Search Query Refinement using context, knowledge and long-term memorization

***Abstract*—In the era of information overload, search tools are crucial, yet users often struggle to articulate their precise data needs, resulting in sub-optimal search outcomes. This is often due to users associating products with influencers or celebrities rather than knowing the specific brand or product name. This research aims to enable users to find products based on real-life associations, emphasizing the importance of upgrading search query refinement for accuracy and relevance. A significant challenge faced by existing web tools is refining queries involving unrelated entities. This research addresses this gap by proposing a comprehensive approach that integrates context, knowledge representation, and long-term memorization. The framework combines contextual information with advanced knowledge representation techniques, enhancing the system's understanding of user intent and domain-specific concepts. Long-term memorization reduces the time complexity of query refinement by leveraging past search experiences. This study underscores the potential of incorporating context, knowledge representation, and long-term memorization in refining search queries, offering more accurate results. As the digital landscape evolves, our approach has the potential to upgrade the search engine experience, providing personalized and contextually relevant results globally.**

***Index Terms*—Search, Search query refinement, Knowledge Graph**

I. Introduction

In the rapidly evolving digital landscape, search engines have become an indispensable tool, connecting users to an ever-expanding ocean of information [1], [2]. As gateways to the vast realm of the internet, search engines play a pivotal role in assisting users to discover relevant and valuable content with unparalleled convenience [2]. However, despite their remarkable capabilities, these search engines often find themselves grappling with a significant challenge: understanding precisely what the user intends to search for [1]–[3]. The inherent ambiguity of human language, coupled with the diverse intentions and preferences of individual users, poses a formidable hurdle that search engines must overcome to deliver accurate and contextually relevant search results [3].

One of the active areas of research in the realm of search is Knowledge Graph Question Answering. This field involves performing searches against structured knowledge bases using structured queries, typically in the form of subject-predicate-object triplets, where entities serve as vertices and relations as edges. To successfully answer questions posed in natural language, such as "Who is the president of the United States?", it becomes imperative to transform these questions into structured query graphs that can be executed against a knowledge graph. For inquiries involving multiple entities, a generic process of generating a refined search query graph is essential. This refinement process has been the focus of numerous research efforts [4]–[8].

II. Prerequisites

A. Knowledge Graph

A knowledge graph is a powerful knowledge representation framework that organizes information in the form of entities, attributes, and relationships. It provides a structured and interconnected view of knowledge, enabling machines to better understand and reason about the world. Familiarity with knowledge graphs is essential for comprehending how structured data is utilized in search query refinement. Key aspects of knowledge graphs include:

- Entities: In a knowledge graph, entities represent real-world objects, concepts, or individuals. Each entity is uniquely identified and may have various attributes associated with it.
- Attributes: Attributes describe properties or characteristics of entities. For example, a person entity may have attributes like name, age, and occupation.
- Predicates: Predicates define connections between entities and specify the nature of their associations. In a triple format (subject-predicate-object), relationships are represented as edges connecting entities.
- Ontologies and Schemas: Knowledge graphs are often governed by ontologies or schemas that define the types of entities, attributes, and relationships that can exist in the graph. Ontologies impose a structured hierarchy on the graph, enabling more effective reasoning.
- Querying Knowledge Graphs: SPARQL (SPARQL Protocol and RDF Query Language) is a standard query language used to retrieve information from knowledge graphs based on graph patterns.

B. Information Retrieval

Information Retrieval (IR) is a fundamental area of study that deals with the efficient and effective retrieval of relevant

information from large collections of unstructured or semi-structured data. In the context of search engines, IR techniques play a vital role in indexing and organizing vast amounts of web content, documents, and knowledge bases. A solid understanding of IR principles is crucial for comprehending how search engines handle user queries and present relevant results.

Some key concepts in Information Retrieval include:

- Indexing: The process of creating an index of terms or keywords from the documents to facilitate fast and efficient retrieval. Various indexing techniques, such as inverted indexing, are employed to map terms to their corresponding document locations.
- Retrieval Models: Information Retrieval employs different models to rank and retrieve relevant documents or knowledge graph entities based on the user's query. Common retrieval models include the Vector Space Model, Probabilistic Models (e.g., Okapi BM25), and language models.
- Query Processing: The steps involved in processing user queries, which include query parsing, term matching, and ranking of retrieved results based on their relevance to the query.
- Evaluation Metrics: Metrics such as precision, recall, F1 score, and mean average precision (MAP) are used to assess the effectiveness of an IR system's performance.

C. Long-Term Memorization and Context Modeling

Long-Term Memorization is a crucial aspect of artificial intelligence that enables machines to retain and utilize information from the past to make informed decisions in the present and future. In the context of search query refinement, long-term memorization techniques play a vital role in capturing temporal dependencies and context in natural language queries. Understanding these techniques is essential for appreciating how search engines can leverage historical information to better understand user intent and enhance query refinement.

Key aspects of Long-Term Memorization and Context Modeling include:

- Recurrent Neural Networks (RNNs): RNNs are a class of neural networks designed to process sequential data, such as sentences or time-series data. These networks introduce loops within their architecture, allowing them to maintain hidden states that encode past information. By utilizing hidden states, RNNs can capture long-range dependencies in language, enabling better context modeling.
- Long Short-Term Memory (LSTM): LSTM is a specialized variant of RNNs that addresses the vanishing gradient problem, which can hinder the learning of long-range dependencies in standard RNNs. LSTMs incorporate gated mechanisms that control the flow of information within the network, facilitating the storage of relevant context over extended periods.
- Transformer-Based Models: Transformer models have emerged as a transformative advancement in natural language processing. Unlike RNNs and LSTMs, Transformers process entire sequences of data in parallel, making them highly efficient for capturing long-range dependencies. The attention mechanism within Transformers enables them to assign varying levels of importance to different elements in the sequence, effectively capturing context and dependencies between distant words in a sentence.
- Contextual Word Embeddings: Word embeddings, such as Word2Vec, GloVe, and FastText, provide dense vector representations for words based on their co-occurrence patterns in large corpora. Contextual word embeddings, such as those generated by ELMo, GPT, or BERT, take context into account by producing word representations that vary depending on the surrounding words in a sentence. This contextual information enhances the ability to understand the meaning and intent behind complex queries.
- Temporal Attention Mechanisms: Some recent research has focused on introducing temporal attention mechanisms that allow models to attend to specific time steps in the input sequence, thus emphasizing the importance of long-term context in the context modeling process.
- Pre-training and Fine-tuning: Many advanced language models leverage pre-training on large datasets to learn general language patterns and contextual relationships. These models are then fine-tuned on task-specific data, such as question answering or search query refinement, to adapt their understanding to domain-specific requirements.

D. Knowledge Graph Query Structure

Knowledge Graph Question Answering is an active area of research in the field of Search. When performing searches against a Knowledge Graph, the process involves formulating structured queries directed towards a structured knowledge base. This structured format follows the subject-predicate-object (triplet) representation, where entities serve as vertices and relations act as edges. Consequently, to effectively answer a question using a Knowledge Graph, it is necessary to transform the natural language question into a structured query graph, enabling seamless querying of the knowledge base. For instance, a question such as "Who is the President of the United States?" is converted into a triplet representation: (?, president, United States). In this context, the entities in the triplet can be classified into two categories: known or grounded entities, such as "United States," and unknown or ungrounded entities represented by "?" placeholders. Understanding this query structure is essential for grasping how search engines navigate and extract information from knowledge graphs to deliver accurate responses to user queries. Similarly answering a query involving multiple entities, needs extraction of multiple facts to represent the query as structured query graph [9] .

III. Methodology

In the initial stages of information retrieval, a commonly employed technique for enhancing queries was query expansion. This involved broadening the user's original query in an attempt to improve the retrieval count from the information retrieval system, with the goal of increasing the likelihood of discovering relevant documents [10]–[12]. While these methods can be effective, the addition of extra terms to the query may alter its meaning and yield results different from the user's intent. Therefore, when engaging in query modification, contextual awareness becomes crucial. Alternative approaches that incorporate contextual information for query refinement often require access to such information, frequently relying on human feedback to train refinement models and assess their effectiveness [13]–[16]. Despite the introduction of Language Models (LLMs), information retrieval against a query remains unreliable, as LLMs are susceptible to hallucinations and are computationally intensive [17]. In this approach, context and knowledge graph supported information is leveraged to autonomously and promptly refine queries, aiming for a more efficient response to user queries.

The goal of the novel approach is for a given query, create a knowledge graph query structure so that it can easily use the knowledge graph database to find ungrounded entities from the knowledge graph and answer the given query with precision.

The novel approach to query refinement begins with understanding a query. First, the process starts by identifying the context of the query. The context describes what kind of relationship or information is sought between the entities. For example, if the given query is "Which Indian artist won Oscar award for Slumdog millionaire for best original score?", various NLP techniques, similar to news categorisation, are used to find the domain of the query. Domains are like subgraphs that help us to navigate in the vast ocean of Knowledge Graph and narrow down the search parameters. For the given example, query belongs to "Media" domain. Then using POS tagging, it finds out what is the expected output of the query. For the given query, it expects that the result should be of "Artist" entity type. This defines the context and is used throughout the process to facilitate more information.

Once it has a context, next it moves on to query understanding. NER and annotation techniques are used to identify entities, predicates and entity types present in the query. Using context, it identifies the entity type required as answer and marks that as a primary entity types. For given example, it can identify "Oscar" and "Slumdog Millionaire" as entities, "Artist" as entity type and "Indian" and "won" as predicates. From context, it can see that "Artist" is the primary entity type. The pseudo code for query understanding is given in appendix 1.

Once it identifies the entities and predicates, it checks to see if all required predicates are present for the given query or not. If knowledge graph query structure is not missing any predicates, it can simply execute the query on the knowledge graph. For given example, the query structure is depicted by Figure 1.

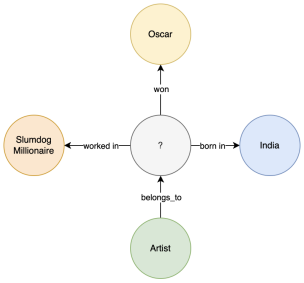

Fig. 1: Knowledge graph query structure for the given example

As seen in Figure 1, no predicates are missing from the query structure, hence it can simply execute the query on the knowledge graph and retrieve the answer.

If there are predicates missing from the query structure, it uses the primary entity type to replace unconnected entity types. For this, lets take a new simple example, lets say the following query is received: "Hrithik Roshan shoes" on an e-commerce app. It identifies the domain as "Retail" and identify the required entity type as "shoes". Then it uses the query understanding algorithm to find that it needs output of "shoes" entity type, hence shoes become the primary entity type. Then, when it finds "Hrithik Roshan" as entity, "shoes" as entity type but when it tries to find predicates it comes up short. Since it

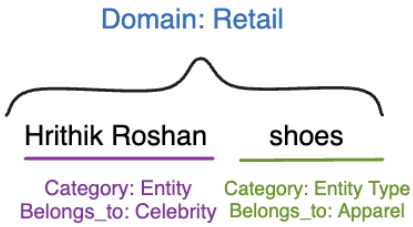

Fig. 2: Query Understanding

doesn't have a predicate connecting the two, it needs to refine the query. The approach refines a query using graph traversal. In the context of knowledge graph, it will refer to primary entities and entity types as primary nodes and non-primary entities and entity types as non-primary nodes. For each non-primary node, it takes a node and try to find all the paths from it to any primary node in the knowledge graph. This can be done by using graph traversal algorithms such as breadth-first search. The paths are represented as lists of nodes and edges. It sorts all the paths by their length (number of edges) in ascending order. The shortest path is at the top of the list and the longest path is at the bottom of the list. For each path in the list, check if the node just before the destination (the last node before reaching a primary node) satisfies the context. If yes, then it returns that node as a result and stop. If not, then it continues with the next path in the list until either a result is found or there are no more paths left. This node is then replaced with the non-primary node from before. This process

keeps happening until all non-primary nodes are replaced. For the example of “Hrithik Roshan” and “shoes”, suppose

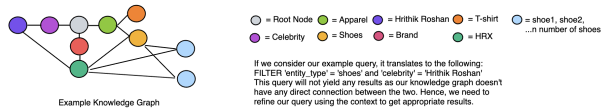

Fig. 3: Sample Knowledge Graph. Considering the example query, it translates to FILTER ’entity type’ = ’shoes’ and ’celebrity’ = ’Hrithik Roshan’. This query will not yield any results as the knowledge graph doesn’t have any direct connection between the two.

one of the path is “Hrithik Roshan” ->“Person” ->“Root” ->“Apparel” ->“Brand” ->“HRX” ->“shoes”. Since “HRX” is a brand connected to “shoes” it will satisfy the context and query will be refined as “HRX shoes”. Pseudo code for Query Refinement is present in Appendix 2. This path is saved as

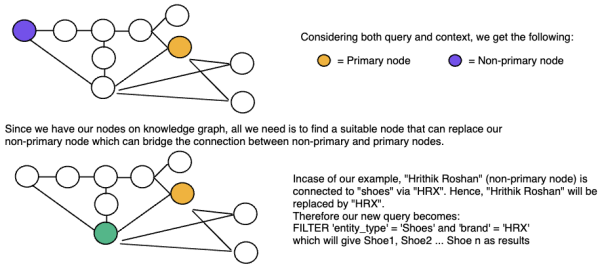

Fig. 4: Query Refinement. For the example, ”Hrithik Roshan” (non-primary node) is connected to ”shoes” via ”HRX”. Hence, ”Hrithik Roshan” will be replaced by ”HRX”. Therefore the new query becomes: FILTER ’entity type’ = ’Shoes’ and ’brand’ = ’HRX’ which will give Shoe1, Shoe2 ... Shoe n as results

part of long-term memorization in a template form to facilitate similar searches in the future. For given example, ”Hrithik Roshan shoes” will be saved as ”belonging to celebrity>type belonging Apparel>” and ”HRX shoes” will be stored as ”belonging to the celebrity>type belonging Apparel>”. Hence in the future if it encounters a similar query say ”Sachin Tendulkar T-shirt”, then it can easily use the previously saved path from ”Hrithik Roshan” to ”shoes” to refine it.

IV. Results

The methodology demonstrates the successful execution of the algorithm and its ability to enhance the accuracy and relevance of search results. To evaluate the performance of the query refinement approach, a diverse and representative dataset of real-world news articles and queries covering a wide range of topics was used. The dataset was manually annotated with section labels, allowing us to assess the precision of the refined queries in aligning with the given context. To test the approach’s effectiveness, standard evaluation metrics were used.

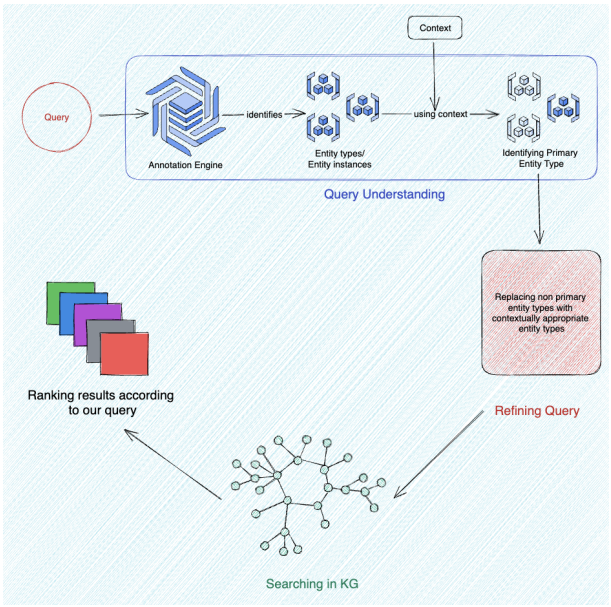

Fig. 5: High Level Design of the Architecture

Testing the approach started with the identification of the context for each query, it applied the query understanding algorithm to identify entities, entity types, and relations within the queries. Subsequently, it verified the connectivity of entities in each query to determine if they were directly linked or required refinement.

For queries with unconnected entities, the query refinement algorithm effectively leveraged knowledge graphs to find all possible paths between primary and non-primary nodes. By systematically evaluating the nodes just before the destination in each path, it successfully replaced unsatisfactory nodes with relevant alternatives that maintained the given context. The iterative process continued until all entities in the query were refined, resulting in contextually accurate and improved queries.

The results of the evaluation demonstrated successful refinement of queries which gave contextually appropriate results are response. In addition to achieving successful results, the methodology showcased notable scalability and efficiency. The query refinement process demonstrated fast execution times even with large-scale knowledge graphs and diverse query contexts. The long-term memorization techniques incorporated in the algorithm allowed for real-time query refinements, making it feasible for practical applications in live search environments.

The evaluation yielded highly promising outcomes, showcasing the successful refinement of queries that delivered contextually appropriate and relevant search results. The implementation of the methodology exhibited commendable scalability and efficiency, overcoming the challenges posed by large-scale knowledge graphs and diverse query contexts. The query refinement process demonstrated exceptional execution times, enabling real-time refinements even in dynamic search

environments.

Moreover, the incorporation of long-term memorization techniques further contributed to the efficiency of the algorithm, empowering it to efficiently handle subsequent searches with improved precision. The ability to leverage historical information for query refinement has the potential to enhance the overall search experience for users.

V. Conclusion

This paper presents a novel and comprehensive approach for search query refinement by integrating context, knowledge graphs, and long-term memorization. The methodology was designed to enhance the accuracy and relevance of search results by analyzing the connectivity of entities in a query, identifying contextual cues, and refining the query based on available information.

The approach began by determining the context of the given query, followed by the application of the query understanding algorithm to identify entities, entity types, and relations. It then assessed whether the entities in the query were directly connected to each other.

For queries where entities were not directly connected, it employed a query refinement algorithm. Leveraging knowledge graphs, it explored all possible paths from unsatisfactory entities to satisfactory nodes, sorting them based on length and considering them as potential routes for query refinement.

To preserve the context during the refinement process, it systematically evaluated nodes just before the destination in each path. Selecting the first path where the preceding node satisfied the context, it replaced the unsatisfactory node iteratively until all entities were aligned with the given context. This iterative approach yielded a refined query that precisely represented the user's intent.

Furthermore, it acknowledged the significance of long-term memorization in facilitating future searches. Hence, the refined path was stored in the long-term memory, enabling more efficient and accurate query refinement in subsequent searches.

By executing the refined query on a knowledge graph, it obtained desired results that exhibited enhanced precision and relevance. By incorporating context, knowledge graphs, and long-term memorization, the methodology significantly improved the overall search experience, leading to increased user satisfaction.

In conclusion, this paper introduced an innovative and effective methodology for search query refinement, showcasing the potential of context-aware techniques and leveraging knowledge graphs to deliver more accurate and contextually relevant search results. The incorporation of long-term memorization further elevated the precision and efficiency of the approach.

VI. Future Work

While the proposed methodology for search query refinement using context, knowledge graphs, and long-term memorization has shown promising results and advancements in the field of search engines, several areas remain open for further exploration and improvement. As the size of knowledge graphs and the volume of user queries continue to grow exponentially, future research can focus on optimizing the scalability and efficiency of the proposed approach. Investigating distributed and parallel processing techniques, as well as exploring ways to reduce computational complexity, would be beneficial in handling large-scale knowledge graphs and real-time query refinements. Expanding the capabilities of context modeling by incorporating dynamic factors, such as user intent evolution over time, browsing history, and real-time situational context, could significantly improve the precision of query refinements. Research in this area could explore techniques like temporal attention mechanisms and recurrent contextual modeling to adapt to ever-changing user needs. Extending the methodology to support multi-lingual query refinement is essential for catering to diverse global users. Investigating cross-lingual knowledge graph alignment and transfer learning approaches could enable more accurate query refinements for users seeking information in languages other than the dominant language of the knowledge graph. Integrating user feedback into the query refinement process can lead to continuous improvement. Developing mechanisms to incorporate user judgments on refined queries and integrating reinforcement learning or active learning techniques could allow the system to learn from user interactions and adapt its refinement strategies accordingly. Enabling personalized query refinement that adapts to individual user preferences, historical search behavior, and interests can further enhance user satisfaction. Research in this direction can explore the integration of user profiling and personalization techniques within the context-aware query refinement framework.

References

[1] Tsai, MF., Wu, YH. User intent prediction search engine system based on query analysis and image recognition technologies. J Supercomput 79, 5327–5359 (2023). https://doi.org/10.1007/s11227-022-04874-w

[2] https://www.searchenginejournal.com/seo/how-people-search/

[3] Liu, X., Guo, W., Gao, H. and Long, B., 2020. Deep Search Query Intent Understanding. arXiv preprint arXiv:2008.06759.

[4] Question Answering with Subgraph Embeddings (https://aclanthology.org/D14-1067) (Bordes et al., EMNLP 2014)

[5] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. 2019. Knowledge Graph Embedding Based Question Answering. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19). Association for Computing Machinery, New York, NY, USA, 105–113. https://doi.org/10.1145/3289600.3290956

[6] Hu, Sen, et al. "Answering natural language questions by subgraph matching over knowledge graphs." IEEE Transactions on Knowledge and Data Engineering 30.5 (2017): 824-837

[7] Zhang, Yuyu, et al. "Variational reasoning for question answering with knowledge graph." Proceedings of the AAAI conference on artificial intelligence. Vol. 32. No. 1. 2018.

[8] Wang, Hongwei, et al. "Knowledge graph convolutional networks for recommender systems." The world wide web conference. 2019.

[9] Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base(https://aclanthology.org/P15-1128) (Yih et al., ACL-IJCNLP 2015).

[10] R. Krovetz and W. B. Croft, "Lexical Ambiguity and Information Retrieval," ACM Transactions on Information Systems (TOIS), vol. 10, no. 2, pp. 115-141 , 1992.

[11] O. Vechtomova, S. Robertson and S. Jones, "Query expansion with long-span collocates," Information Retrieva, vol. 6, no. 2, pp. 251-273, 2003.

[12] G. Cao, J.-Y. Nie, J. Gao and S. Robertson, "Selecting good expansion terms for pseudo-relevance feedback," in SIGIR '08 Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, New York, 2008.
[13] B. Vélez, R. Weiss, M. A. Sheldon and D. K. Gifford, "Fast and effective query refinement," in SIGIR '97 Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval, New York, 1997.
[14] J. Koenemann and N. J. Belkin, "A case for interaction: a study of interactive information retrieval behavior and effectiveness," in CHI '96 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, 1996.
[15] P. Anick, "Using terminological feedback for web search refinement: a log-based study," in SIGIR '03 Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, New York, 2003.
[16] E. Sadikov, J. Madhavan, L. Wang and A. Halevy, "Clustering Query Refinements by User Intent," in WWW '10 Proceedings of the 19th international conference on World wide web, New York, 2010.
[17] M. Shanahan, "Talking about large language models," CoRR, vol. abs/2212.03551, 2022.

APPENDIX

Algorithm 1: Query Understanding Algorithm

Require: query = Input Query, kg = Knowledge Graph

domain = DomainIdentifier(query, kg)
add domain to context
Extracting entities (e), entity types (et) and predicates (p) from given query
e, et, p = AnnotationEngine(query, kg)
Identifying entity type of result (primary entity type)
primary nodes = list()
for $word$ in $query$ **do**
  if $word$ connected to $5W1H$ **then**
    primary node = entity type where $word$ is present
    add entity type to context
  end if
  Add primary node to list of primary nodes
end for
non primary nodes = list(e, et) - primary nodes

Algorithm 2: Query Refining Algorithm

Require: primary node, non-primary nodes, kg

Changing non-primary nodes to contextually appropriate nodes
for $node$ in $non-primarynodes$ **do**
  paths = PathFinder(node, primary node)
  for $path$ in $paths$ **do**
    if path[-2] connected to $primarynode$ and contextually appropriate **then**
      replace non-primary node with path[-2]
    end if
  end for
end for