Int. j. inf. tecnol. (March 2024) 16(3):1605–1615
https://doi.org/10.1007/s41870-023-01692-y

ORIGINAL RESEARCH

Explicable knowledge graph (X-KG): generating knowledge graphs for explainable artificial intelligence and querying them by translating natural language queries to SPARQL

Numair Shaikh[1] · **Tavishee Chauhan**[1] · **Jayesh Patil**[1] · **Sheetal Sonawane**[1]

Received: 6 October 2023 / Accepted: 9 December 2023 / Published online: 21 January 2024

Abstract Knowledge graphs represent a potent instrument for the classification and exhibition of data, as they encompass a systematic approach for the containment and retrieval of multifarious datasets. In finance, the utilization of knowledge graphs for the organization of company-oriented data constitutes an invaluable source of insights, thus enabling informed decision-making. In a parallel, knowledge graph systems centered on COVID-19 within the healthcare sphere may assist medical professionals in the making of resolute choices. These applications highlight knowledge graphs' ability to revolutionize decision-making procedures by providing a comprehensive comprehension of the given subject. To tackle this, we propose a solution that begets and implements knowledge graphs in two separate domains: finance and healthcare. To ensure the creation of explicable AI systems and improve the accessibility of information within these knowledge graphs, we introduce the conversion of natural language queries into SPARQL queries. By fine-tuning our model, we illustrate the system's superior performance. Furthermore, we appraise the adequacy of the constructed knowledge graphs and contrast them with widely employed alternatives. Our work accentuates the adaptability of the proposed solution, as it can operate seamlessly with diverse datasets requiring minimal modifications.

Keywords Explainable artificial intelligence · Natural language processing · Knowledge graphs · Query translation · Q& A systems

1 Introduction

Recent research has exhibited remarkable success in utilizing Knowledge Graphs across various domains, particularly in the realm of Explainable Artificial Intelligence (X-AI), in order to augment transparency in complex systems. This carries significant implications for fields such as finance and healthcare. The rapid expansion of data across numerous disciplines necessitates effective strategies for extracting useful information. Informed decision-making requires access to and retrieval of pertinent data, especially in critical domains like healthcare and finance.

A Knowledge Graph refers to a semantic network of real-world entities (objects, instances, principles, or events) that illustrate relationships among them. For instance, consider entities such as "Banana", "Yellow", and "Fruit". The relationships between them establish that a "Banana" is both a variety of "Fruit" and is characterized by the attribute "Yellow". This simplified Knowledge Graph demonstrates how entities are interconnected via relationships and attributes, facilitating a clear visualization of their associations. Explainable Artificial Intelligence (X-AI) is a branch of Artificial Intelligence (AI) that strives to render the decision-making process of AI models more comprehensible and interpretable for humans. Traditional AI models,

N. Shaikh, T. Chauhan, J. Patil and S. Sonawane have contributed equally to this work.

✉ Numair Shaikh
numairsh77@gmail.com

Tavishee Chauhan
chauhantavi@gmail.com

Jayesh Patil
patiljayeshsunil@gmail.com

Sheetal Sonawane
sssonawane@pict.edu

[1] Department of Computer Engineering, SCTR's Pune Institute of Computer Technology, Dhankawdi, Pune, Maharashtra 411043, India

particularly deep learning algorithms, often function as "black boxes", making it difficult to discern how a given input leads to a particular output. This system endeavors to provide insights into why a particular decision was made, particularly in vital domains such as healthcare, finance, and legal systems. Numerous existing efforts concentrate on rendering AI systems interpretable for specialists possessing machine learning and related fields expertise. These specialists are capable of altering the mathematical functions within complex algorithms to enhance their transparency. While this approach is effective for experts, it may not be suitable for non-experts who lack technical proficiency. In the context of X-AI, Knowledge Graphs offer an alternative, easier-to-understand technique than mathematical function manipulation. Rather than delving into the internals of algorithms, Knowledge Graphs provide a means of representing the underlying concepts and relationships that AI systems utilize to make decisions. By this approach, we aim to bridge the gap between how AI systems function and how humans comprehend complex information. However, the querying of these knowledge graphs typically necessitates technical expertise, as it requires the utilization of query languages such as SPARQL (SPARQL Protocol and RDF Query Language). SPARQL is a query language specifically crafted to query RDF (Resource Description Framework) data, which is the foundation of knowledge graphs. RDF, an acronym for Resource Description Framework, is defined by the World Wide Web Consortium (W3C) recommendation as a method for representing information as a set of triples. Each triple comprises a subject, a predicate, and an object. The subject is a Uniform Resource Identifier (URI) that identifies a resource, the predicate is a URI that describes the relationship between the subject and the object. The object can be a value that is a URI, a literal, or another RDF triple. SPARQL provides a standardized and expressive syntax for the retrieval of data by leveraging the underlying graph structure. Nonetheless, it can prove to be quite challenging for non-technical users who lack proficiency in query languages. To tackle the challenge of effectively writing SPARQL queries, we offer the conversion of natural language queries into SPARQL queries, thereby bridging the gap between non-technical users and the wealth of knowledge graphs content by streamlining the data retrieval process. This enables a broader variety of stakeholders to benefit from the insights offered by the COVID-19 hospitalization data and the Stock Market data, allowing users to engage with knowledge graphs using natural language.

This study outlines a methodology for the construction of Knowledge Graphs and the answering of Natural Language questions for both the stock market and COVID-19 hospitalization data, thereby improving model transparency and justifications for both domains.

2 Literature review

This literature survey delves into the diverse methodologies employed in knowledge graph construction and the translation of natural language queries to SPARQL.

2.1 Construction of knowledge graph

Gupte et al. [1] introduced an attention-RNN and transformer (BERT) model to convert textual queries into SPARQL "INSERT" queries for knowledge graph creation. The model was trained and evaluated on the DBpedia dataset, yielding compelling outcomes. Wang et al. [2] in their research, presented, a framework for constructing COVID-19 lature knowledge graphs (KGs) and generating drug repurposing reports is proposed. The knowledge graph is built by extracting fine-grained multimedia knowledge elements (entities and their visual chemical structures, relations, and events) from scientific literature. The paper also explores a Q &A method over the knowledge graph that extracts entities and generates a subgraph covering paths between them. Kejriwal [3], in their book, comprehensively cover all aspects of constructing knowledge graphs for specific domains. The book encompasses the entire knowledge graph construction process, from data collection and cleaning to evaluation and deployment. The authors explain that knowledge graphs are a powerful tool for representing and reasoning about knowledge in a domain, and the construction process can be divided into four steps: data collection and cleaning, knowledge extraction, knowledge integration, and knowledge evaluation. The authors explore rule-based, machine learning, and hybrid methods of knowledge graph creation. Li et al. [4] present a systematic approach for constructing a medical knowledge graph (KG) from electronic medical records (EMR). The process involves multiple steps, such as entity recognition, relation extraction, and graph embedding, using a novel quadruplet structure. The results show how the knowledge graph constructed significantly improves understanding of disease, disease classification, and decision-making. Ye et al. [5] review recent progress in generative knowledge graph construction (KGC). The paper comprises two main paradigms for generative knowledge graph construction: relation extraction and knowledge graph completion. Relation extraction methods focus on extracting new relations from textual data, while knowledge graph completion methods focus on filling in missing triples in a knowledge graph. Results from their research show that relation extraction methods are generally more effective than knowledge graph completion methods, however, knowledge graph completion methods tend to be more scalable.

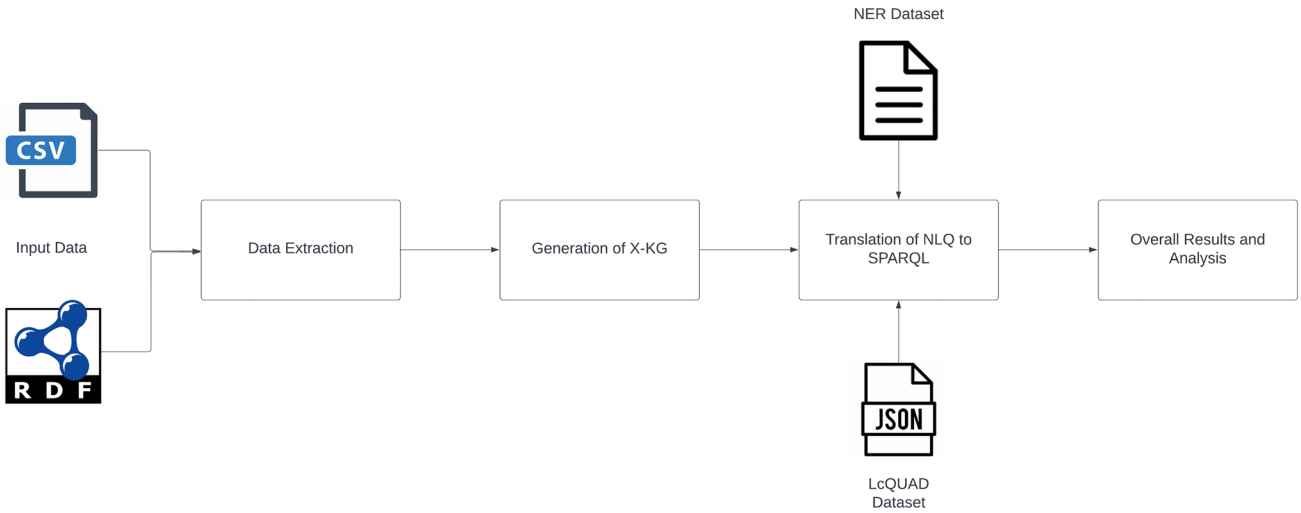

Fig. 1 Architecture diagram of the proposed system

2.2 Translation of NLQ to SPARQL

Dubey et al. [6] proposed a framework, AskNow, a methodology for translating natural language queries into SPARQL. The process involves normalizing the queries into an intermediary canonical syntactic form, known as Normalized Query Structure (NQS), which facilitates the identification of the expected output information and the user-provided input information, and establishes their mutual semantic relationship. Yin et al. [7] suggested an approach that involves tokenizing and encoding the input queries, and then training a neural machine translation (NMT) model to map the tokens from the natural language queries to the tokens from the SPARQL queries. However, this approach only supports the "SELECT" type of queries and not other types. Bhajikhaye [8] proposed using a tree LSTM model with a dependency parse tree in English to classify the different query templates in the LcQuad dataset. Word Disambiguation and Named Entity Recognition (NER) were used to identify query entities, and map them to the template using the Falcon 2.0[1] library, with a specific focus on Wikidata. Ferré [9] proposed SPARKLIS, which uses faceted search, entity linking, schema matching, and query optimization to generate SPARQL queries. However, it is limited by the dataset of NLQs on which it was trained and may not handle queries specific to other knowledge graphs. Ochieng [10] proposed PAROT, which uses dependency parsing to extract triples from natural language questions, and then uses these triples to generate SPARQL queries. The framework was evaluated on a dataset of simple and complex natural language questions, and was found to be able to translate 80% of the questions into correct SPARQL queries. Finally, in their work, Chen et al. [11] employed a two-stage Maximum Entropy Markov Model (MEMM) model for entity type identification and RDF type identification to generate SPARQL queries on the QALD[2] dataset.

3 Proposed methodology

In this section, we elaborate on the system's ability to effectively generate an eXplicable Knowledge Graph while simultaneously extracting its constituent elements through the usage of natural language queries (NLQ). The research presented in this paper is divided into two distinct parts (Fig. 1):

1. The development of a comprehensive eXplicable Knowledge Graph (X-KG)
2. The conversion of natural language queries into SPARQL queries.

3.1 Overview of the datasets

The following section offers a brief overview of the data employed for the purpose of constructing and training our models.

3.1.1 X-KG generation dataset

The X-KG is created using a combination of two datasets, namely

1. Stock Market data
2. COVID-19 Hospitalization data

[1] https://falcon.readthedocs.io/en/2.0.0/index.html.

[2] https://github.com/Perevalov/qald_9_plus.

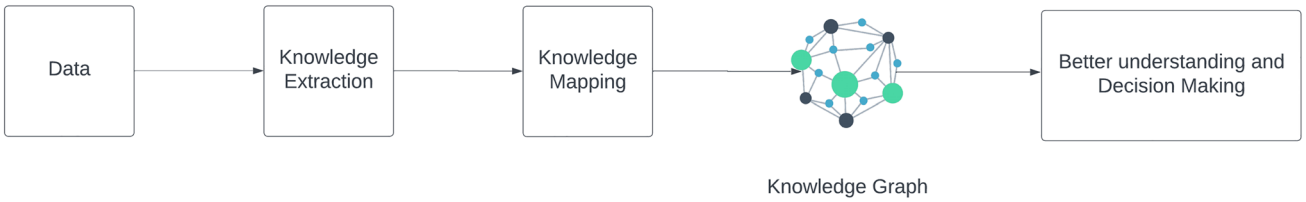

Fig. 2 Sneak-peek into the eXplicable Knowledge Graph (X-KG)

Stock Market Data The Stock Market data utilized in this study was sourced from a variety of online platforms such as DBpedia[3] and Crunchbase.[4] It includes data on prominent IT companies and banks, and is available in a range of formats including RDF data, text and CSV files.

COVID-19 Hospitalization Data This data is sourced from the Covid-19 hospitalization network, as documented by [12]. The primary objective of the COVID-19 Hospitalization Surveillance Network (COVID-NET)[5] is to conduct comprehensive surveillance of hospitalizations linked to confirmed COVID-19 cases across various age groups, encompassing both pediatric and adult populations.

3.1.2 NLQ to SPARQL translation dataset

We have used the LcQuad[6] dataset for the training and fine-tuning of our models.

3.2 Generation of eXplicable Knowledge Graph (X-KG)

The knowledge graph comprises three essential components: nodes, edges, and labels. It serves as a tool for end-users of the system, as well as any individual, seeking to comprehend the results obtained by the underlying system and connect them with the possible reasons and justifications outlined in the diagram. Furthermore, it enables the mapping and elucidation of relationships with diverse ontologies. The benefits of knowledge graph creation and implementation are visualized in Fig. 2. The knowledge graph is developed from multiple datasets from various sources, which need not necessarily possess identical structures for X-KG implementation. The proposed knowledge graph is elaborated through the use of algorithm 1. Subsequently, the requisite data is

Algorithm 1 eXplicable-Knowledge Graph (X-KG)

Input: Classified model results
Output: Comprehensive knowledge graph

1: Start
2: Data Integration
3: Extract data from data source X, Y.
4: Integrate and map the knowledge to the given taxonomies $(x_1, x_2, x_3, .., x_n)$ of X data source and Y data source if required
5: **if** *taxonomy* in *datasource* : **then** X, Y
6: Pre-process the data
7: combine X, Y : $(X \in Y) or (Y \in X)$
8: Map relationships with ontologies $(X \cup Y)$
9: Combine and map knowledge
10: Append the knowledge graph G
11: Post-processing the Sample S.
12: **end if**
13: End

[3] https://www.dbpedia.org/.
[4] https://www.crunchbase.com/.
[5] https://www.cdc.gov/coronavirus/2019-ncov/covid-data/covid-net/purpose-methods.html.
[6] https://github.com/AskNowQA/LC-QuAD.

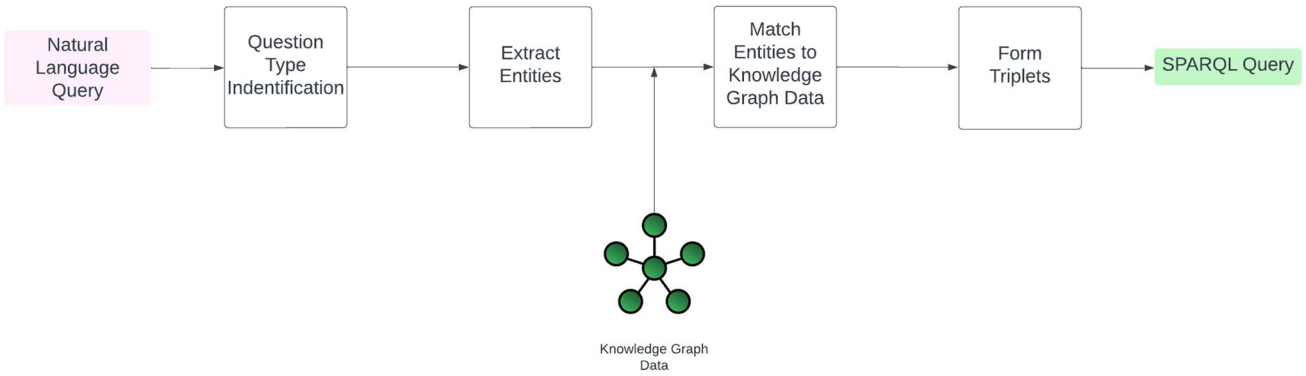

Fig. 3 Architecture diagram for Translation of NLQ to SPARQL

extracted from the desired source and mapped to the applicable data source taxonomy. The details of constructing X-KG are represented mathematically in Eq. 1.

$$f(k) : \sum_{x,y=1}^{n} (X \in Y | Y \in X) \tag{1}$$

In Eq. 1, $f(k)$ is the summation of elements in the data source X and Y, such that elements of data source X belong to elements of data source Y so that a relationship can be built across with ontologies.

$$G = f(X(x_1, x_2, .., x_n) + Y(y_1, y_2, .., y_n)) + k(X \cup Y) + \delta(S) \tag{2}$$

While, from Eq. 2, G represents the knowledge graph, $f(X(x_1, x_2, .., x_n) + Y(y_1, y_2, .., y_n))$ creates a function for data integration. $k(X \cup Y)$ represents the mapping of knowledge function with original sources. The $\delta(S)$ is for examining the sample S and its post-processing, where we identify areas requiring manual intervention.

3.3 Translation of NLQ to SPARQL query

This section delves into the translation process of natural language queries to SPARQL queries, thereby bridging the gap between non-technical users and the vast knowledge graphs' content. This streamlined data retrieval process enables a broader range of stakeholders to gain insights offered by the X-KG, allowing users to interact with knowledge graphs through natural language. The flexibility of our approach is a primary motivation behind this research, as it can be effortlessly implemented for different types of knowledge graphs with minor adjustments to the training dataset. The benefits of converting natural language queries into SPARQL queries can be utilized in various domains outside the healthcare and stock market (finance), which are the primary areas of focus in this research, thanks to the versatility offered. Our system integrates domain-specific training data to comprehend the distinct structure and language of various knowledge graphs, enabling it to be efficient and applicable in a variety of settings. Moreover, our solution's potential impact is amplified by its adaptability to the data retrieval needs of different businesses and disciplines, thereby increasing knowledge graphs' accessibility and usability. This work concentrates on simple (single-level) SPARQL queries.

The process of transforming Natural Language Input into SPARQL Queries entails a significant contribution that can be succinctly summarized in the following series of steps:

1. Identification of question/query type.
2. Extraction of Entities from input query.
3. Fuzzy matching of the Entities extracted.
4. Formation of triplets and SPARQL query generation.

A visual representation of the entire pipeline for NLQ to SPARQL can be seen in Fig. 3.

3.3.1 Identification of question/query type

The initial aspect of the system involves recognizing the types of questions to ensure efficient query classification. In this study, we have concentrated on three specific categories of labeling for classification.

1. SELECT
2. COUNT
3. BOOLEAN (Yes/No)

The LcQuaD dataset comprises an extensive compilation of questions and corresponding answers, with each question being linked to a SPARQL query. The SPARQL queries in the LcQuad dataset are divided into three categories, namely SELECT, COUNT, and BOOLEAN. To classify the queries, we generate labels based on the type of SPARQL query used for a specific question. To modify the dataset, we introduce

Table 1 Comparison of supervised classification models for question type classification

Model name	Accuracy	Precision	Recall
Decision Tree	92.0	84.75	84.90
SVC	91.8	87.80	83.76
XGBoost	92.40	86.26	85.32
DistilBERT (SoTA)	**99.76**	**99.68**	**99.60**

a label column that is assigned type 2 if the SPARQL query contains the term "BOOLEAN". Similarly, if the SPARQL query contains the term "COUNT", we assign type 1 to the label column. If neither term is present in the query, the label column is marked as type 0. Prior to model training, the subsequent pre-processing measures are implemented upon the textual data to improve the caliber and effectiveness of the ensuing analysis.

1. White Space Removal.
2. Converting text to Lowercase
3. Stemming and Lemmatization

By implementing these preprocessing measures, the textual data is standardized, extraneous words are eliminated, and words are converted to their fundamental or core forms.

To classify triplet entities, we utilize and compare various models including RandomForestClassifier [13], Support Vector Machines (SVM) [14], and XGBoost [15]. In addition, we have attempted to incorporate State-of-The-Art (SoTA) deep learning models such as BERT [16] for classification purposes. For classical machine learning algorithms like RandomForestClassifier, SVM, and XGBoost, a critical step in preparing the data for training the models is vectorization. Vectorization involves converting textual data into numerical representations, which can be processed by these algorithms. In our approach, we utilized the TF-IDF [17] vectorizer for this task. We conducted an experimental study utilizing the state-of-the-art (SoTA) model, DistilBERT [18], which is a lightweight variant of the SoTA model, BERT, in order to compare it with other models. Unlike traditional approaches that involve the use of the TF-IDF vectorizer, SoTA models such as DistilBERT employ advanced techniques like tokenization and attention mechanisms to process input text. The built-in tokenizer is utilized to produce input IDs and attention masks, which effectively capture the information and structure of the text, rendering external vectorization methods unnecessary. This methodology harnesses the power of pre-trained language models to capture semantic meaning and contextual relationships within the text, thereby obviating the need for explicit vectorization using TF-IDF. In Table 1, using the LcQuad dataset, we compare the performance of various classification models for predicting question types. From the information presented in Table 1, it can be deduced that the SoTA model, DistilBERT, has demonstrated a noteworthy level of precision in achieving the classification objective at hand. As a result, we have opted to utilize DistilBERT as our model for the identification of question types.

3.3.2 Extraction of entities from input query

In this step, a pipeline for Named Entity Recognition (NER) is executed. To augment its efficacy and address our specific requirements, we have trained this model using a personalized dataset. Our customized dataset will comprise the anticipated question types for a certain knowledge graph, in addition to all the subject, predicate, and object entities marked. In the NER pipeline, three pivotal phases are involved:

1. Pre-Processing
2. Dependency Parsing
3. Entity Extraction

The NER pipeline is depicted in Fig. 4, which visualizes the operations executed in every phase.

Pre-Processing In this step, the pipeline performs various preprocessing steps to obtain a clean textual query, such as

1. Tokenization
2. Lemmatization
3. Stop Word Removal

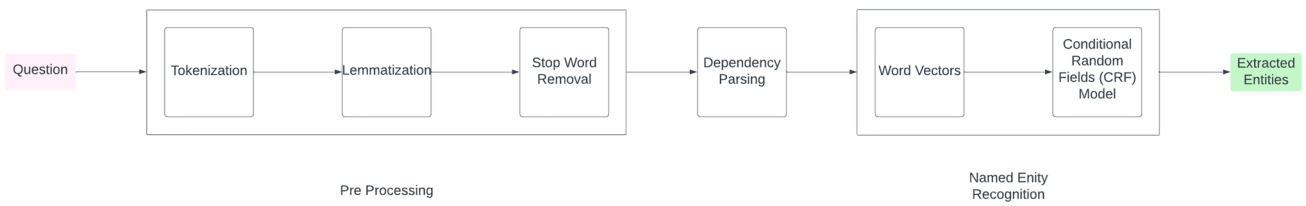

Fig. 4 Named entity recognition pipeline

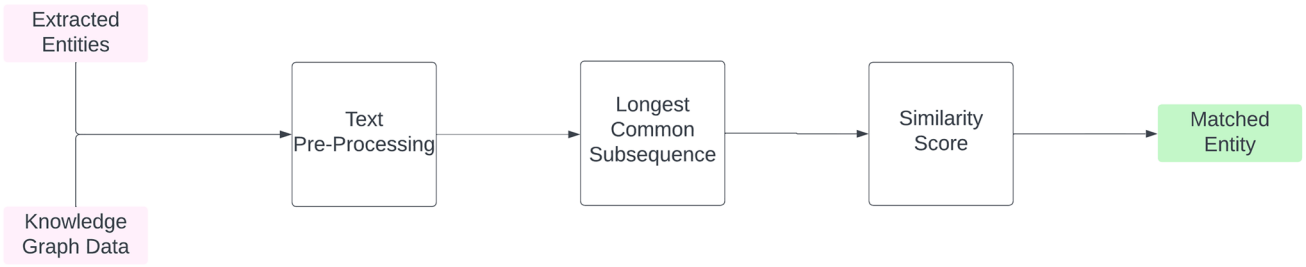

Fig. 5 Fuzzy entity mapping of extracted entities with X-KG data

Dependency Parsing Dependency Parsing constitutes a crucial step in the context of Named Entity Recognition (NER) as it facilitates the identification of the contextual significance of a word. Specifically, if a word is dependent on a verb, it is more likely to be a noun. This step entails the acquisition of knowledge pertaining to diverse linguistic features and patterns, thereby enabling the production of accurate predictions concerning the syntactic relationships among the constituent words in a sentence. To perform Dependency Parsing, we employ spaCy[7]'s neural network transition-based parser. This approach to dependency parsing involves the prediction of a sequence of parsing actions that collectively construct a dependency tree for a given sentence. These actions may entail the shifting of the next word onto the stack, the creation of a dependency between words, or the reduction of words from the stack. Subsequently, the model acquires the ability to predict the most appropriate action at each parsing step based on the current state of the stack, buffer, and previously generated dependencies. This stage provides valuable information regarding the grammatical structure of a sentence and the syntactic relationships between the constituent words. This contextual knowledge proves useful in determining the boundaries of named entities during the NER step.

Entity Extraction This step is further divided into two steps:

1. Conversion of Input into Word Vectors.
2. Sequence Labelling using Conditional Random Fields (CRF) Model

In the initial stage, the input text undergoes a conversion process to produce the corresponding Word Embedding. These embeddings are compact and reduced-dimensional representations of words that effectively capture both semantic and syntactic similarities among them. To achieve this conversion, pre-trained models such as Word2Vec are utilized. The output of this first step, which is a dense vector, is subsequently employed as input to the CRF Model. The CRF model, as described in the work of [19], is utilized for the purpose of labeling words/sequences. Specifically, this model is highly effective in the task of Named Entity Recognition (NER), which entails the identification and classification of named entities. By integrating Dense Vectors into the CRF model, the model is able to learn and harness semantic information about words, thus enhancing its ability to accurately identify entities based on their contextual meaning.

3.3.3 Fuzzy entity matching

Upon extraction of the Entities from the Input text, our approach employs the utilization of fuzzy matching to correspond the Extracted Entities with the data housed within our Knowledge Graph (X-KG) database. Fuzzy matching is a technique that facilitates the comparison of strings for the purpose of determining their similarity, even in the presence of minor deviations or variations within the text. This process is further partitioned into three sub-steps, namely:

1. Pre-Processing of Text
2. Determining the Longest Common Subsequence
3. Generating the Similarity Score

The Fuzzy Entity Mapping process, together with the operations conducted, is illustrated in Fig. 5.

Pre-Processing Prior to comparison, we perform various preliminary preprocessing activities on the input. These activities encompass converting the strings to lowercase, removing leading/trailing white spaces, and eliminating duplicate spaces.

Longest Common SubSequence The utilization of the longest common subsequence algorithm, which leverages dynamic programming, enables the discovery of the longest common substring (LCS) between the input sequences.

Generating Similarity Score Similarity score is generated by using the longest common subsequence obtained.

It can be mathematically represented as

[7] https://spacy.io/.

Table 2 Comparison of the structural quality metric evaluations for the knowledge graphs generated by X-KG and other popular knowledge graphs

Metric	X-KG (Stock Market)	X-KG (COVID-19 Hospitalization)	Raftel	Wikidata	Google KG
Instantiated class ratio	0.928	1	0.941	0.004	0.099
Instantiated class ratio	0.934	0.228	1	1	0.660
Class Instantiation	1	0.577	0.941	0.716	0.660
Inverse Multiple Inheritance	1	0.400	0.975	0.962	0.952
Subclass Property Instantiation	0	0.115	0.0857	0.0133	0

$$S(s_1, s_2) = \frac{LCS(s_1, s_2)}{len(s_1) + len(s_2)} \tag{3}$$

Where s_1 and s_2 are the strings for which the similarity score is to be calculated. $LCS(s_1, s_2)$ is the function to calculate the longest common subsequence for two strings s_1 and s_2.

To identify the most appropriate RDF entity from the knowledge graph, a meticulous process of type iteration is undertaken. This is essential as there exists three distinct types of entities, namely Subject, Predicate, and Object. Each of these types of entities is meticulously stored in three segregated lists. The iteration process involves a thorough assessment of these lists while simultaneously computing the similarity score. This computation can be mathematically represented as follows. Let, R is the set of RDF entities belonging to a particular type. E be the entity of the same type, extracted from the natural language query. S be the similarity score calculated for each entity R_i in R with E. The candidate entity C for the query can be determined by selecting the entity with the highest similarity score amongst the set R.

$$C = \arg\max_{R_i \in R} S(R_i, E) \tag{4}$$

Where, $\arg\max_{R_i \in R}$ returns the entity $R_i \in R$ that maximizes the function, $S(R_i, E)$ which is the similarity score between the entity R_i from the knowledge graph and the extracted entity E from the query. This similarity score is calculated by using Eq. 3.

When there exist several entities within the knowledge graph that share the identical entity type as E and also possess the highest degree of similarity, designated as the maximum score (max), all such entities possessing equivalent maximum scores are taken into account during query generation. Consequently, numerous queries that relate to the same user query are produced. This process is reiterated until all the extracted entities have been linked with the data obtained from the underlying Knowledge Graph Database.

3.3.4 SPARQL triplet generation

Upon the extraction of entities from the sentence, mapping them to their respective components of the SPARQL query triplet can be accomplished. In the event that any of these components are absent, the utilization of placeholders (such as "?subject" or "?object"), within the SPARQL query is permissible, thereby enabling more versatile querying patterns. To illustrate, consider the query "Who is the CEO of Microsoft". The entities extracted from this query are "CEO" and "Microsoft", where "CEO" is categorized as PROPERTY and "Microsoft" as SUBJECT. Subsequently, both extracted entities are mapped to their corresponding RDF entities within the knowledge graph, thereby generating the triplet {*<ns:microsoft <np:chief_executive_officer> ?object*}, where "?object" is the missing entity. By merging the extracted entities with appropriate SPARQL syntax (e.g., using angle brackets to indicate RDF resources), a valid SPARQL triplet is generated to showcase the desired semantic relationship between the entities. During this stage, the question type that was extracted is utilized to construct the initial part of the query, which includes (SELECT *, SELECT COUNT(?a) as ?count, ASK). Moreover, the extracted entities are employed to produce the SPARQL query triplets (subject, object, predicate), which represent semantic relationships between the entities. It is plausible that one or more of these triplets may be absent, in which case the missing attribute is then returned within the SELECT or the COUNT query. While the formed triplet is provided inside the WHERE block.

Table 3 Dataset statistics for translation of NLQ to SPARQL for the Stock Market and the COVID-19 hospitalization custom datasets

Parameter	X-KG Stock	X-KG (COVID-19)
Number of records in dataset	136	30
The average number of words in a query	8.07	10.4
Number of SELECT type queries	99	24
Number of COUNT type queries	12	0
Number of BOOLEAN type queries	25	6

Table 4 Results for the translation of NLQ to SPARQL on the custom Dataset for the Stock Market

Type	Correctly generated	Total queries	Accuracy
SELECT	91	99	91.91%
COUNT	12	12	100%
BOOLEAN	22	25	88%

Table 5 Results for the translation of NLQ to SPARQL on the custom Dataset for COVID-19 hospitalization data

Type	Correctly generated	Total queries	Accuracy
SELECT	22	24	91.91%
COUNT	–	–	–
BOOLEAN	6	6	88%

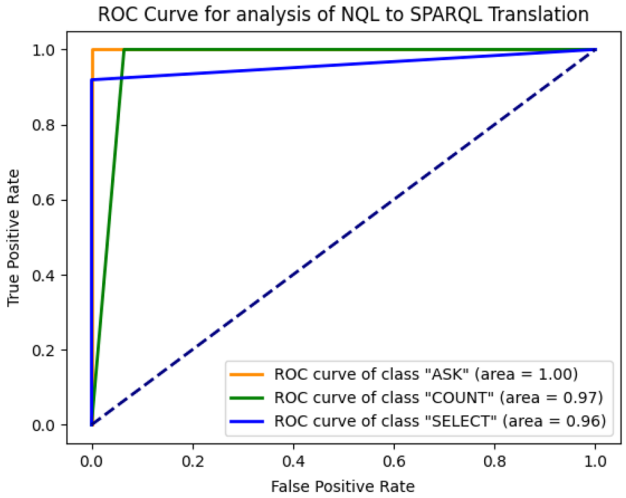

Fig. 6 ROC curve for NLQ to SPARQL model

4 Results and analysis

In this section, we present an analysis of the development of knowledge graphs and the transformation of natural language queries to SPARQL.

4.1 Analysis of X-KG

In order to evaluate the effectiveness of the knowledge graph generation model, a thorough analysis has been carried out on a set of structural quality metrics that are relevant to the generated X-KG knowledge graphs. The evaluation of the knowledge graph's structural quality is defined in the metrics outlined by [20], namely,

1. Instantiated Class Ratio
2. Instantiated Property Ratio
3. Class Instantiation
4. Inverse Multiple Inheritance
5. Subclass Property Instantiation

Instantiated Class Ratio[8] refers to the ratio of classes with instances among classes defined in the ontology. It is an indicator of how well the class of ontology is being used.

Instantiated Property Ratio[8] refers to the ratio of properties actually used in RDF triplet among the properties defined in the ontology. It is an indicator of how well the properties of the ontology are actually being used.

Class Instantiation[8] is a metric that assesses how much in detail classes are defined in the ontology and how much they are instantiated. For each class included in the knowledge graph, the class instantiation is calculated and summed to be used as an indicator representing the knowledge graph.

Inverse Multiple Inheritance[8] evaluates the simplicity of the knowledge graph. If multiple inheritance occurs frequently, in which a single class has numerous superclasses, it might make it challenging to use the knowledge graph because of the complexity of the class relationship. Inverse multiple inheritance was devised to measure how little multiple inheritance appears. The average number of superclasses per class is computed to obtain the average multiple inheritance and take the reciprocal of it. Therefore, the higher the Inverse Multiple Inheritance, the simpler the knowledge graph is.

Subclass Property Instantiation[8] quantifies how much the properties are used in the RDF triples when the properties of the subclass that are not in the superclass are defined in the ontology.

4.2 Analysis of NLQ to SPARQL translation

4.2.1 Dataset statistics

To evaluate the performance of the model, a specialized dataset was prepared.[9] This dataset consists of a Natural

[8] definitions taken from [20]

[9] Datasets will be published subsequent to manuscript acceptance.

Language Query (NLQ) and its corresponding SPARQL query, pertaining to both the stock market knowledge graph (X-KG) and the COVID-19 hospitalization knowledge graph (X-KG). The custom dataset statistics are described in Table 3.

4.2.2 Results of NLQ to SPARQL translation

Tables 4 and 5 serve to evaluate our model in addressing three distinct and elementary query types: *SELECT*, *COUNT*, and *BOOLEAN*. The results obtained demonstrate a significant level of accuracy across these categorizations. These discoveries, in aggregate, serve to highlight the robustness of our model's aptitude in comprehending and converting NLQ to SPARQL. From the results above, we have noted that the NLQ to SPARQL conversion system sometimes fails to extract all entities from an input query, resulting in incorrect SPARQL query formation, which explains the accuracy in Tables 4 and 5. Also, in certain cases, the question identification misclassifies a SELECT-type query to a COUNT-type query. For example, consider the query "How many employees work at Microsoft", Even though this sounds like a COUNT-type query it is not. The correct query of SELECT-type, where "number of employees" is a property of the entity "Microsoft".

From Fig. 6, we can infer that the AUC score for all the types of queries is >= 0.95, indicating a strong ability of our model to discriminate between the different query types.

5 Conclusion

In this research, a comprehensive workflow was presented for the creation of eXplicable Knowledge Graphs (X-KG) designed for explainable AI, alongside a system that facilitates natural language input translation to SPARQL queries, thereby enabling simplified search and retrieval. Throughout this research, we conducted experiments and evaluated the performance and effectiveness of the system. To evaluate the effectiveness of X-KG generation, we employed a few structural quality metrics and compared the generated X-KGs with other popular knowledge graphs. The results, as shown in Table 2, indicate that the X-KGs performed remarkably well across most metrics, albeit falling short in some areas such as Subclass Property Instantiation, which was mainly due to the utilization of lesser volumes of data. Furthermore, the NLQ to SPARQL translation was intended to facilitate the retrieval of data by non-technical users via simple one-level queries. We evaluated its performance using a customized dataset curated for the X-KGs generated and found that it exhibits reasonable accuracy in generating single-level queries.

The future prospects of this study involve the integration of Natural Language translation for more complex SPARQL queries, resulting in an improved Q &A system for the X-KG.

Acknowledgements All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

The authors have no financial or proprietary interests in any material discussed in this article.

Author Contributions All authors have contributed equally in this work.

Funding The authors acknowledge that no external funding was received for this research.

Availability of supporting data On acceptance of this manuscript, datasets used, and source code will be made available.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethical approval Not applicable

References

1. Gupte A, Sapre S, Sonawane S (2021) Knowledge graph generation from text using neural machine translation techniques. In: 2021 International Conference on Communication information and Computing Technology (ICCICT), pp 1–8, https://doi.org/10.1109/ICCICT50803.2021.9510164
2. Wang Q, Li M, Wang X, et al (2021) COVID-19 literature knowledge graph construction and drug repurposing report generation. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations. Association for Computational Linguistics, Online, pp 66–77, https://doi.org/10.18653/v1/2021.naacl-demos.8
3. Kejriwal M (2019) Domain-specific knowledge graph construction. Springer. https://doi.org/10.1007/978-3-030-12375-8
4. Li L, Wang P, Yan J et al (2020) Real-world data medical knowledge graph: construction and applications. Artificial Intelligence in Medicine 103:101817 https://doi.org/10.1016/j.artmed.2020.101817, www.sciencedirect.com/science/article/pii/S0933365719309546
5. Ye H, Zhang N, Chen H, et al (2022) Generative knowledge graph construction: A review. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, pp 1–17, https://doi.org/10.18653/v1/2022.emnlp-main.1
6. Dubey M, Dasgupta S, Sharma A, et al (2016) Asknow: A framework for natural language query formalization in sparql. In: The Semantic Web. Latest Advances and New Domains: 13th

International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29–June 2, 2016, Proceedings 13, Springer, pp 300–316
7. Yin X, Gromann D, Rudolph S (2021) Neural machine translating from natural language to sparql. Futur Gener Comput Syst 117:510–519
8. Bhajikhaye SS (2021) Translating natural language queries to sparql. Master's thesis, San Jose State University, https://doi.org/10.31979/etd.54xm-q833, https://scholarworks.sjsu.edu/etd_projects/989
9. Ferré S (2017) Sparklis: An expressive query builder for sparql endpoints with guidance in natural language. Semantic Web 8(3):405–418. https://doi.org/10.3233/SW-150208
10. Ochieng P (2020) Parot: Translating natural language to sparql. Expert Systems with Applications: X 5:100024 https://doi.org/10.1016/j.eswax.2020.100024, www.sciencedirect.com/science/article/pii/S2590188520300032
11. Chen YH, Lu EJL, Ou TA (2021) Intelligent sparql query generation for natural language processing systems. IEEE Access 9:158638–158650 https://doi.org/10.1109/ACCESS.2021.3130667, https://ieeexplore.ieee.org/document/9627128
12. Delahoy MJ, Ujamaa D, Whitaker M et al (2021) Hospitalizations associated with covid-19 among children and adolescents-covid-net, 14 states, march 1, 2020-august 14, 2021. Morb Mortal Wkly Rep 70(36):1255
13. Breiman L (2001) Random forests. Mach Learn 45(1):5–32. https://doi.org/10.1023/A:1010933404324
14. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297. https://doi.org/10.1007/BF00994018
15. Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Association for Computing Machinery, New York, NY, USA, KDD '16, p 785-794, https://doi.org/10.1145/2939672.2939785
16. Devlin J, Chang MW, Lee K, et al (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Association for Computational Linguistics, Minneapolis, Minnesota, pp 4171–4186, https://doi.org/10.18653/v1/N19-1423
17. Ramos JE (2003) Using tf-idf to determine word relevance in document queries. https://api.semanticscholar.org/CorpusID:14638345
18. Sanh V, Debut L, Chaumond J, et al (2019) Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. CoRR abs/1910.01108. arXiv:1910.01108
19. Lafferty JD, McCallum A, Pereira FCN (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '01, p 282–289
20. Seo S, Cheon H, Kim H, et al (2022) Structural quality metrics to evaluate knowledge graphs. version 2, arXiv:2211.10011