Conversational Information Retrieval using Knowledge Graphs

Pruthvi Raj Venkatesh[1,*,†], K Chaitanya[2,†], Rishu Kumar[3,†] and P Radha Krishna[4,†]

[1,2,3]*Openstream Technologies,No.25, S.V.Arcade, South End Main Road , 9th Block, Jayanagar, Bengaluru, Karnataka,India, 560069*

[4]*National Institute of Technology Campus, Hanamkonda, Telangana,India,506004*

Abstract

Recent years have seen a huge increase in the popularity of information retrieval(IR) systems, which enable users to hold natural language conversations. IR Systems such as conversational agents are typically goal-oriented and use predefined queries to retrieve information from backend systems. Researchers have improved these agents to adapt to different modalities, such as images, sound, and video, to enhance the conversational experience. Though these systems effectively address users' information requirements, there is a need for an approach that can easily adapt to diverse use cases and meet all user's information needs without the user being aware of the backend system. In this work, we propose a novel approach called Multimodal Information Retrieval System using Knowledge Graph (MIR-KG) to address the information requirement of the user. In the proposed approach, the data surfaced through the conversation agents are stored in a backend database called knowledge graphs (KG). The approach takes multimodal input, uses an offline representation of KG called ontology to identify entities and relations, and generates dynamic KG queries. The paper introduces a context-building technique called Multimodal Context Builder(MCB) to preserve user-provided entities in long conversations and use the ontology to build the KG queries over the context information. We compared our results with a Multi-headed Hierarchical Encoder-Decoder with attention approach and found that the proposed approach gives a more detailed response to user queries. The Training Data Generator (TDG) generates the base training set for setting up the conversation agent. This approach eliminates the time required to collect question and answer pairs needed in the case of goal-based modules. The proposed approach is demonstrated using an already constructed KG with data from the MMD[1] website. The approach can also be applied to other domains..

Keywords

Question Answering System, QnA, Knowledge Graph, Multimodal, Neo4J, Ontology, Entity Identification, Entity masking, Semantic parsing, Logical forms

1. Introduction

Conversational agents(CA) enable users to post questions and get answers. Organizations have recently concentrated on enabling people to engage with these conversational agents

PASIR-2022: First Workshop on Proactive and Agent-Supported Information Retrieval, October 21st, 2022, Atlanta, Georgia

*Corresponding author.

†These authors contributed equally.

pruthvi@openstream.com (P. R. Venkatesh); chaitanya@openstream.com (K. Chaitanya); rishu@openstream.com (R. Kumar); prkrishna@nitw.ac.in (P. R. Krishna)

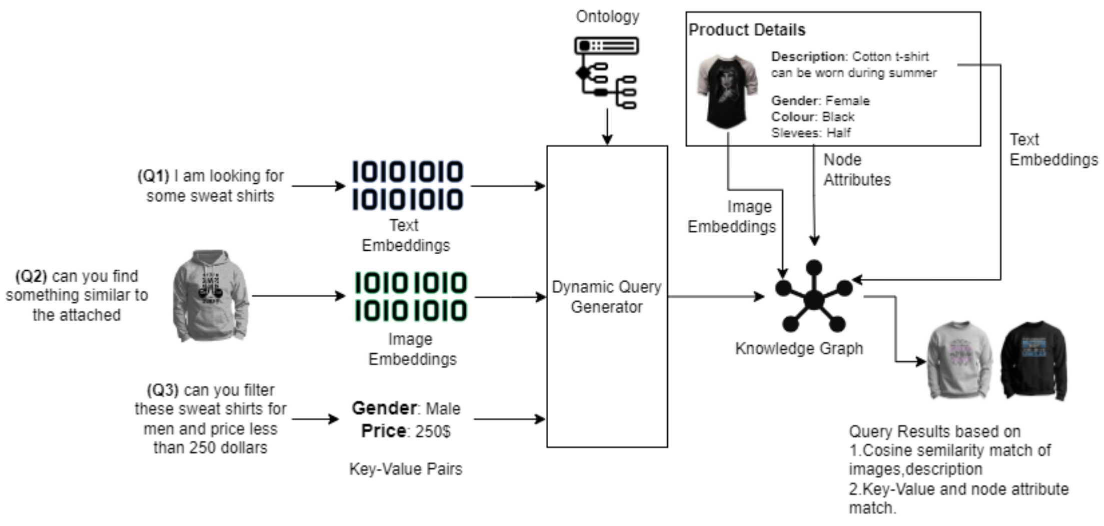

Ontology
Product Details
Description: Cotton t-shirt can be worn during summer
Gender: Female
Colour: Black
Slevees: Half
Text Embeddings
Node Attributes
Image Embeddings
(Q1) I am looking for some sweat shirts
Text Embeddings
(Q2) can you find something similar to the attached
Image Embeddings
Dynamic Query Generator
Knowledge Graph
(Q3) can you filter these sweat shirts for men and price less than 250 dollars
Gender: Male
Price: 250$
Key-Value Pairs
Query Results based on
1.Cosine semilarity match of images,description
2.Key-Value and node attribute match.

Figure 1: Multi-turn IR in the fashion domain.

through multimodal interactions. The approach described in this research, MIR-KG (Multimodal Information Retrieval System utilizing Knowledge Graph), uses ontologies and knowledge graphs to power the question-answering process. The proposed approach combines transformer-based intent and multimodal entity detection from user questions with procedural dynamic query creation. The multimodal entities provided as input to the conversational agent are progressively collected into the conversation context. The dynamic query generation engine uses the context to generate multimodal knowledge graph queries. Multimodal Knowledge Graphs (MMKG)[2] is used to store data and ontology to represent the structural information of the MMKG database. Ontology is used by the Natural Language Understanding (NLU) engine to convert the user question into a logical form. Figure 1 shows an example of a multi-turn information retrieval session where the user is looking for a product and the user is providing text and image input. The user in the first leg(Q1) asks for a product by giving a product description. In the second turn(Q2), the user uploads an image as input; in the third turn(Q3), the user provides the price and gender requirements of the product. The main contribution of this paper is a novel approach for multimodal conversation using MMKG, dynamic query generation using ontologies, and an MCB approach for supporting context-driven information retrieval systems using ontologies.

2. Literature Survey

There has been considerable work in question answering systems that use knowledge graphs. There has also been research on multimodal conversational IR systems. However, little work has been done using KG and transformer-based intent and entity identification models to generate responses in multimodal IR systems. Earlier works in literature [3-7] concentrated on knowledge graph IR through queries generated from a trained model or an intent and query mapping repository. These approaches needed question and query mapping training data and have limited usability in domains with minimal or no training data. The proposed approach does not depend on training data as ontology-based query generation is used, thus making the proposed approach easily implementable in new domains. Literature work [8-9] focused on intermodal representations, which required the development of a dedicated IR module for multimodal search. The proposed approach does not require any complex IR system as the multimodal data is represented as node attributes in the knowledge graph that is queried with other attributes in the node. Literature works [10-15] are not the ideal IR approach for real-time interactions because they only focus on one model representation at a time. The proposed approach does not have these limitations as information can be retrieved from different multimodal fields or text attributes simultaneously.

3. Basic Concepts

3.1. Multimodal Knowledge Graph

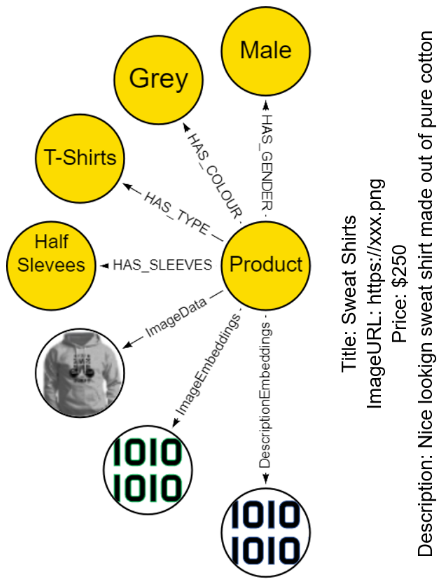
A Multimodal Knowledge Graph(MMKG) is a KG where the attribute set will have values, which are a multimodal representation of data such as text, image, video, sound, human expression, and so on. The representation of data will include binary data, references (image URL, video URL), and embeddings(image embeddings[16], sentence embeddings[17]). Figure 2(a) shows an example of a multimodal knowledge graph where the entities E include Product, Sleeves, Type, Color, and, Gender. Relation R includes HAS_SLEEVES, HAS_TYPE, HAS_COLOUR, FOR_GENDER. The attribute A includes both text and image data. The Product entity has text attributes Title, Price, and Description and Image attributes ImageURL, ImageData, and ImageEmbeddings

3.2. Ontology

The ontology acts as an offline schema definition of the MMKG database. It holds enough information necessary for generating the dynamic query. Ontology is defined as a set $O = \{E_o, R_o, A_o, T_o\}$ where $E_o \subseteq E$ is a set of entities, $A_o \subseteq A$ is a set of attributes. $Ro = \{a_n, r_{nm}, a_m\}$ is a set of relation paths where a_n and a_m are entities in A_o and $r_{nm} \in R$ is the relation between a_n and a_m. T_o ={'Text', 'Numeric', 'Image', 'Embeddings', 'Date'} is a set of data types. Each attribute $a_n \in A_o$ will have a type $t_n \in T_o$. Figure 2(b) shows the entities, attributes, and types. Figure 2(c) shows the relations between the entities.

3.3. Conversational Agent Framework

CA Framework[18] provides a platform for developing and hosting conversational agents, or BOTS. A CA Framework contains templates, standard development kits(SDK), and tools to facilitate the creation of conversational agents. The SDK provides a dialogue or conversation flow feature to manage a long-running conversation. A dialog flow is configurable to perform tasks such as sending messages to users, asking users to enter questions, providing more context, and calling backend API. The conversation flow can be limited to a single or many turns, as



(a) - Multimodal Knowledge Graph

```
"nodes": {                                   "Color": {
  "Product": {                                  "Title": "Text"
    "Title": "Text",                          },
    "ImageURL": "Text",                       "Gender": {
    "Price": "Numeric",                         "Title": "Text"
    "ImageData": "Image",                     },
    "Description": "Text",                    "Sleeves": {
    "ImageEmbeddings":                          "Title": "Text"
"Embeddings",                                 },
    "Description                              "Type": {
Embeddings": "Embeddings"                       "Title": "Text"
    },                                        }
```

(b) – Ontology Nodes

```
"relationtrees": {
    "paths": [
      "(P:Product)-[HT:HAS_TYPE]-(T:Type)",
      "(P:Product)-[HC:HAS_COLOR]-(C:Color)",
      "(P:Product)-[HG:HAS_GENDER]-(G:Gender)",
      "(P:Product)-[HS:HAS_SLEEVES]-(SL:Sleeves)"
    ]
  }
```

(c) – Ontology Relations

Figure 2: Multimodal Knowledge Graph and Ontology.

shown in Fig 1. CA Framework has a persistent store that allows developers to store user-provided information in user variables. User variables are configurable to persist though out the chat session, initialized, reinitialized, or destroyed depending on the user input during the dialogue session. In the example in Figure 1, the dialogue flow is configured to ask the user an initial question(Q1). After Q1 results are provided, the user is asked to provide more context in Q2 and Q3. The dialogue is configured to request more context until the user is satisfied with the results or signals to close the chat.

4. Proposed Approach

This section captures the implementation approach for our proposed Multimodal Information Retrieval System using Knowledge Graph (MIR-KG) and Multimodal Context Builder(MCB) using modalities of text and image. Figure 3 shows the representation of the process flow.

1. The user enters the question or provides more context in a CA that is hosted on a web application or an app
2. CA framework receives the question provided. The dialogue flow triggers the NLU service to determine the intent and entities and produce logical forms. Intents are matched with the utterances generated using the training data generator. (Refer to Sec 4.1)
3. Depending on the identified intent, the dialogue flow either a) Initiates the dynamic query generation module providing the context object collected in user variables, or b) stores the intents and entities identified in the user variable and requests more context by sending a message to the user, such as "can you provide more information."

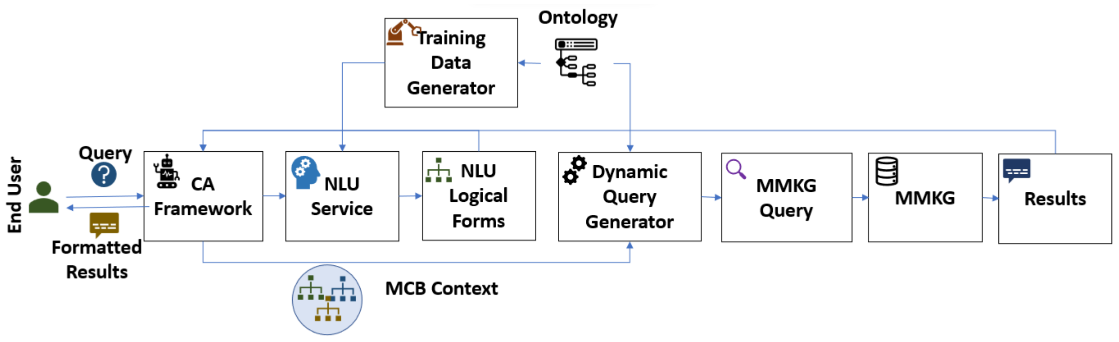


Figure 3: Multimodal Information Retrieval System using Knowledge Graph (MIR-KG) and Multimodal Context Builder(MCB)

4. The context object and the ontology are provided as input to the dynamic query generation module, which creates the query to execute against the MMKG.
5. The results are returned to the CA framework. The CA framework formats the results to show results in a user-friendly format.

4.1. Training Data Generator

The problem with CA based on trained models is obtaining or producing the necessary data set for training the backend models. Though there are approaches such as synthetic data generation[19] used for generating the training data, there is still a requirement of ground truth question-answer pairs which may exist or may take time to gather from subject matter experts(SME). The paper proposes a novel Training Data Generator(TDG) approach to create training data containing question-answer pairs using ontology to address this problem.

The proposed TDG approach generates a set of tuples $T = \{I, Q, E_o\}$ where I is the set of intents, Q is the set of questions, and E_o is the entity in the ontology that will be provided as a query output of question Q. The approach uses a common phrase set $CP = \{P, E_o\}$ where P is the set of common phrases to generate the training data for questions related to entity E_o. The algorithm for TDG is shown in Algorithm 1. For every entity e_n in E_o, the algorithm look for common phrases p_n in P and generates the question q_n obtained by concatenating the string p_n with the entity name of e_n and a where clause w_n. The where clause w_n is obtained by concatenating the string "where" with every combination of attribute $\{e_i, e_j\} \in A_o$ where A_o is the set of attributes defined in the ontology.

The following scenario explains the TDG process for the ontology shown in Figure 2(b). Consider $E_o = \{\text{"}Product\text{"}, \text{"}Color\text{"}\}$, attribute set $a_{product} = \{\text{"}Price\text{"}, \text{"}Description\text{"}\}$ and $a_{color} =$ "$Title$" and $p_{product}$ = {"I am looking for a","show me "}. The training tuples generated for this setup are {"getProduct", "I am looking for a product where product price is #ent and color title is #ent"," Product"}, {"getProduct", "I am looking for a product where product description is #ent and color title is #ent"," Product"}, {"getProduct", "show me product where product price is #ent and color title is #ent"," Product"}, {"getProduct", "show me product, where product description is #ent and color title, is #ent"," Product"}. It can be noted from the example that the TDG approach

Algorithm 1 Training data generator

```
Input: $E_o, a_e$ from Ontology file
Output: List of utterances ($q_{ij}$)
 1: for each $e_i \in E_o$ do
 2:     for each $a_{ie} \in a_e$ do
 3:         for each $e_j \in E_o$ do
 4:             for each $a_{je} \in a_j$ do
 5:                 if $a_{ie} \neq a_{je}$ then
 6:                     $p_e \leftarrow fetch\ common\ phrase\ CP\ from\ entity\ e_i$
 7:                     $i_e \leftarrow concat(\text{"}get\text{"}, e_i)$
 8:                     $w_e \leftarrow concat(\text{"}where\text{"}, e_i, a_{ie}, \text{"}is\ \#ent\ and\text{"}, e_j, a_{je}, \text{"}is\ \#ent\text{"})$
 9:                     $q_{ij} \leftarrow concat(p_e, e_i, w_e)$
10:                 end if
11:             end for
12:         end for
13:     end for
14: end for
```

can generate multiple combinations of questions depending on the entities and attributes present in the ontology. The training tuples generated are used for intent identification by the NLU in Section 4.2.3. This approach thus reduces the overall training time because of the lesser dependency on the base training data

4.2. NLU Service

The NLU service converts the natural language question into a logical form used by the dynamic query generator to generate MMKG queries. The NLU service requires certain intent and entity configurations to perform the following activities 1) Intent identification, 2) Entity identification, and 3) Logical form generation. The NLU service is built on a standard NLP framework such as spacy[20]. Figure 4 shows the process flow of NLU service.

4.3. Intent and Entity Configuration

This process involves configuring the intents and entities the CA system should support. The NLP pipeline will use the configuration to identify entities and intent from the question. The configuration process involves updating the intent(TDG output) and entity information into the NLU service configuration. Entity information include

1. **Synonym Entities**: Configuration to identify entities that have discrete values. The configuration information includes tuples $\{E_t, A_t, V_t\}$ where $E_t \in E_o$ and E_o are the entities in ontology, A_t is the attribute of E_t with value $V_t \in V$ where V is the set of distinct values in MMKG for entity E_t. For the MMKG in Figure 2, the distinct values for Color include {Color, Blue},{Color, Green}. The distinct values for gender include {Gender, Male},{ Gender

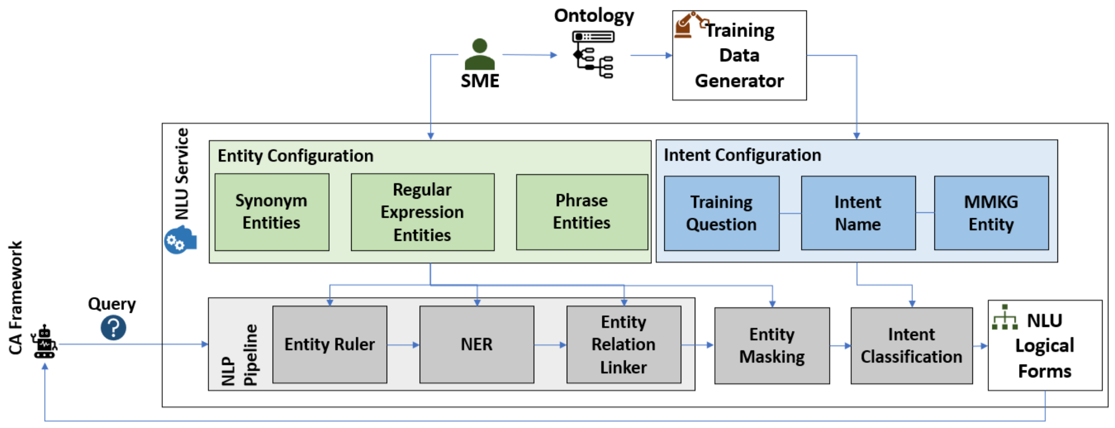


Figure 4: NLU Service Pipeline Training Data Generator.

, Female}, {Gender , Male | Men},{Gender , Female | Ladies}. It can be noted that {Gender, Female | Ladies} also captures synonyms for the value ”Female” as ”Ladies.”

2. **Regular Expression Entities**: Similar to the Synonym entities, this configuration identifies entities with continuous values. In Figure 2, the ”Price” attribute in the ”Product” entity is a continuous variable. ”250$” can be identified using the regular expression [0-9]$.
3. **Phrase Entities**: Configuration to identify entities that are embedded in phrases.

4.4. Entity Identification

Entity recognition uses the NLP pipeline and entity configuration to identify the entities provided as part of the query. The NLP pipeline is a feature provided by the NLP framework and consists of Entiry Ruler, Named Entity Recognition(NER) and Entity Relation Linker. Entity Ruler identifies and marks tokens in the supplied question as entities after comparing them with the configured entities. NER assigns labels to the identified entities, and the Entity relation linker identifies the relationship between the entities and the entity values depending on the relation terms such as ”greater than”, ”lesser than”, ”similar to”, ”having a value”, and so on. The entity recognition module generates quadruple $ER = \{E_o, A_o, V, R\}$ where R is the relation value. For the question, ”can you filter these sweatshirts for men and price less than 250 dollars,” the word ”Male” is identified using Synonym entities, and ”250 dollars” is identified using Regular Expression entities. NER maps ”Male” with the ”Gender.Title” attribute and ”250” with the ”Product.Price” attribute. The Entity relation linker maps entities and attributes with values and generate quadruple {”Gender”,” Title”,” Male”,” =”} and {”Product”,” Price”,” 250”,” <”}.

4.5. Intent Identification

The question sent to the NLU service matches the predefined questions generated by TDG. The proposed approach uses transformer-based models for sentence similarity and identifying the most relevant TDG utterance matching the query passed. If Q is the question passed, the entity masking modules mask the entities identified by the NLP pipeline and generate Q_{mask}. The sentence embedding model[17] generates the sentence vector V_{mask} for Q_{mask}. The sentence vector V_{mask} is compared for similarity with all the questions in tuples $T = \{I, Q, E_o\}$ generated by TDG using the algorithm mentioned in Algorithm 2. The tuple with the maximum similarity score and greater than the minimum threshold t_{min} is identified as the intent of the question. t_{min} is usually set to 0.75 or above. If question Q = "I am looking for t-shirts in red color and price less than 250 dollars", the entity masking module generates Q_{mask} = "I am looking for #ent in #ent color and price less than #ent dollars." The intent identification algorithm identifies the TDG tuple = {"getProduct", "I am looking for a product where product price is #ent and color title is #ent"," Product"} as the intent tuple.

Algorithm 2 Intent Identification

```
Input: $V_{mask}, T$
Output: TDG tuple, $T = \{I, Q, E_o\}$
1: for each $q_i \in T$ do
2:     $q_{vector} \leftarrow getSentenseVector(q_i)$
3:     $q_{iscore} \leftarrow \frac{q_{vector} * V_{mask}}{||q_{vector}|| * |V_{mask}|} = \frac{\sum_{i=1}^{n} q_{ivector} * v_{imask}}{\sqrt{\sum_{i=1}^{n} q_{ivector}^2}\sqrt{\sum_{i=1}^{n} v_{ivector}^2}}$
4: end for
5: $q_{max} \leftarrow max(q_{iscore})$
6: if $q_{iscore} \geq t_{min}$ then
7:     return $t_i = \{I_i, Q_i, E_{oi}\}$
8: else
9:     return unknown intent
10: end if
```

4.6. Logical Form Generation

A logical form is generated containing the output from entity and intent identification. The logical form generated is a set $L = \{Q, ER, t_i, q_{max}\}$. Q is the question asked, ER is the set of all the recognized quadruples, t_i is the entity tuple identified by the intent identification algorithm and q_{max} is the maximum similarity score of the identified intent question with the supplied question.

4.7. MultiModal Context Building(MCB)

The MCB technique uses ontology and entity lists provided by the NLU to generate the context objects. The context object is a collection of entities recognized by the NLU in multiple turns. The CA framework(Refer 3.3) plays a key role in creating and updating the context object. Table 1 shows the working of the MCB approach for the sample conversation shown in Figure 1. The table captures the states and responses of the different components involved in a multi-turn conversation. It can be noted from the table that the CA Framework is configured to control the flow by asking questions shown in the "CA Response" column. The user-provided response is shown in ”User Question/Context”. Once the intent I is identified in the first turn, the CA framework considers only the entities recognized by the NLU engine to build the context object in subsequent turns. If no entities are identified, the CA framework creates the context object by mapping the question provided with the product ”Description” column. The results of the dynamic query generated using the CO will be shown to the user at each turn. If unsatisfied, the user provides more context to filter the results further. The MCB approach thus incrementally builds the context in multiple turns and becomes a convenient option to maintain the context. This approach is not limited by the total number of turns in the conversational flow.

Table 1
MCB Context Generation

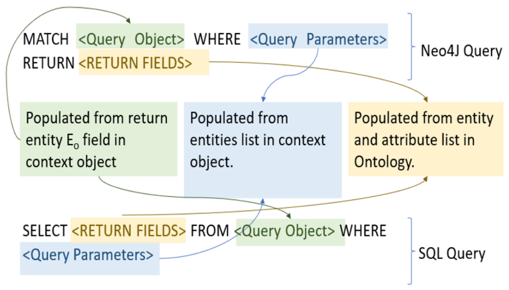
Turn	CA Response	User Question/Context	NLU Entities and Intents identified	Context Object State
1	Please let us know what you are looking for.	(Q1) = I am looking for some sweat shirts	ER = {}, I = “getProduct” , E_o = “Product”	CO = ({{“Product”, ”Description”, Q1, ”Cosine”}}, I = “getProduct” , E_o = “Product”)
2	These are the results, Do you want to provide more filters.	(Q2) = can you find something similar to the attached (user attaches a image of the shirt)	ER = {{“Product”, ”ImageURL”, “http://....png”, ”Cosine”}}	CO = ({{“Product”, ”Description”, Q1, ”Cosine”}, {“Product”, ”ImageURL”, “http://....png”, ”Cosine”}}, I = “getProduct” , E_o = “Product”)
3	These are the results, provide more context to get more filtered results.	(Q3) = can you filter these sweat shirts for men and price less than 250 dollars	ER = {{“Gender”, ”Title”, “men”, ”=”}, {“Product”, ”Price”, “250”, ”<”}}	CO = ({{“Product”, ”Description”, Q1, ”Cosine”}, {“Product”, ”ImageURL”, “http://....png”, ”Cosine”}, {“Gender”, ”Title”, “men”, ”=”}, {“Product”, ”Price”, “250”, ”<”}}, I = “getProduct” , E_o = “Product”)

4.8. Dynamic Query Generator

The dynamic query generation module processes the context object stored in the CA Framework and builds the KG query. Figure 5(a) shows the query mapping between entity fields, return entity, ontology fields, and the query components for Neo4J and SQL. Figure 5(b) shows the dynamic query generation for query Q1 in Figure 1. The dynamic query generation module contains procedural code for handling data types such as embeddings, text, numeric, and date. These rules can be altered depending on the target platform. The query object is the entity whose attributes will be displayed in the results, and this value is fetched from E_0 in the context object. The query’s return fields are obtained from the entity $E_0's$ attribute set in the ontology. The ”where” clause for the question is populated by the entities listed in the context object. Since the dynamic query generator contains generic rules depending on the data types, the approach becomes extensible. This approach requires the creation of ontology to apply to a different domain saving significant effort in training and testing.

5. Results

The proposed approach was tested by ingesting 1743 JSON records from the MMD[1] data into the Neo4J KG database, resulting in 2438 nodes and 12201 relations. The TDG generated 152 utterances, 12 intents, and 38 entities. The average response time for results generation was less than 2 sec. KG data load time was approximately 10 minutes, and embedding generation time

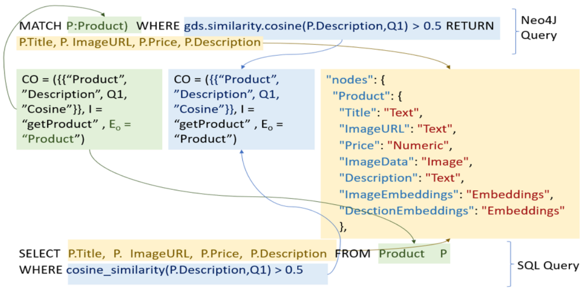


```
MATCH <Query Object> WHERE <Query Parameters>
RETURN <RETURN FIELDS>
```

Neo4J Query
Populated from return entity E$_o$ field in context object
Populated from entities list in context object.
Populated from entity and attribute list in Ontology.
SQL Query

```
SELECT <RETURN FIELDS> FROM <Query Object> WHERE
<Query Parameters>
```

(a) Dynamic Query Generation Rules



```
MATCH P:Product)  WHERE gds.similarity.cosine(P.Description,Q1) > 0.5 RETURN
P.Title, P. ImageURL, P.Price, P.Description
```

```
CO = ({{"Product",
"Description", Q1,
"Cosine"}}, I =
"getProduct" , Eo =
"Product")
```

```
CO = ({{"Product",
"Description", Q1,
"Cosine"}}, I =
"getProduct" , Eo =
"Product")
```

```
"nodes": {
  "Product": {
    "Title": "Text",
    "ImageURL": "Text",
    "Price": "Numeric",
    "ImageData": "Image",
    "Description": "Text",
    "ImageEmbeddings": "Embeddings",
    "DesctionEmbeddings": "Embeddings"
  },
```

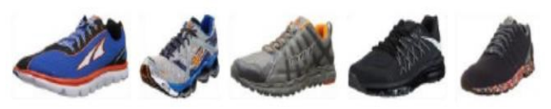
```
SELECT  P.Title,  P.  ImageURL,  P.Price,  P.Description  FROM  Product    P
WHERE cosine_similarity(P.Description,Q1) > 0.5
```

(b) Dynamic Query Generation for Q1

Figure 5: Dynamic generation of KG queries.

was around 2 seconds. The results were compared with Multi-headed Hierarchical Encoder-Decoder with attention(MHRED-attn) results. Table 2 shows the comparison of results for a sample scenario. It can be noted that the user is asking about multiple product attributes, and the MHRED-attn model could not provide the required results. However, the proposed approach returned the product attributes from the KG containing the required information

Table 2
MHRED-attn Vs MIR-KG Response

	**MIR-KG Response** The product details are **Title** : Men Flex Fury Athletic Black Wolf Running Shoes **Description** : A pair of black walking shoes, has regular Styling, lace-ups detail, Mesh upper, Cushioned footbed and Textured and patterned outsole. It has a feet tip and rubber sole. **Price** : 25$ **Image** :
User Query: What is the sole material and tip in the 4th result?	
Gold Response: The running shoes in the 4th image has feet tip rubber sole material.	
Baseline Response (MHRED-attn): The running shoes in the 4th image has rubber sole.	

6. Conclusion

Conversation agents enable users with IR by querying the underlying systems. This paper presents a novel predictive and rule-based approach that reduces training time and supports multimodal conversations using a multimodal knowledge graph. Domain ontology defines entities, attributes, and rules and drives the dynamic query generation module, which helps create conversational IR systems for a new domain with minimal effort.

References

[1] MMD: Towards Building Large Scale Multimodal Domain-Aware Conversation Systems -https://amritasaha1812.github.io/MMD/.

[2] Xiangru Zhu, Zhixu Li, Xiaodan Wang, Xueyao Jiang, Penglei Sun, Xuwu Wang, Yanghua Xiao, and Nicholas Jing Yuan, ”Multimodal Knowledge Graph Construction and Application: A Survey”, 2022, doi: https://doi.org/10.48550/arXiv.2202.05786

[3] H.Jiang, B.Yang, L.Jin and H.Wang, ”A BERT-Bi-LSTM-Based Knowledge Graph Question Answering Method,” 2021 International Conference on Communications, Information System and Computer Engineering (CISCE), 2021, pp.308-312, doi: 10.1109/CISCE52179.2021.9445907.

[4] L.Ma, P.Zhang, D.Luo, M.Zhou, Q.Liang and B.Wang, ”Answer Graph-based Interactive Attention Network for Question Answering over Knowledge Base,” 2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing and Communications, Social Computing Networking (ISPA/BD-Cloud/SocialCom/SustainCom), 2020, pp.521-528, doi: 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00091.

[5] Y.Li, J.Cao and Y.Wang, ”Implementation of Intelligent Question Answering System Based on Basketball Knowledge Graph,” 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2019, pp.2601-2604, doi: 10.1109/IAEAC47372.2019.8997747.

[6] S.Hu, L.Zou, J.X.Yu, H.Wang and D.Zhao, ”Answering Natural Language Questions by Subgraph Matching over Knowledge Graphs (Extended Abstract),” 2018 IEEE 34th International Conference on Data Engineering (ICDE), 2018, pp.1815-1816, doi: 10.1109/ICDE.2018.00265.

[7] X.Dai, J.Ge, H.Zhong, D.Chen and J.Peng, ”QAM: Question Answering System Based on Knowledge Graph in the Military,” 2020 IEEE 19th International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC), 2020, pp.100-104, doi: 10.1109/ICCICC50026.2020.9450261.

[8] S. Zoghbi, G. Heyman, J. C. Gomez, and M-F. Moens. Fashion Meets Computer Vision and NLP at e-Commerce Search. International Journal of Computer and Electrical Engineering (IJCEE), Vol. 8, No 1, pp. 31–43, February 2016.

[9] K. Laenen, S. Zoghbi, and M-F. Moens. Web Search of Fashion Items with Multimodal Querying. In Proceedings of WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining, Marina Del Rey, CA, USA, February 5–9, 2018

[10] S. Bell and K. Bala. Learning Visual Similarity for Product Design with Convolutional Neural Networks. ACM Transactions on Graphics (TOG), vol. 34 , No 4, pp. 1-10, July 2015.

[11] J.-H. Hsiao and L.-J. Li. On Visual Similarity based Interactive Product Recommendation for Online Shopping. 2014 IEEE International Conference on Image Processing (ICIP), pp. 3038-3041, 2014.

[12] B. Zhao, J. Feng, X. Wu, and S. Yan. Memory-Augmented Attribute Manipulation Networks for Interactive Fashion Search. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017). pp. 6156–6164. 2017

[13] X. Han, Z. Wu, P. X. Huang, X. Zhang, M. Zhu, Y. Li, Y. Zhao, and L. S. Davis.Automatic Spatially-Aware Fashion Concept Discovery. 2017 IEEE International Conference on Computer Vision (ICCV), pp. 1472-1480, 2017

[14] C. R. Sapna, M. Anagha, K. Vats, K. Baradia, T. Khan, S. Sarkar, and S. Roychowdhury. Recommendence and fashionsence online fashion advisor for offline experience. ACM International Conference Proceeding series, pp. 256–259, 2019

[15] A. Paranjape, A. See, K. Kenealy, H. Li, A. Hardy, P. Qi, K. R. Sadagopan, N. M. Phu, D. Soylu, and C. D. Manning. Neural generation meets real people: Towards emotionally engaging mixed-initiative conversations. Stanford NLP, 3rd Proceedings of Alexa Prize, arXiv:2008.12348, 2020
[16] Vision Transformer -https://en.wikipedia.org/wiki/Vision_transformer
[17] Sentence Embedding -https://en.wikipedia.org/wiki/Sentence_embedding
[18] Bot Framework SDK -https://docs.microsoft.com/en-us/azure/bot-service/bot-service-overview?view=azure-bot-service-4.0
[19] Synthetic Data -https://en.wikipedia.org/wiki/Synthetic_data
[20] Language Processing Pipelines · spaCy -https://spacy.io/usage/processing-pipelines