highest degree in the graph

Authorized licensed use limited to: University Town Library of Shenzhen. Downloaded on July 30,2024 at 06:36:27 UTC from IEEE Xplore. Restrictions apply.

is connected to 289 neighbors. There are 353 nodes whose degree is less than 92, which is 20% of the maximum possible degree of 460. The entire process of KG construction took approximately ten minutes. All computations, excluding calls to the OpenAI API, are carried out on a CPU with an Intel i9-9900. Both keyword extraction and KG construction take approximately five minutes each. For the subsequent hybrid search described in Algorithm 3, we use the parameters $(s_0^t = 15, s_1^k = 5, s_1^t = 3, s_2^k = 3, s_2^t = 2)$ and ensure, through an adaptive approach, that the input prompt remains under 10,000 tokens in length. The maximum length of response is set as 1024. As an illustrative example, when querying: *"Please introduce PaLM in detail, and tell me about related applications."*, the temporary KG structure during the hybrid search is shown in the Figures 2 and 3. Both images represent subgraphs of the same KG, with the input query depicted in blue nodes. The image on the left (Figure 2) showcases only the keyword nodes (in green), while the image on the right (Figure 3) includes the additionally retrieved text blocks (in pink nodes). The edges displayed are those connecting similar keywords directly retrieved from the query (shown as inner circle nodes in the left figure) as well as edges connecting these similar keywords to the keywords obtained via adjacency search (connecting the inner and outer circles in the left figure). While there may be existing edges between the outer circle keywords, they are omitted from the visualization for clarity. The model has a lengthy response which is shown in the Appendix. For those interested in further exploration, all pertinent code and test cases are available at https://github.com/wispcarey/AutoKG.

C. *Efficiency Analyzation*

Given the flexibility in regulating the volume of retrieved information, both the proposed method and the RAG approach can, in theory, support knowledge bases of any size. This means they can encompass any number of text blocks, each subject to the maximum token limit. As outlined in Section II-C, the efficiency of the AutoKG method for automated knowledge graph construction is $O(N \log N)$ when the number of text blocks N is large.

The constructed keyword KG contains M keywords where $M < N$ (empirically, $M \approx 0.1N$). During the hybrid search process, with parameters $(s_0^t, s_1^k, s_1^t, s_2^k, s_2^t)$, the overall time complexity for the search is:

$$O((s_0^t + s_1^k \cdot s_1^t + s_1^k \cdot s_2^k \cdot s_2^t)N) + O((s_1^k + s_1^k \cdot s_2^k)M). \quad (6)$$

For the semantic vector similarity search method to retrieve the same volume of text blocks, the time complexity is:

$$O((s_0^t + s_1^k \cdot s_1^t + s_1^k \cdot s_2^k \cdot s_2^t)N). \quad (7)$$

From the above, it's evident that the time complexity of our hybrid search approach is the same as that of the semantic vector similarity search. For sufficiently large N, both complexities tend towards $O(N)$.

Based on the KG generated from the 40 references of this article, as described in Section IV-B, we perform a hybrid search using parameters $(s_0^t = 15, s_1^k = 5, s_1^t = 3, s_2^k = 3, s_2^t = 2)$. The theoretical maximum number of text blocks that can be searched using this configuration is 60. For comparison, we conduct a semantic vector similarity search for 30 text blocks. Using a query composed of 50 random characters, we carry out both the hybrid search and semantic vector similarity search methods and record the time taken for each (this includes the embedding computation time). After repeating the experiment 100 times, we calculate the average time taken. The hybrid search method had an average duration of 0.0310 seconds, while the semantic vector similarity search took slightly less, with an average time of 0.0305 seconds. This experiment aligns well with our theoretical analysis of the time complexity.

V. CONCLUSION

This paper addressed the inherent challenges faced by semantic similarity search methods when linking LLMs to knowledge bases. Our method, AutoKG, presents a refined and efficient strategy for automated KG construction. In comparison to traditional KGs, the innovative architecture of AutoKG offers a lightweight and simplified version of KG, shifting the focus from specific entities to more abstract keywords and utilizing weighted undirected edges to represent the associations between keywords. Based on the generated KG, our approach harnesses these capabilities by presenting the LLMs with a more interconnected and comprehensive knowledge retrieval mechanism through the hybrid search strategy. By doing so, we ensure that the model's responses are not only richer in quality but also derive insights from a more diverse set of information nodes.

We tested AutoKG with a hybrid search in experimental evaluations. Because of dataset limitations, our tests were mostly qualitative. The outcome highlights the benefits of our method compared to typical RAG methods with semantic similarity search. In summary, AutoKG provides a valuable step to combine knowledge bases with LLMs. It is computationally lightweight and paves the way for more detailed interactions in LLM applications. Moreover, our hybrid search and the semantic vector similarity search have the same order of time complexity.

Further analysis of the *AutoKG* approach requires the identification or creation of an appropriate dataset to evaluate its integration with LLMs. Wang et al. [31] developed their own dataset to evaluate a similar idea to ours. While the evaluation criteria should resemble that of RAG, a more structurally intricate and complex dataset is desired. Another avenue for improvement revolves around keyword extraction. Currently, the method leverages prompt engineering; however, future work could explore fine-tuning larger models or even training specialized models to achieve enhanced results.

ACKNOWLEDGMENT

The authors acknowledge the assistance of ChatGPT-4 in a first draft of the exposition of the manuscript.

Authorized licensed use limited to: University Town Library of Shenzhen. Downloaded on July 30,2024 at 06:36:27 UTC from IEEE Xplore. Restrictions apply.

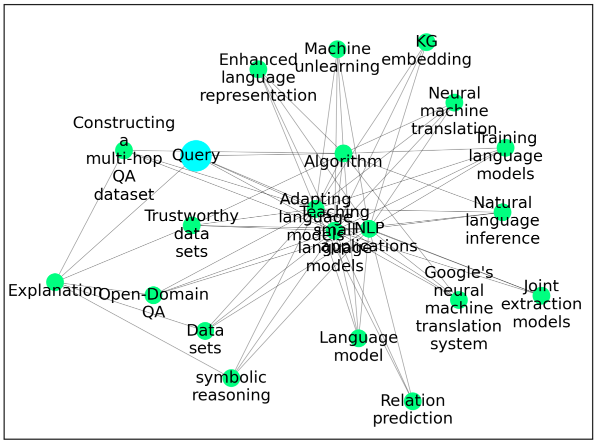

Fig. 2. Subgraph Visualization: Keyword Nodes

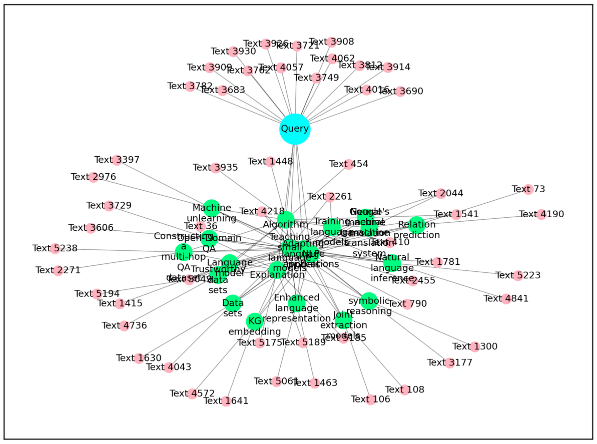

Fig. 3. Subgraph Visualization: Keyword and Text Block Nodes

REFERENCES

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[3] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[4] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, *et al.*, "Palm: Scaling language modeling with pathways," *arXiv preprint arXiv:2204.02311*, 2022.

[5] K. Tirumala, A. Markosyan, L. Zettlemoyer, and A. Aghajanyan, "Memorization without overfitting: Analyzing the training dynamics of large language models," *Advances in Neural Information Processing Systems*, vol. 35, pp. 38274–38290, 2022.

[6] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *International Journal of Computer Vision*, vol. 130, no. 9, pp. 2337–2348, 2022.

[7] S. Welleck, I. Kulikov, S. Roller, E. Dinan, K. Cho, and J. Weston, "Neural text generation with unlikelihood training," *arXiv preprint arXiv:1908.04319*, 2019.

[8] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[9] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?," *arXiv preprint arXiv:1909.01066*, 2019.

[10] T. Scialom, T. Chakrabarty, and S. Muresan, "Fine-tuned language models are continual learners," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, (Abu Dhabi, United Arab Emirates), pp. 6107–6122, Association for Computational Linguistics, Dec. 2022.

[11] G. Mialon, R. Dessì, M. Lomeli, C. Nalmpantis, R. Pasunuru, R. Raileanu, B. Rozière, T. Schick, J. Dwivedi-Yu, A. Celikyilmaz, *et al.*, "Augmented language models: a survey," *arXiv preprint arXiv:2302.07842*, 2023.

[12] A. Asai, X. Yu, J. Kasai, and H. Hajishirzi, "One question answering model for many languages with cross-lingual dense passage retrieval," *Advances in Neural Information Processing Systems*, vol. 34, pp. 7547–7560, 2021.

[13] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.

[14] Y. Luan, J. Eisenstein, K. Toutanova, and M. Collins, "Sparse, dense, and attentional representations for text retrieval," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 329–345, 2021.

[15] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap," *arXiv preprint arXiv:2306.08302*, 2023.

[16] H. He, H. Zhang, and D. Roth, "Rethinking with retrieval: Faithful large language model inference," *arXiv preprint arXiv:2301.00303*, 2022.

[17] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions," *arXiv preprint arXiv:2212.10509*, 2022.

[18] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang, "Improving question answering over incomplete kbs with knowledge-aware reader," *arXiv preprint arXiv:1905.07098*, 2019.

[19] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 2, pp. 494–514, 2021.

[20] J. Zhang, B. Chen, L. Zhang, X. Ke, and H. Ding, "Neural, symbolic and neural-symbolic reasoning on knowledge graphs," *AI Open*, vol. 2, pp. 14–35, 2021.

[21] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, *et al.*, "Never-ending learning," *Communications of the ACM*, vol. 61, no. 5, pp. 103–115, 2018.

[22] B. Abu-Salih, "Domain-specific knowledge graphs: A survey," *Journal of Network and Computer Applications*, vol. 185, p. 103076, 2021.

[23] L. Zhong, J. Wu, Q. Li, H. Peng, and X. Wu, "A comprehensive survey on automatic knowledge graph construction," *arXiv preprint arXiv:2302.05019*, 2023.

[24] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *Lingvisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.

[25] R. Grishman and B. M. Sundheim, "Message understanding conference-6: A brief history," in *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*, 1996.

[26] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[27] J. Ramos *et al.*, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, vol. 242, pp. 29–48, Citeseer, 2003.

[28] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 1003–1011, 2009.

[29] L. Luo, Y.-F. Li, G. Haffari, and S. Pan, "Normalizing flow-based neural process for few-shot knowledge graph completion," *arXiv preprint arXiv:2304.08183*, 2023.

Authorized licensed use limited to: University Town Library of Shenzhen. Downloaded on July 30,2024 at 06:36:27 UTC from IEEE Xplore. Restrictions apply.

[30] G. Wan, S. Pan, C. Gong, C. Zhou, and G. Haffari, "Reasoning like human: Hierarchical reinforcement learning for knowledge graph reasoning," in *International Joint Conference on Artificial Intelligence*, International Joint Conference on Artificial Intelligence, 2021.

[31] Y. Wang, N. Lipka, R. A. Rossi, A. Siu, R. Zhang, and T. Derr, "Knowledge graph prompting for multi-document question answering," *arXiv preprint arXiv:2308.11730*, 2023.

[32] Y. Tian, H. Song, Z. Wang, H. Wang, Z. Hu, F. Wang, N. V. Chawla, and P. Xu, "Graph neural prompting with large language models," *arXiv preprint arXiv:2309.15427*, 2023.

[33] M. Yasunaga, A. Bosselut, H. Ren, X. Zhang, C. D. Manning, P. S. Liang, and J. Leskovec, "Deep bidirectional language-knowledge graph pretraining," *Advances in Neural Information Processing Systems*, vol. 35, pp. 37309–37323, 2022.

[34] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[35] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung, *et al.*, "A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity," *arXiv preprint arXiv:2302.04023*, 2023.

[36] M. Agarwal, P. Sharma, and A. Goswami, "Analysing the applicability of chatgpt, bard, and bing to generate reasoning-based multiple-choice questions in medical physiology," *Cureus*, vol. 15, no. 6, 2023.

[37] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297, Oakland, CA, USA, 1967.

[38] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[39] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, pp. 395–416, 2007.

[40] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 891–923, 1998.

[41] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, p. 912–919, AAAI Press, 2003.

[42] M. Ho, A. Sharma, J. Chang, M. Saxon, S. Levy, Y. Lu, and W. Y. Wang, "Wikiwhy: Answering and explaining cause-and-effect questions," *arXiv preprint arXiv:2210.12152*, 2022.

APPENDIX
DIALOGUES WITH LLM

Dialogue with GPT-4 for Section IV-A

If we only provide information retrieved from the semantic vector similarity search, GPT-4 can not answer the question.

User:

Answer the question "Was it raining this morning when Alex left his home?" with the following information: 1. After leaving his home in the morning, Alex goes to Cafe A to buy a coffee and then takes a bus to Company B for work; 2. Alex bought an iced Americano at Cafe A; 3. Alex talked to the staff at Coffee Shop A about whether they were planning to open a branch; 4. Alex was criticized by his boss at Company B for not completing work on time.

GPT-4:

Based on the provided information, there is no direct evidence or indication about the weather condition when Alex left his home in the morning. Therefore, the answer to the question "Was it raining this morning when Alex left his home?" is "The information provided does not specify."

If we provide information about the clue of weather retrieved from the hybrid search based on AutoKG, GPT-4 can complete reasoning and answer the question correctly.

User:

Answer the question "Was it raining this morning when Alex left his home?" with the following information: 1. After leaving his home in the morning, Alex goes to Cafe A to buy a coffee and then takes a bus to Company B for work; 2. Many people were chatting and drinking coffee in the square outside Cafe A; 3. The car wash located down-stairs of Company B was bustling with business today.

GPT-4:

Based on the provided information, it is likely that it was not raining this morning when Alex left his home. The clue is that "many people gathered in the square in front of the Cafe A, drinking coffee and chatting," which indicates a pleasant outdoor environment, not typical of rainy weather. Additionally, the car wash shop doing very well suggests it might be a clear day since car washes tend to be less busy on rainy days.

Dialogue with GPT-3.5 for Section IV-B

Input query:

Please introduce PaLM in detail, and tell me about related applications.

Preprocess:

With the input query, information is retrieved from the knowledge base through the hybrid search process based on the KG. The prompt is reconstructed with the retrieved information and sent to the *gpt-3.5-turbo-16k* model for the final output.

LLM response:

PaLM, which stands for Pathways Language Model, is a large-scale language model that has been developed to achieve state-of-the-art results in few-shot learning across a wide range of natural language processing (NLP) tasks. It is a dense Transformer language model with 540 billion parameters and has been trained on 780 billion tokens of high-quality, diverse text.

The development of PaLM has been driven by the goal of exploring novel architectural choices and training schemes in order to create a large-scale, modularized system with broad generalization capabilities across multiple modalities. PaLM has been designed to be highly scalable, and it leverages the scaling capabilities of the Pathways infrastructure, which enables training a single model across thousands or tens of thousands of accelerator chips in an efficient manner.

PaLM has demonstrated outstanding performance on various NLP tasks. It has achieved breakthrough results on tasks such as open-domain closed-book question answering, cloze and completion tasks, common sense reasoning, in-context reading comprehension, and more. It has also shown strong capabilities in multilingual tasks and source code generation. In fact, PaLM has outperformed the state-of-the-art on a suite of multi-step reasoning tasks and has even surpassed the

Authorized licensed use limited to: University Town Library of Shenzhen. Downloaded on July 30,2024 at 06:36:27 UTC from IEEE Xplore. Restrictions apply.

average human performance on the recently released BIG-bench benchmark.

The primary application of PaLM is research on language models, including NLP applications such as machine translation and question answering. It is being used within Google for research on a variety of open-ended text and code generation tasks, including reasoning and code synthesis. PaLM has also been evaluated for its translation capabilities across a variety of language pairs and settings, particularly focusing on English-centric language pairs.

It is important to note that PaLM is primarily designed for research purposes and has not been extensively tested in settings outside of research. Further analysis is required before using PaLM for downstream applications. Additionally, ethical considerations and risks associated with large language models, including potential misuse, have been discussed in the context of PaLM.

In conclusion, PaLM is a highly scalable language model that has achieved state-of-the-art results in few-shot learning across a wide range of NLP tasks. Its applications include research on language models, machine translation, question answering, code generation, and more.

APPENDIX
PROMPT EXAMPLES

Task: Keyword Extraction

This prompt requires the following preset variables: 1. ’main_topic’, a high-level topic(s) of the knowledge base; 2. ’l_1’, the maximum number of extracted keywords; 3. ’l_2’, the maximum length of each keyword; 4. ’language’, the language of output keywords; 5. ’previous keywords’, a list of previously extracted keywords. Once the **text blocks** are sampled from a certain cluster, we use the following prompt for keyword extraction:

Prompt for Keyword Extraction:

*You are an advanced AI assistant, specializing in analyzing various pieces of information and providing precise summaries. Your task is to determine the core theme in the following series of *-separated information fragments, which are delimited by triple backticks. Ensure your answer focuses on the topic and avoids including unrelated content. DO NOT write complete sentences.*

You should obey the following rules when doing this task: 1, Keywords in your answer should related to the topic ’main_topic’; 2, Your answer should include at most ’l_1’ keywords; 3, Each keyword should be at most ’l_2’ words long; 4, avoid already appeared theme keywords, marked inside $\langle\rangle$; 5, Write your answer in ’language’; 6, Separate your output keywords with commas (,); 7, Don’t include any symbols other than keywords.

Information:’ ’ ’text blocks’ ’ ’

Please avoid the following already appeared theme terms:
$\langle$’previous keywords’$\rangle$
Your response:

Task: Incorporation between KGs and LLMs

For a given query $\mathbf{q}$, we search for its related text blocks $\mathcal{X}_{\text{final}}$ and keywords $\mathcal{K}_{\text{final}}$ according to the hybrid search algorithm (Algorithm 3). Given a preset variable ’language’ for the output language, we use the following prompt to provide retrieved information from the KG and original knowledge base:

Prompt for Query Response:

I want you to do a task, deal with a query, or answer a question with some information from a knowledge graph. You will be given a set of keywords directly related to a query, as well as adjacent keywords from the knowledge graph. Relevant texts will be provided, enclosed within triple backticks. These texts contain information pertinent to the query and keywords.

Please note, you should not invent any information. Stick to the facts provided in the keywords and texts. These additional data are meant to assist you in accurately completing the task. Your response should be written in ’language’.

Avoid showing any personal information, like Name, Email, WhatsApp, Skype, and Website in your polished response.

Keywords information (directly related to the query or find via the adjacent search of the knowledge graph): $\mathcal{K}_{final}$

Text information: ’ ’ ’ $\mathcal{X}_{final}$ ’ ’ ’

Your task: $\boldsymbol{q}$
Your response:

Authorized licensed use limited to: University Town Library of Shenzhen. Downloaded on July 30,2024 at 06:36:27 UTC from IEEE Xplore. Restrictions apply.