Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2020) 28: 1474 – 1490

doi:10.3906/elk-1908-102

Implicit relation-based question answering to answer simple questions over DBpedia

Maryam JAMEHSHOURANI, Afsaneh FATEMI*, MohammadAli NEMATBAKHSH
Department of Software Engineering, Faculty of Computer Engineering, University of Isfahan, Isfahan, Iran

Received: 20.08.2019 • **Accepted/Published Online:** 17.01.2020 • **Final Version:** 08.05.2020

Abstract: RDF-based question answering systems give users the capability of natural language querying over RDF data. In order to respond to natural language questions, it is necessary that the main concept of the question be interpreted correctly, and then it is mapped to RDF data. A natural language question includes entities, classes, and implicit and explicit relationships. In this article, by focusing on identification and mapping of implicit relationships in the question (in addition to the explicit relationships), the mapping step has been improved. In the proposed solution (IRQA), entities and implicit/explicit relationships are identified by means of the provided rules and then will be presented as a graph. In the next phase, according to the determined priority of graph nodes, unknown nodes and their adjacent relationships are mapped to their peers in the RDF dataset. The proposed solution is evaluated on the QALD-3 test set. The results prove that the proposed work has improved performance in mapping implicit relationships and reveals higher precision and F-measure values.

Key words: Semantic web, question answering, linked data, natural language, SPARQL, RDF

1. Introduction

Linked data are published in the RDF format [1]. The RDF structure includes a triplet in the form of (subject, predicate, object) and a set of these triplets forms a large graph of linked data. In order to extract information from this set of data, the SPARQL query language was introduced. Users would need to be familiar with the SPARQL query language and the relevant linked data scheme in order to obtain the information from the linked data. As a result, RDF-based question answering systems were introduced, which masked these complexities from the users and enabled querying without the need for familiarity with the SPARQL language and RDF data scheme [2, 3].

Question answering systems receive the user's question in natural language form and return the exact corresponding answer [3]. Identification of elements and relationships in natural language questions and mapping them to the entities, classes, and predicates in the RDF dataset and disambiguation are the key steps in these systems [4–6]. The purpose of this paper is to improve and enhance the responsiveness of RDF-based question answering systems by enhancing identification of question elements, mapping, and disambiguation. In the identification stage, it is necessary that entities, classes, and existing relationships in the natural language queries are identified. In the mapping stage, identified concepts are mapped to their corresponding elements in the RDF dataset. In this process the ambiguity is very important. Ambiguity refers to the possibility of mapping a natural language expression to several elements (entity, class, or predicate) in the RDF dataset [7].

*Correspondence: a_fatemi@eng.ui.ac.ir

Most of the prior solutions have used dictionaries to map relationships [5, 6]. These dictionaries, such as patty [8] and reverb [9], map natural language patterns to their peers in the RDF dataset. Relationship dictionaries are useful in general, but they are not comprehensive due to the complexity of natural language and the diversity of its relationships. Hence, approaches based on relationship dictionaries are faced with limitations. Meanwhile, implicit relationships are neglected in these methods. "Implicit relationships" refers to the relationships in which the concept of them is not explicitly stated in the sentence and the concept could be comprehended based on other information in the sentence. On the other hand, explicit relationships include a clear and direct statement of the concept through a relationship. Due to the differences between implicit and explicit relationships, new methods are required to map implicit relationships, which were not considered in the most of prior research.

In the proposed solution (IRQA), both identification and mapping stages are considered. In the identification stage, a natural language question is structured based on principles derived from dependencies of the question elements and then will be presented in the form of a graph. The created graph depicts entities and classes of the query as nodes and also represents relationships by edges. Since the structure of linked data is based on the triples, converting the input query into a structure similar to the linked data can simplify the mapping operation. Therefore, the input query is converted to a graph that consists of a set of triplets.

In the mapping phase, graph elements are mapped to the corresponding elements in DBpedia datasets. In this stage, each natural language query might include one or several subqueries. In IRQA, each subquery is taken as a separate query and mapping of relationships and disambiguation takes place only by considering its elements. Each subquery consists of an unknown node, the value of which is obtained after mapping of the subquery relationships. Based on the obtained values for the unknown node in this subquery, other subqueries are processed. The order of responding to subqueries is also based on the assigned priority to the unknown nodes of the graph.

In IRQA, the relationships of each subquery are mapped based on the known information of that subquery. Hence, the mapping of relationships can be done without the need for a dictionary of relationships. Therefore, the limitations of using a relationship dictionary are eliminated. In addition, in IRQA, implicit relationships can be mapped to their corresponding elements.

2. Background

This section provides an overview of the frameworks presented for question answering systems. DEANNA [10] uses a dictionary to map entities and relationships to their corresponding elements in linked data. Then a disambiguation graph is created and each candidate graph that has the highest similarity score is determined. In this system, the use of the relationship dictionary leads to some restrictions in mapping the relationships, since the relationship dictionaries are not complete and cannot cover all relationships in the natural language. Because of these limitations, the mapping of implicit relationships also has some drawbacks.

CASIA [6] carries out the mapping phase in three stages: class mapping, mapping of entities, and the mapping of relationships. It uses the dictionaries in this process. Therefore, it has restrictions arising from the use of the relationship dictionaries.

gAnswer [5] converts the input query into a graph called the query graph and considers the main problem as a subgraph matching problem. For each node and relationship in the graph, by using the existing dictionaries, candidate elements are determined. Then all possible graphs with different candidates are created and scored. Each of these graphs, which have more scores, is considered as the corresponding graph of the query graph.

There is a limitation of the relationship dictionary in this system and also the mapping of implicit relationships has not been considered in it.

Zhu et al. [11], in their proposed framework, convert the input question into a graph. Using the Wikipedia miner tool [12], concepts from DBpedia that correspond to specific question elements are extracted, and based on the structure of the query graph, a subgraph is considered around these elements. Subsequently, all paths in these subgraphs are scored, the paths between these subgraphs that structurally match the query graph are identified, and the path with the highest similarity score is presented as the final mapping. Unlike previous systems, this system does not use the relationship dictionary. However, it cannot efficiently identify the implicit relationships and map them to corresponding elements.

RTV [4] mapped the natural language question to the Markov hidden model. The Viterbi solution [13] is a complete disambiguation for the query elements. This system, like the proposed system of Zhu et al. [11], has problems in identifying and mapping implicit relationships.

In [14], some methods in deep learning have been analyzed for use in question answering systems. First, AIML Chabot was studied, which can solve fact-based question answering problems. Then the LSTM model [15] was investigated. Then memory networks were used, but they are not effective for complex tasks, so alternative dynamic memory networks [16] were proposed.

In one of the works using deep learning,[1] the baseline models, GRU and Sequence-to-Sequence [17], were analyzed. Next, dynamic memory networks were studied. Finally, end-to-end memory networks [18] were used as the model.

The study in [19] used the CNN semantic model (CNNSM) to create two mapping models. The CNNSM calculates the similarity between the entities in the query and knowledge base, and also the similarity between relations in the query and knowledge base. The authors of [20] presented a multicolumn model of CNN (MCCNN) that considers different aspects such as answer path, answer context, and answer type in the scoring process of the matching between question and answer.

In [21] a two-way RNN model was used to understand the concept of the question. These two sections have been used to understand the question and determine the weight of the matching of questions and answers in turn.

In [22] SPARQL patterns were used, and based on the selected dataset, element tagging was performed. Then, by using the Stanford parser, tokens were identified and finally, the proper pattern was selected. In this approach, aggregation questions are also respondable.

In [23] the first step is the identification of candidates for entities in the question and then an action is taken to resolve the ambiguity. In this stage, two approaches have been used: GTSP and Connection Density. The results of the evaluation of it show its superiority to approaches such as the DBPedia Spotlight.

Abujabal et al. in [24] provided a continuous learning approach. In the offline stage, the initial learning is done based on a small training set and then answers to the user's questions. If at some point it is not possible to answer the questions, it will continue the learning process. At all stages, including the response phase as well as the learning phase, the user's feedback is used to improve the system performance.

In Sparklis [25], a query-based faceted search (QFS) is presented that is based on different SPARQL endpoints and offers suggestions for each focus in the user query. Selecting each of the suggested elements will complete a portion of the query until it is fully generated, based on which the desired answers are extracted.

[1]Stanford University (2019). Question answering using deep learning [online]. Website https://www.cs224d.stanford.edu/reports/StrohMathur.pdf [accessed 20 June 2019].

One of the most important features of Sparklis is portability on different endpoints. It also pursues the goals of scalability. The former version of Sparklis is Scalewelis [26], which, like Sparklis, is a QFS system but it has less expressivity compard to Sparklis.

CANaLI [27] is a controlled natural language question answering system that offers suggestions at each step of the user's typing, from which to select and complete the query. CANaLI manages its operations on tokens based on a finite state machine and, based on user input, enters new states and offers suggestions to the user to complete the query.

3. The proposed method for implicit relation-based question answering

The proposed solution of this paper includes three main phases: creating a graph, mapping the graph elements to the DBpedia elements, and answer retrieval. A general overview of conducted steps of the proposed framework is depicted in Figure 1. In the following sections, each phase of the IRQA is described in detail.

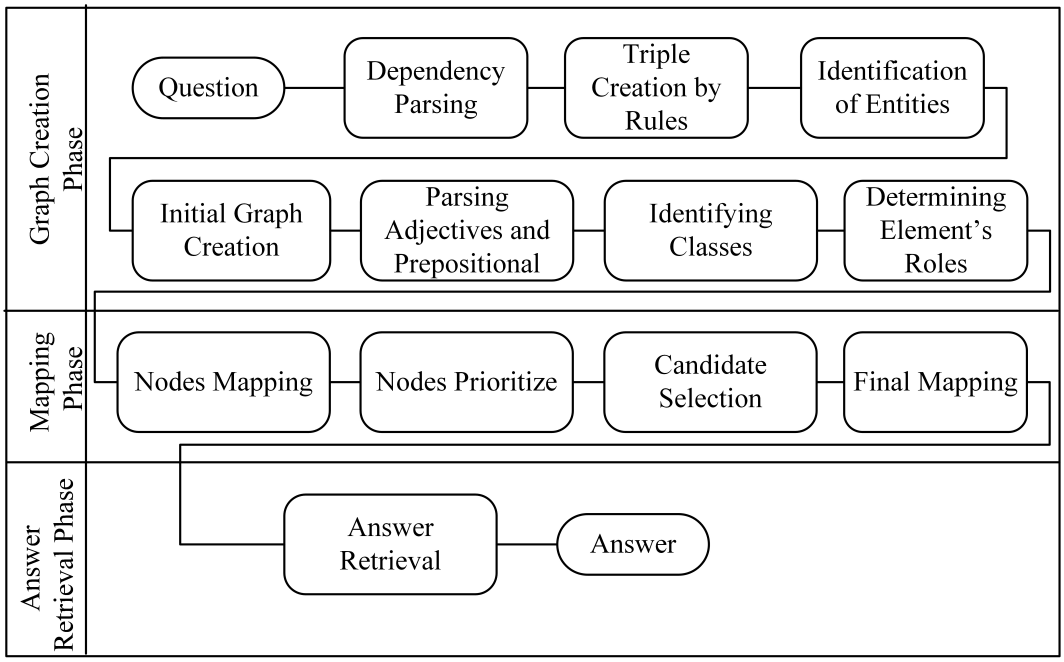

Figure 1. Overall framework.

3.1. Graph creation phase

A natural language query is converted to a graph (G_q) in order to eliminate the structural distance of natural language query and RDF data. This graph is a query graph, introduced in Definition 1.

Definition 1 *G_q consists of a number of nodes and edges between them. Each node corresponds to an entity or class in the natural language question and each edge corresponds to the relationship between those entities or classes.*

The triplet creation is based on a number of rules, which are dedicated to determining query triplets (entities and relationships) by using the grammatical dependencies of the elements of the sentence. The relationships

might show up explicitly (Definition 2) or implicitly (Definition 3).

Definition 2 *"Explicit relationship" refers to a relationship that transfers the main intended concept. The concept of this relationship can be found without using the other information in the sentence.*

Definition 3 *"Implicit relationship" refers to a relationship in which the main concept is understood by means of existing information. These relationships might have corresponding words in the sentence or, alternatively, they might be hidden in the sentence. Even if these relationships have corresponding words in the sentence, their concept is only comprehensible by means of existing information.*

Each relationship and its connected nodes form a triplet, which corresponds to a triplet in the RDF dataset. For example, the query in Figure 2, "What is the founding year of Canadian universities?", includes (What, founding year, X), (X, type, University), and (X, null, Canada) triplets. The relationship between "Canadian" and X is indicated implicitly. The other existing relationship in this query is the explicit relationship of "founding year". The graph creation process includes a number of steps, which are introduced in the following.

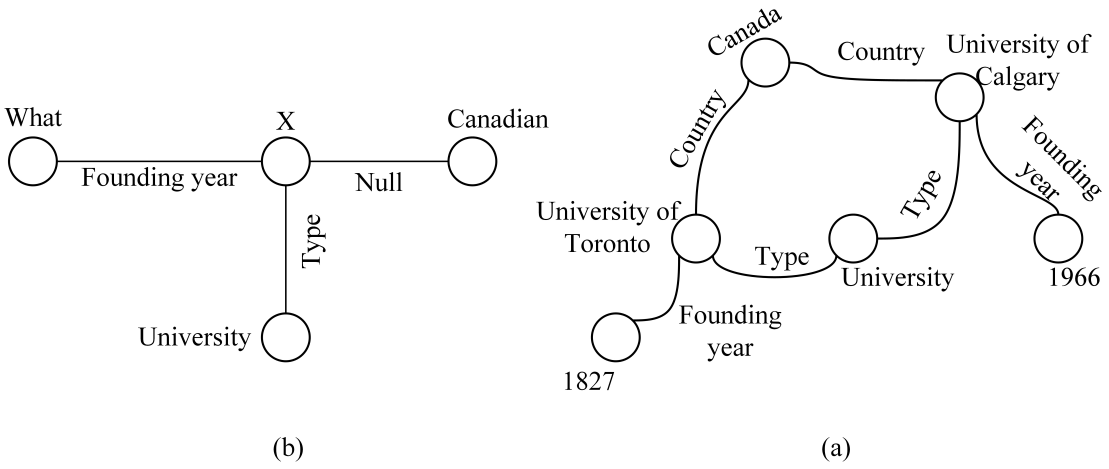

Figure 2. An example of natural language query: (a) subgraph in DBpedia, (b) question graph.

3.1.1. Identification of entities

Some of the expressions in the sentence correspond to entities in the dataset. These expressions might be composed of a number of words. However, they are considered as a single entity. Since the parser will parse sentence elements without taking this into account, proper measures must be taken to makes changes in the graph structure to prevent the breakdown of these expressions. Therefore, in order to prevent them from being broken down in the parsing process, these expressions are identified prior to grammatical analysis of the query. DBpedia Spotlight [28] is used to perform this identification. This tool is capable of identifying natural language entities and their corresponding elements in the DBpedia dataset. The longest subexpression of the question (Q) that corresponds to an entity in DBpedia is taken as an entity without the possibility of being broken down.

3.1.2. Creation of initial structure of the graph

In order to create a graph, the query structure must be analyzed grammatically. The type of dependencies between the words in the sentence must be identified and, according to them, placement of the words in the

form of a graph is determined. The Stanford dependency parser [29] is used to accomplish this task. This parser can provide a set of dependencies among the input sentence elements. The dependencies are in the form of dep(arg1, arg2), where dep determines the type of dependency and arg1 and arg2 are elements of the sentence such that dependencies (dep) exist between them [30]. In this step, relationships and arguments are identified based on the determined dependencies among the sentence elements (note that relationships are represented by edges and arguments are represented by nodes of the graph). The rules in Table 1 are used to do so. These rules that can be developed are based on a number of dependencies such as nsubj, nsubjpass, dobj, amod, and compound [30].

Table 1. Rules for creating query graph.

Rules	First dependency	Second dependency	Triple
1	subj(x,y) nsubj(x,y)	dobj(y,z) iobj(y,z) nmod(y,z)	(x,y,z)
2	subj(x,y) nsubj(x,y)	—	(x, null, y)
3	dobj(x,y) iobj(x,y) nmod(x,y)	—	(x, null, y)
4	nmod(x,n) compound(x,n)	—	(x, y, z) –>(nx, y, z)
5	nummod(x,n)	—	(x, y, z) –>(nx, y, z)

Table 1 indicates that if each of the dependencies in the first column is extracted at the same time as any of the dependencies from the second column, the resulting triplet would be in the form of the third column. These rules are applied based on the order indicated in this table; namely, the first rule is applied first, then the second rule, and so on.

The relationship values in rules 2 and 3 of Table 1 are labeled null, which means that these relationships are implicit. Another implicit relationship type is identifiable by the situation in rule 4, which will be determined by further processes on the graph.

In general, the implicit relationships existing in natural language queries can be classified into two groups:

1. **Implicit relationships with expressions of light information (light expressions):** These relationships are indicated by words with low information. These words can be further explained in two groups.

   - Prepositions: For instance, in the query "list all lakes in Iran", the preposition "in" shows a relationship between "Iran" and "lake" and corresponds to "location" or "country" in the linked dataset. However, the concept is implicitly indicated in this example.

   - Auxiliary verbs: For example, the auxiliary verb "were" in "give me all politicians that were chemists" has taken the place of a concept, which is indicated implicitly in this sentence. This concept could correspond to the "FieldofStudy" relationship in the linked dataset.

2. **Implicit relationships indicated as adjectival and prepositional phrases:** These compounds might include implicit relationships. If any of the adjectival or prepositional elements correspond to an entity,

their relationship would be shown by the label "null". For instance, in the query "give me all Canadian universities", the compound "Canadian universities" has an implicit relationship that corresponds to "country" in the RDF dataset and relates "Canada" and "university" to each other.

Rules number 2 and 3 in Table 1 are capable of identifying implicit relationships of type 1. Relationships of type 2 are identified by rule number 4 and will be explained in the following.

After creating the initial query (Q) triplets, the triplets with common nodes are joined together and form the graph structure.

3.1.3. Parsing adjective and prepositional phrases

According to rule 4, which was depicted in Table 1, adjectival and prepositional compounds are assumed as labels of the same node. These compounds might include implicit relationships. In order to identify these types of relationships, we analyze their corresponding entities by means of DBpedia Spotlight. If the entire compound is identified as a single entity, there is no implicit relationship between its elements. If at least two elements of this compound are identified as separate entities, there exists an implicit relationship between them.

For instance, in the aforementioned example, "university" and "Canadian" are identified as separate entities. Hence, there is an implicit relationship between them. In this case, the adjectives and possessions that were identified as entities will be removed from the respective label and will be added again as a separate node adjacent to their initial location. The edge between them will be valued by "null" as well. For instance, in Figure 2, "Canadian" would be added as a separate node next to X with a type of "University" and the edge label between them is set to null.

3.1.4. Identifying classes

The created graph includes nodes that correspond to the entities or classes in DBpedia. Classes are word to category entities with the same type; for instance, the class "country" includes the set of entities of this type. In this step, the nodes of the query graph (G_q) that correspond to a class in the DBpedia set of classes are identified. In order to identify classes, a list of existing classes in DBpedia has been used and each node with a semantic similarity to any of the elements in this list is identified as a class. The identified node as a class is denoted by N_O. Class identification results in adding a new node (N_n) to the graph, which is placed adjacent to N_O, and an edge is drawn between them and its label is "type". Next, the label of N_O is removed and N_n is assigned with the identified class.

3.1.5. Determining roles of the graph elements

As mentioned earlier, a triplet in the query graph corresponds to a triplet in the RDF dataset. The triplets in the RDF data are determined as (subject, predicate, object). In the created query graph, the roles of relationship arguments are not determined separately. As a result, both of the situations where arguments are taken as subject or object are assumed in the following steps. In IRQA, the roles of arguments are determined only if the relationship includes the preposition "of", such as "child of", "successor of", or any case matching [predicate]+of where the predicate is optional. In this case, if the structure of a part of the input sentence that includes this relationship matches arg1+[predicate]+of+arg2, the two triplets will be identified as (arg1, [predicate]+of, arg2) and (arg2, [predicate]+of, arg1), where the triplet in which arg2 takes the subject role and arg1 takes the object role has a higher priority. If the response could not be extracted in this format, the other triplet will be used.

3.2. Mapping phase

The mapping phase is focused on how to map query graph (G_q) elements to the elements of the linked dataset and disambiguation originating from that. The main purpose of mapping is finding the section of DBpedia that is related to the query and then finding the answer. Since the query is converted to a graph that contains triplets and has a similar structure to that of the DBpedia graph, by using the known nodes of a subquery, the place of the subquery in DBpedia is estimated.

A step by step approach is utilized to accomplish this task and known nodes of the graph (Definition 4) are mapped to the corresponding elements in the DBpedia. Consequently, the original query is assumed as a set of subqueries where the response to each subquery corresponds to an unknown node (Definition 5) in the graph.

Definition 4 *A known node in a graph refers to a node whose value can be identified from the DBpedia dataset in the mapping stage.*

Definition 5 *"Unknown node" refers to a node whose corresponding value from the DBpedia dataset is not identified after the mapping stage. These nodes are generally classified as main unknown and middle unknown. The main unknown of the query corresponds to the final response to the query. This unknown node usually corresponds to the nodes of the graph whose labels are question words of the query. If the query does not contain any question words, the node that corresponds to the type of requested entity is taken as the main unknown. Each unknown node except for the main unknown is assumed as a middle unknown.*

Therefore, identification of implicit relationships requires special attention as the concepts of these relationships are not clearly stated in the sentence as opposed to explicit relationships. For instance, Figure 2 shows a node with label X, which is a middle unknown node, and the one with label "What" (a question word) is a main unknown node. A query has only one main unknown but might have several middle unknown nodes.

3.2.1. Mapping nodes

The known nodes of the graph are mapped to their corresponding elements in DBpedia in this step by using DBpedia Spotlight. DBpedia Spotlight returns a list of corresponding elements accompanied by a numerical value as the score for each of the identified entities. Here, the element with the highest score among candidate elements suggested by DBpedia Spotlight is always selected as the corresponding element. DBpedia Spotlight is based on an input parameter called confidence, which determines the credibility level of the outputs. According to the observations, in this paper, this parameter is given a value of 0.35 that results in identifying most of the entities.

At the end of this step, all of the known nodes of the query graph have identified their corresponding values in DBpedia and the nodes without any corresponding element are assumed as unknown nodes.

3.2.2. Prioritization of graph nodes

In the IRQA, known information of the graph is used to determine unknowns. In each step, the subquery relationships of a node with higher priority are mapped to their corresponding relationships in DBpedia and the unknown node of that subquery is assigned a value, which turns it into a known node. The new known node will help in identifying the value of other unknowns. Therefore, each node is assigned a priority to determine

the value of unknowns according to the node prioritization process. The assigned priority has a numerical value and the lower value indicates the higher priority.

Algorithm 1 is provided to prioritize nodes. In this algorithm, node U represents the main unknown of the query graph, Array Tag includes node labels of the graph, and array P_r contains the assigned priorities of each node. The main idea is that each node N_k of the query graph G_q, which was assigned a label in the node mapping stage and had a corresponding element in DBpedia, will be given the highest priority, namely priority 1 (Lines 1–5). After the priorities of known nodes are determined, a procedure starts and continues until the main unknown node is assigned a priority (Line 6). According to this algorithm, for each known node n with specified priority, all of the neighboring nodes (such as N_u) are assigned equal priorities with a value of one unit higher than N_k (Lines 6–16). Therefore, N_u has also been assumed as a prioritized node and can be used to assign priority to other nodes in the next steps. At the end of this process, array P_r includes determined priorities for nodes of query graph G_q.

Algorithm 1 Node priority.

Input: Query Graph G_q=(V,E), Array Tag[1..n], Node U.
Output: Array P_r[1..n]

```
1:  for N_k ∈ V do
2:    if Tag[N_k] ≠ NULL then
3:      P_r[N_k] = 1
4:    end if
5:  end for
6:  while P_r[U] == 0 do
7:    for n ∈ V do
8:      if P_r[n] ≠ 0 then
9:        for N_u ∈ V do
10:         if (n, N_u) ∈ E&P_r[N_u] == 0 then
11:           P_r[N_u] = P_r[n] + 1
12:         end if
13:       end for
14:     end if
15:   end for
16: end while
```

3.2.3. Mapping relationships

As discussed earlier, after the known nodes are mapped to their corresponding elements, their relationships must be mapped to their peers as well and main and middle unknown nodes should be determined. In order to do so, two procedures are introduced: candidate selection and final mapping. These two procedures are called respectively for every unknown node with the highest priority and then assign its value and map the adjacent relationships.

3.2.3.1. Candidate selection

In this step, candidate elements are selected for each unknown node and its neighboring relationships.

At this stage, a procedure based on the known information of the graph is undertaken, and according to the information of each subquery, the candidates for unknown nodes and their relationships are determined. In

[11] a similar operation is done, but it is for the whole question, while in our work we use it for a single subquery, not for the whole question structure. Other differences between them will be specified in the following.

In order to accomplish this task, the process of determining candidate elements for each unknown node N with the highest priority compared to other unknown nodes must be performed. As shown in Figure 3a, for unknown node N, the neighboring known nodes are identified. These nodes are named n_1, n_2, … n_m. Since these are known nodes, their corresponding elements are specified in DBpedia. The corresponding elements are named C_{n1}, C_{n2}, … C_{nm}. Node N is neighboring all of the nodes n_1, n_2, … n_m and therefore its corresponding node in DBpedia, T, is adjacent to C_{n1}, C_{n2}, … C_{nm}. If all of the neighbors of each of nodes C_{n1}, C_{n2}, … C_{nm} is obtained, node T would be a member of all of these sets. Hence, it is located in the subset of the neighbors of nodes C_{n1}, C_{n2}, … C_{nm}.

Each of the nodes in this subset, such as T_i, is in a relationship with nodes C_{n1}, C_{n2}, … C_{nm} through relationships denoted by k_{i1}, k_{i2}, .., k_{im}. The nodes in this subset accompanied by relationships k_{i1}, k_{i2}, .., k_{im} are in the set I:

$$I = \{(T, k_{i1}, k_{i2}, .., k_{im})\}. \tag{1}$$

Figure 3b shows the process of selecting candidate elements for relationships. For each node T_i, its relationship to C_{n1}, which is denoted by k_{i1}, is a candidate element for relationship R_1 in the query graph. Moreover, relationship k_{i2} corresponds to R_2 and k_{im} corresponds to R_m.

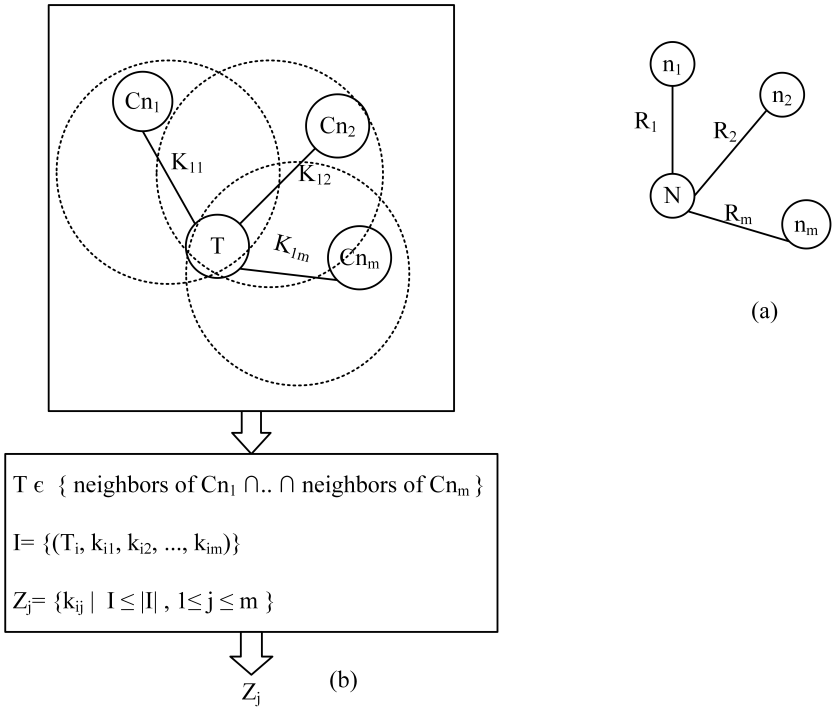

Figure 3. (a) Query graph; (b) candidate selection for relations.

Eventually, a set of candidate relationships will be obtained for each R_i of the query graph. This set is named Z_j and then provided as input to the final mapping step in order to determine final elements. Z_j is defined as in Equation 2 for each $1 < j < m$:

$$Z_j = \{k_{ij} |\ i < |I|\}. \tag{2}$$

In this set, element k_{ij} shows relationships from set I that are determined as candidate relationships. Each of the sets (Z_j) includes candidate elements for R_j in the query graph. |I| in Equation 2 stands for the number of members of set I.

3.2.3.2. Final mapping

In this step, the final corresponding elements are selected from all of the elements identified as candidates. This action is conducted for each relationship (such as R_j) of the query graph and set Z_j, which includes candidate elements for R_j. As a result, it is determined which members of Z_j can be correspondent to R_j. Using known information of the graph for the mapping phase eliminates the need for a dictionary of relationships. In addition, implicit relationships can be mapped by this approach.

The procedure of final mapping is shown in the flowchart in Figure 4. In the final mapping step, the semantic similarity of elements is utilized. Z_j and R_j in the query graph are entered as input in this step. R_j can have a null label or hold a value. If the label is null, it is not possible to calculate the semantic similarity between the label and candidate relationships. Therefore, the information of neighboring nodes will be used. If label R_j is not null, initially the label is analyzed from the complete similarity aspect to the members of candidates set. If none of the candidate elements have a similarity score of 1 with the relationship label, the relationship label will be expanded from a semantic aspect, which means that a number of its semantic synonyms are extracted and then stored in C_1 accompanied with the label itself. By doing so, all of the concepts that might be deducted from that relationship are considered, and a better understanding is achieved. Onelook Dictionary is used to extract synonyms of relationships.

The C_1 set is compared to candidate elements in Z_j and the semantic similarity of all of their members is calculated pairwise. In order to calculate semantic similarity, the UMBC tool [31] is used, which does not impose limitations with regards to the grammatical role of any two words. The words can be nouns or verbs and semantic similarity is calculated for them. The output is a value in the range of [0, 1].

Each element of candidates in the set, Z_j, which has a semantic similarity value equal to or higher than the threshold value, θ, with one of the elements of C_1 is added to the response set. If none of the members of Z_j has a similarity score with C_1 elements that satisfies the threshold level, the algorithm enters a stage where the label of the main relationship in the graph is ignored and is assumed null. In fact, we are dealing with an implicit relationship, which is not hidden in the query; however, its concept can be understood from the information in the query. For instance, in the sentence "give me a list of all countries connected by the Rhine", the relationship "connected" is identified in the sentence structure. The relationship identified in DBpedia to answer this question is "Country". We can observe that the relationship "connected" has no semantic similarity with the relationship assigned to it in DBpedia. Thus, if candidate elements cannot maintain the desired similarity threshold level with any of C_1 members, it is possible that the relationship contains an implicit concept while having a nonnull label.

Therefore, they must be considered as an implicit relationship as well. In these cases, it is not possible to calculate the similarity of candidate elements with the relationship label and its synonyms, like in the previous section. Neighboring nodes shall be used in these cases. Observations show that the type of nodes in the neighborhood of a relationship in DBpedia has the same value with its assigned relationship label in many cases.

In this step, the type of neighboring nodes to the relationship under analysis is extracted and stored as C_2. Afterward, the semantic similarity of Z_j and C_2 elements is calculated. Any of the candidate elements

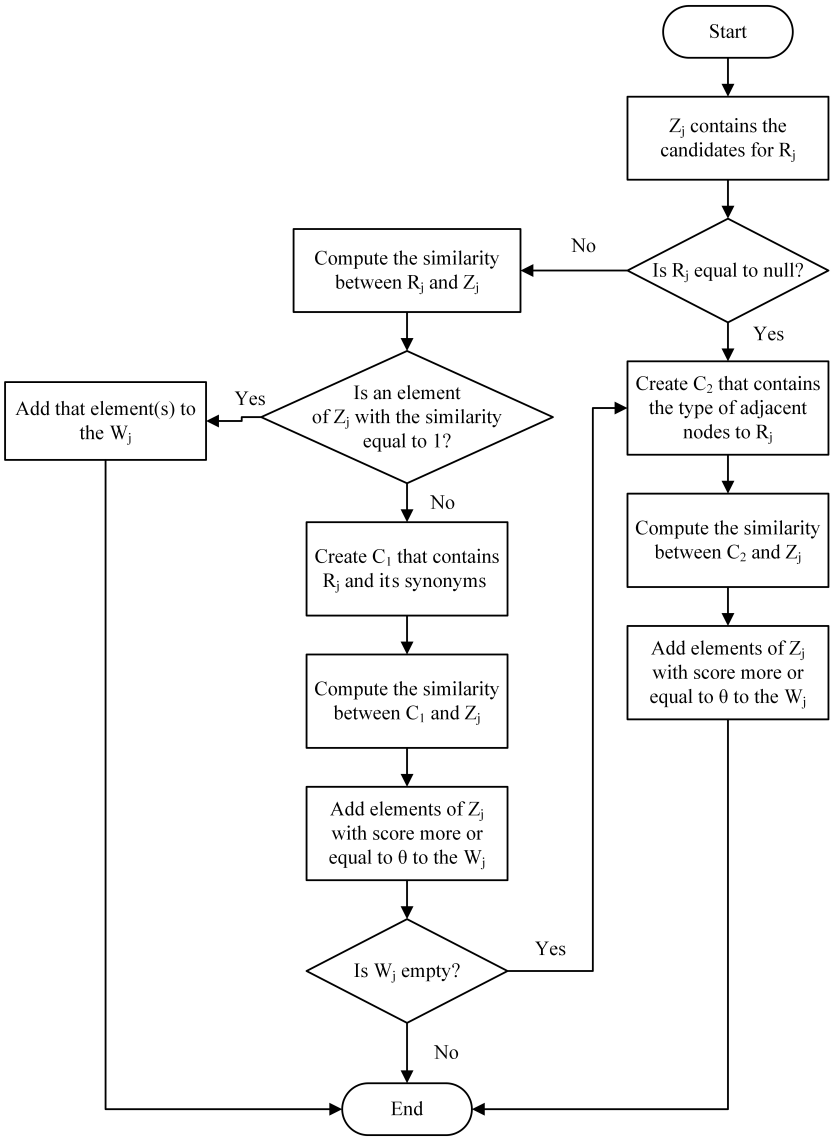

Figure 4. Calculating the semantic similarity for candidate relations.

that maintain a semantic similarity equal to or higher than the threshold with any of the members of C_2 would be added to the response set. Finally, the response set is returned as the output and includes the final relationships corresponding to the query graph. It should be noted that for each relationship, several corresponding relationships might be identified.

In this research, the threshold value of θ is assumed as 0.25. This value is determined empirically, and for each two given elements in the same semantic domain, the similarity value might turn out higher than 0.25 in many cases. In order to determine this threshold, QALD-3 [32] training queries were used.

In the end, for each relationship of R_j in the query graph, there will be a response set named W_j. Compatible elements with the graph structure should be extracted from the elements of this set. Therefore, the final mapping will be conducted for the unknown node under analysis, for which Algorithm 2 is provided. In this

algorithm, N is an unknown node and R_1, R_2, ..., R_m are the relationships connected to N. After determining the corresponding elements to the unknown node with the highest priority and also finding the corresponding relationships that are connected to the unknown node, this algorithm is iterated for the rest of the unknown nodes until the main unknown node of the graph is assigned a value. In this algorithm, each member of set I such that all of its relationships are in the final relationships is considered as the correspondence to N. If some of the members of set I do not satisfy this condition, but they have the conditions of Line 5 of algorithm, we consider that member as the correspondence to N.

Algorithm 2 Final selection.

1: **for** $(T, k_1, k_2, .., k_m) \in I$ **do**
2: **if** $k_1 \in w_1 \& k_2 \in w_2 \& ... \& k_m \in w_m$ **then**
3: $(T, k_1, k_2, .., k_m) \simeq (N, R_1, R_2, .., R_m)$
4: **end if**
5: **if** $k_1 \in w_1 \& k_2 \in w_2 \& ... \& k_m \notin w_m$ & $R_M == Null$ and k_m is the only candidate for R_m and $\sum_{i=1}^{m-1} Scoreofsimilarity(k_i, R_i)$ / m $\geq \theta$ **then**
6: $(T, k_1, k_2, .., k_m) \simeq (N, R_1, R_2, .., R_m)$
7: **end if**
8: **if** $k_1 \in w_1 \& k_2 \in w_2 \& ... \& k_m \notin w_m$ & $[R_M \neq Null$ or k_m is not the only candidate for R_m or $\sum_{i=1}^{m-1} Scoreofsimilarity(k_i, R_i)$ / m $< \theta]$ **then**
9: $(T, k_1, k_2, .., k_m) \nsim (N, R_1, R_2, .., R_m)$
10: **end if**
11: **end for**

Conditions in line 5 show that the relationship is implicit and only one candidate for it is identified, and also show that although $k_m \notin w_m$ the average of the similarity score of other relationships is still higher than or equal to θ and other relationships have a proper similarity score that can increase the reliability of this mapping.

3.3. Answer retrieval phase

As discussed before, there is a main unknown node in this graph, which corresponds to the final response of the input query. It is only required to return the mapped value to the main unknown node as the result without the need to create a SPARQL query.

4. Evaluation

4.1. Evaluation dataset and metrics

In order to evaluate the proposed solution, the QALD-3 dataset is utilized. This measure is a set of simple and complex questions. Since IRQA has been presented on simple questions over DBpedia, all complex and aggregation questions and also questions that have non-DBpedia knowledge bases have been removed from the QALD-3 test set and it has been renamed as QALD-3-Nonaggregation-DBpedia [11]. The new test set has been used to evaluate IRQA in comparison with other systems. QALD-3 contains 99 questions and only 61 questions of it are prepared for the IRQA process.

There are three metrics in QALD for evaluation of the question answering systems. These metrics are precision, recall, and F-measure [32]. We consider the average of these parameters in our evaluation on QALD.

4.2. Evaluation results

IRQA focuses on the identification of implicit relationships and maps them to the corresponding elements in DBpedia. In order to prove the successfulness of this approach, the percentage of queries that include implicit relationships and were answered correctly or partially by the systems in QALD-3 are presented in Figure 5. These results are obtained based on the published results of the QALD-3 challenge. The chart shows that IRQA has better results with regards to the mapping of implicit relationships.

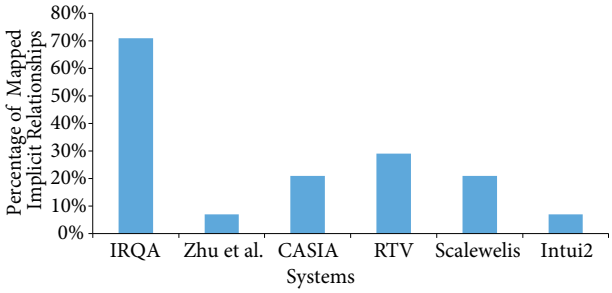

Figure 5. Percentage of mapped implicit relationships in QALD-3 systems.

The presented results of Table 2 prove the superiority of the IRQA in average precision and average F-measure. The IRQA can only respond to simple queries, and if the simple structure of the question is maintained, it can also respond to Yes/No questions. The reason for these better results in the IRQA is the mapping of implicit relationships to the corresponding elements. In general, the IRQA has improved performance and capability of correctly mapping elements by focusing on identification and mapping of implicit relationships without using a dictionary of relationships.

Table 2. Results on QALD-3-Nonaggregation-DBpedia.

	Total	Processed	Right	Partial	Avg. recall	Avg. precision	Avg. F-1
IRQA	61	56	31	17	0.67	0.70	0.66
Zhu et al.	61	53	30	13	0.67	0.61	0.61
CASIA	61	43	26	6	0.50	0.50	0.50
gAnswer Demo	61	38	21	7	0.41	0.45	0.42
RTV	61	41	24	3	0.43	0.41	0.41

In Table 3, although the evaluation is performed over all of the questions, the proposed method has still yielded a good performance compared to the systems that were initially designed to answer all types of queries.

Although some systems, such as CANaLI, have better results than IRQA and other systems, the reason is that this system and systems such as Sparklis and Scalewelis use controlled natural language that can improve results and reduce errors in understanding the queries. These systems restrict users to submitting questions based on system suggestions, while the ideal approach in question answering systems is to use natural language without any restrictions.

5. Conclusion

In this paper, an implicit relation-based question answering solution is proposed on DBpedia. In these queries, aggregation functions, counting functions, and filters are not used. The proposed method focuses on creating a

Table 3. Results on QALD-3-DBpedia.

	Total	Processed	Right	Partial	Avg. recall	Avg. precision	Avg. F-1
IRQA	99	59	34	17	0.45	0.46	0.44
Zhu et al.	99	60	31	17	0.46	0.40	0.40
CASIA	99	52	29	8	0.36	0.35	0.36
gAnswer	99	76	32	11	0.40	0.40	0.40
RTV	99	55	30	4	0.34	0.32	0.33
CANaLI	99	99	92	7	0.98	1.0	0.98
Scalewelis	99	70	32	1	0.33	0.33	0.33

graph of the query and then maps its elements to the DBpedia data elements. In these steps, not only are the entities, classes, and explicit relationships identified, but implicit relationships are also identified and mapped to their corresponding elements in the DBpedia dataset.

At the same time, due to only using the known information of the graph to determine other unknowns, there is no need to utilize extra information such as relationship dictionaries, which are used in many prior research works. Obtained results from evaluating the IRQA on a set of QALD-3 queries proves that the proposed method has an improved performance in mapping implicit relationships compared to other solutions, which helps improve precision and F-measure values.

One of the important aspects of question answering systems that remain open to further research is focusing on systems that process and answer nonsimple questions. This area has been considered in various systems but still has a big potential for improvement and betterment. In addition, this type of question has a higher capability of indicating implicit concepts and their identification plays an important role in the correct understanding of the question.

References

[1] Bizer C, Heath T, Berners-Lee T. Linked data: The story so far. In: Sheth A (editor). Semantic Services, Interoperability and Web Applications: Emerging Concepts. New York, NY, USA: IGI Global, 2011. pp. 205-227.

[2] Lopez V, Unger C, Cimiano P, Motta E. Evaluating question answering over linked data. Journal of Web Semantics 2013; 21: 3-13. doi: 10.1016/j.websem.2013.05.006

[3] Hirschman L, Gaizauskas R. Natural language question answering: the view from here. Natural Language Engineering 2001; 7 (4): 275-300. doi: 10.1017/S1351324901002807

[4] Giannone C, Bellomaria V, Basili R. A HMM-based approach to question answering against linked data. In: Proceedings of the Question Answering over Linked Data Lab (QALD-3) at CLEF; Valencia, Spain; 2013.

[5] Huang R, Zou L. Natural language question answering over RDF data. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data; New York, NY, USA; 2013. pp. 1289-1290.

[6] He S, Liu K, Zhao J. Casia @ QALD-3: A question answering system over linked data. In: Proceedings of the Question Answering over Linked Data lab (QALD-3) at CLEF; Valencia, Spain; 2013.

[7] Unger C, Freitas A, Cimiano P. An introduction to question answering over linked data. In: Reasoning Web International Summer School; Athens, Greece; 2014. pp. 100-140.

[8] Nakashole N, Weikum G, Suchanek F. PATTY: A taxonomy of relational patterns with semantic types. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning; Jeju Island, Korea; 2012. pp. 1135-1145.

[9] Manning C, Raghavan P, Schütze H. Introduction to information retrieval. Natural Language Engineering 2010; 16 (1): 100-103. doi: 10.1017/S1351324909005129

[10] Yahya M, Berberich K, Elbassuoni S, Ramanath M, Tresp V et al. Deep answers for naturally asked questions on the web of data. In: Proceedings of the 21st International Conference on World Wide Web; Lyon, France; 2012. pp. 445-449.

[11] Zhu C, Ren K, Liu X, Wang H, Tian Y et al. A graph traversal based approach to answer non-aggregation questions over DBpedia. In: Joint International Semantic Technology Conference; Yichang, China; 2015. pp. 219-234.

[12] Milne D, Witten IH. An open-source toolkit for mining Wikipedia. Artificial Intelligence 2013; 194: 222-239. doi: 10.1016/j.artint.2012.06.007

[13] Forney GD. The Viterbi algorithm. In: Proceedings of the IEEE 1973; 61 (3): 268-278. doi: 10.1109/PROC.1973.9030

[14] Sharma Y, Gupta S. Deep learning approaches for question answering system. Procedia Computer Science 2018; 132: 785-794. doi: 10.1016/j.procs.2018.05.090

[15] Sak H, Senior A, Rao K, Beaufays F. Fast and accurate recurrent neural network acoustic models for speech recognition. In: Proceedings of INTERSPEECH 2015; Dresden, Germany; 2015. pp. 1468-1472.

[16] Kumar A, Irsoy O, Ondruska P, Iyyer M, Bradbury J et al. Ask me anything: Dynamic memory networks for natural language processing. In: International Conference on Machine Learning; New York, NY, USA; 2016. pp. 1378-1387.

[17] Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems; Montreal, Canada; 2014. pp. 3104-3112.

[18] Sukhbaatar S, Weston J, Fergus R. End-to-end memory networks. In: Advances in Neural Information Processing Systems; Montreal, Canada; 2015. pp. 2440-2448.

[19] Yih WT, He X, Meek C. Semantic parsing for single-relation question answering. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics; Baltimore, MD, USA; 2014. pp. 643-648.

[20] Dong L, Wei F, Zhou M, Xu K. Question answering over freebase with multi-column convolutional neural networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics; Beijing, China; 2015. pp. 260-269.

[21] Hao Y, Zhang Y, Liu K, He S, Liu Z et al. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics; Vancouver, Canada; 2017. pp. 221-231.

[22] Atzori M, Mazzeo GM, Zaniolo C. QA3: A natural language approach to question answering over RDF data cubes. Semantic Web Journal 2019; 10 (3): 587-604. doi: 10.3233/sw-180328

[23] Dubey M, Banerjee D, Chaudhuri D, Lehmann J. Earl: Joint entity and relation linking for question answering over knowledge graphs. In: International Semantic Web Conference; Monterey, CA, USA; 2018. pp. 108-126.

[24] Abujabal A, Saha Roy R, Yahya M, Weikum G. Never-ending learning for open-domain question answering over knowledge bases. In: Proceedings of the 2018 World Wide Web Conference; Lyon, France; 2018. pp. 1053-1062.

[25] Ferré S. Expressive and scalable query-based faceted search over SPARQL endpoints. In: International Semantic Web Conference; Riva del Garda, Italy; 2014. pp. 438-453.

[26] Guyonvarch J, Ferre S, Ducassé M. Scalable query-based faceted search on top of SPARQL endpoints for guided and expressive semantic search. In: Proceedings of the Question Answering over Linked Data Lab (QALD-3) at CLEF; Valencia, Spain; 2013.

[27] Mazzeo GM, Zaniolo C. CANaLI: A System for Answering Controlled Natural Language Questions on RDF Knowledge Bases. Technical Report. Los Angeles, CA, USA: University of California, 2016.

[28] Daiber J, Jakob M, Hokamp C, Mendes PN. Improving efficiency and accuracy in multilingual entity extraction. In: Proceedings of the 9th International Conference on Semantic Systems; Graz, Austria; 2013. pp. 121-124.

[29] Nivre J, De Marneffe MC, Ginter F, Goldberg Y, Hajic J et al. Universal dependencies v1: A multilingual treebank collection. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation; Portorož, Slovenia; 2016. pp. 1659-1666.

[30] De Marneffe MC, Manning CD. Stanford Typed Dependencies Manual. Technical Report. Stanford, CA, USA: Stanford University, 2008.

[31] Han L, Kashyap AL, Finin T, Mayfield J, Weese J. UMBC_EBIQUITY-CORE: Semantic textual similarity systems. In: Second Joint Conference on Lexical and Computational Semantics; Atlanta, GA, USA; 2013. pp. 44-52.

[32] Cimiano P, Lopez V, Unger C, Cabrio E, Ngomo AC et al. Multilingual question answering over linked data (QALD-3): Lab overview. In: International Conference of the Cross-Language Evaluation Forum for European Languages; Valencia, Spain; 2013. pp. 321-332.