

A Methodology for Interpreting Natural Language Questions and Translating into SPARQL Query over DBpedia

Davide Varagnolo^{1,3}^a, Dora Melo^{2,3}^b and Irene Pimenta Rodrigues^{1,3}^c

¹*Department of Informatics, University of Évora, Évora, Portugal*

²*Polytechnic University of Coimbra, Coimbra Business School—ISCAC, Coimbra, Portugal*

³*NOVA Laboratory for Computer Science and Informatics, NOVA LINCS, Caparica, Portugal*

Keywords: Natural Language Processing, Question Answering, Knowledge Representation, Knowledge Discovery, Semantic Web, SPARQL Queries, Ontology.

Abstract: This paper presents a methodology that allows a natural language question to be interpreted using an ontology called Query Ontology. From this representation, using a set of mapping description rules, a SPARQL query is generated to query a target knowledge base. In the experiment presented, the Query Ontology and the set of mapping description rules are designed over DBpedia as target knowledge base. The methodology is tested using QALD-9, a dataset of natural language queries widely used to test question-answering systems on DBpedia.


1 INTRODUCTION


The ability of a system to correctly understand a natural language question can be useful in solving one of the most important tasks in the field of Question Answering (QA), discipline within the fields of natural language processing (NLP) which concerns the construction of systems capable of answering questions posed in a natural language (Hirschman and Gaizauskas, 2001). A branch of research focuses on the interpretation of natural language through semantic understanding using ontologies, able to create a different representation of the natural language question. A good representation of the natural language question is essential to proceed into next step of the QA process based on Knowledge Bases (KBs), which is the creation of a query in a target query language able to retrieve the correct information. In knowledge bases, the query language is SPARQL Protocol and RDF Query Language (SPARQL). Several approaches were presented in last years about the translation of natural language question into SPARQL queries (Zlatareva and Amin, 2021; Steinmetz et al., 2019; Silva et al., 2023). Literature shows that semantic parsing is the predominant paradigm in the Knowl-


edge Base Question Answering (KBQA). Can be observed that an often-used methodology is to break the problem of semantic parsing a complex question into more manageable sub-tasks, by a pipeline of three modules (Omar et al., 2023): understanding the question, entity and relation linking, and answer filtering. Several systems adapt their strategies based on this type of pipeline approach (Hu et al., 2021; Cornei and Trandabat, 2023).

These systems are often designed and tested on DBpedia¹, one of the largest open source knowledge bases available on the web based on information in Wikipedia².

In this paper, it is described in section 2 an ontology-based Question-Answering system capable of constructing a SPARQL query from a natural language question. In section 3, an ontology for DBpedia is presented. From this representation, a strategy based on mapping description rules to build a SPARQL query is presented in section 4. Section 5 shows the experience of this strategy with a natural language question dataset, QALD-9³.

^a <https://orcid.org/0009-0001-9212-9586>

^b <https://orcid.org/0000-0003-3744-2980>

^c <https://orcid.org/0000-0003-2370-3019>

¹<https://www.dbpedia.org/>

²<https://www.wikipedia.org/>

³<https://github.com/ag-sc/QALD/tree/master/9/data>

2 QUESTION ANSWERING SYSTEM

In this section, the question-answering system which create a SPARQL Query for a target KB from a natural language question is described. In Figure 1, the full architecture is presented. The architecture, presented in (Varagnolo et al., 2023), includes a pipeline with three modules: Partial Semantic Representation, Pragmatic interpretation, and SPARQL generator.

The Partial Semantic Representation is composed by two modules: a dependency parser, Stanza, is applied to the question and the resulting parser tree is transformed into a set of partial Discourse Representation Structures (DRSs). DRS refers to a set of Discourse Referents (DRs) and the relations on them. The DRS, like the algorithm used for its construction, is based on Discourse Representation Theory (Kamp and Reyle, 2013).

The Pragmatic interpretation module rewrites partial semantic representations into domain-specific meanings. It uses an ontology-based domain representation and multi-objective optimization to find the best interpretation within the Query Ontology, acting as an intermediary to the target KB.

Finally, the SPARQL Query Builder uses mapping description rules to generate a SPARQL query from the Semantic Query Representation by iteratively analyzing all triples.

This architecture follows the pipeline from related work: question understanding is done by Partial Semantic Representation and Pragmatic interpretation, creating a Query Ontology representation. Entity and relation linking, and filtering occur during mapping to the reference KB using mapping description rules.

3 QUERY ONTOLOGY

The Query Ontology is used as an intermediate ontology between the Partial DRS and the KB Target, so that a semantic, albeit partial, representation of the natural language question can be given. The classes chosen serve to represent the DRs of the application, which is why classes were chosen to contain the possible entities that can be found in the question.

In Figure 2 is presented the set of classes that make up the ontology. The main classes are:

- *Action*, which encloses the different actions, represented by a verb, within the Partial DRS;
- *Actor*, that represent subjects able to make actions, like people or organisations.
- *Object*, which represent inanimate objects.

- *Concept*, that represent immaterial concepts or feature linked to other entities, like Objects or Actors.
- *Place*, which represents places, and finally
- *Date*, that represents dates.
- *Event*, which represents events.

Each class is associated with a list of annotations which are used in the calculation phase of the best solution to assign the correct class to the discourse reference in the Partial DRS. In Table 2 a partial list of annotations is presented, which are used in the best solution calculation phase to assign the correct class to the discourse reference in the Partial DRS. The table shown a part of the whole set of annotations, the ones used in example questions. Among the other Classes, *Qualifier* and *Query* are used to represent the structure of the sentence, and *Cselect* and *Language-Model*, which represent namely the entities that can be part of the SELECT clause within the SPARQL Query and the entities that can be used to represent the elements within the sentence.

The Query Ontology's object properties model relationships between entities. Table 3 shows a partial set, only ones used in examples. To select the correct relationship, additionally to define the domain and range within the ontology, annotations which represent syntactic relationships between DRs in the Partial DRS are provided, being used similarly to class annotations.

Table 1: Query Ontology Data Properties used in example questions.

Data Prop.	Domain	Range
has_name	Thing	xsd:string
has_text	Thing	xsd:string
has_modifier	Thing	xsd:string

Furthermore, Table 1 shows data properties that preserve each entity's lemma and associated adjectives (has_text, has_modifier), and names for Proper Names identified (has_name).

Three different example questions from the dataset QALD-9 are presented to understand the process to achieve the solution.

3.1 Example Question 1

Question 1 : 'Which country was Bill Gates born in?'

The partial DRS in Figure 4 and the Stanza analysis in Figure 3 identify three DRs: *X1-country*, *X2-Bill Gates*, and *X3-born*. These DRs are linked through *obl_in* syntactic relation between *X3* and *X1*,

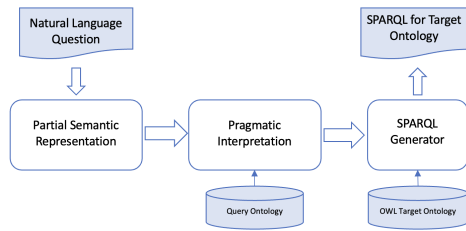


Figure 1: Natural Language Questions Representation as SPARQL Queries Architecture.

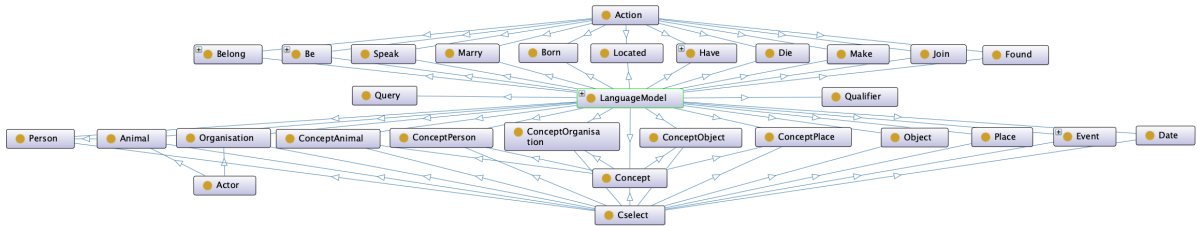


Figure 2: Classes of Query Ontology.

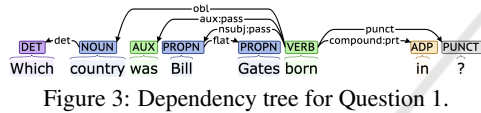


Figure 3: Dependency tree for Question 1.

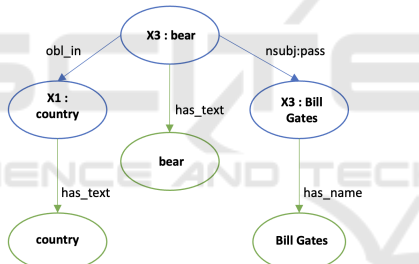


Figure 4: Partial DRS for Question 1.

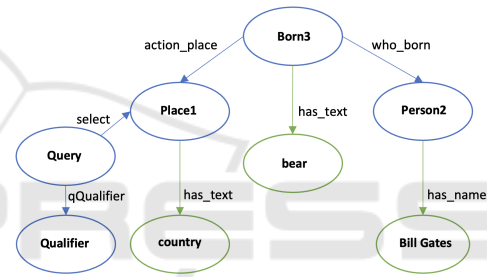


Figure 5: Query Ontology Solution for Question 1.

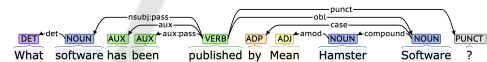


Figure 6: Dependency tree for Question 2.

and the *nsubj:pass* syntactic relation between *X3* and *X2*.

Since ‘born’ is defined, the object property *who_born* is preferred due to its *nsubj:pass* annotation, as ‘born’ is always passive. *X2* will be of type *Animal* or *Person*. The relation between *X3* and *X1* is *obl_in*. The *Born* class appears as the domain of *who_born*, *action_place*, and *action_date*. For the DR with lemma ‘country’, *Place* is preferred, as ‘country’ represents a place.

Thus, there are two possible interpretations within the natural language query, differing only in the interpretation of *X2* (as *Animal* or *Person*). Chosen solution is shown in Figure 5.

3.2 Example Question 2

Question 2 : ‘What software has been published by

Mean Hamster Software?’.

The partial DRS in Fig. 7 shows three DRs: *X1-software*, *X2-publish*, and *X3-Mean Hamster Software*. *X3-Mean Hamster Software* was identified as a unique DR using Named-entity Recognition (NER) to correct Stanza’s misinterpretations. The syntactic relationships are: *nsubj:pass* between *X2* and *X1*, and *obl_by* between *X2* and *X3*.

The action ‘publish’ can be interpreted as *Make*, both involving an *Actor* creating an *Object*. The *Make* class covers actions like directing a film or writing a book. The lemma ‘publish’ identifies *X2* as an individual of the *Make* class. The *Make* class has several object properties, narrowed down to *who_make* and *make_what* (Table 3). *X1* is assigned as *Object* and *X3* as *Actor* or a subclass thereof.

In Fig. 8 is presented the solution for the DRS partial of the natural language question.

Table 2: Query Ontology Classes used in example questions.

Class	subClassOf	Annotations
Person	Animal, LanguageModel, Cselect	[...]
ConceptPlace	Concept, LanguageModel, Cselect	area code, [...]
Place	LanguageModel, Cselect	country, [...]
Object	LanguageModel, Cselect	software, [...]
Organisation	Actor, LanguageModel, Cselect	[...]
Make	Action, LanguageModel	publish, [...]
Be	Action, LanguageModel	be
Born	Action, LanguageModel	bear

Table 3: Query Ontology Object Properties used in example questions.

Obj. Prop.	Domain	Range	Annotation
what_is	Be	Thing	subj
is_what	Be	Thing	obj
who_make	Make	Actor	subj, obl_by
make_what	Make	Object	obj, nsubj:pass
who_born	Born	Animal	nsubj:pass
action_place	Action	Place	obl_in, obl_on
conceptOfPlace	ConceptPlace	Place	of
qQualifier	Query	Qualifier	n obj, subj
select	Query	Cselect	n subj, obj, nsubj:pass

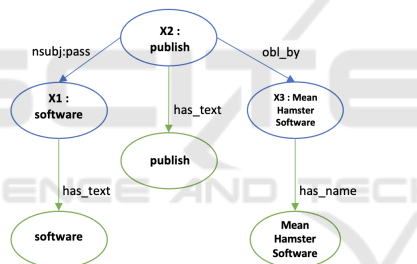


Figure 7: Partial DRS for Question 2.

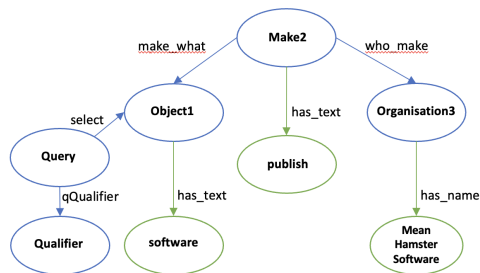


Figure 8: Query Ontology Solution for Question 2.

3.3 Example Question 3

Question 3 : 'What is the area code of Berlin?'

Four different DRs can be found in this case, which are shown in Fig. 10, result of the Stanza analysis shown in Figure 9: X1-what, X2-area code, X3-

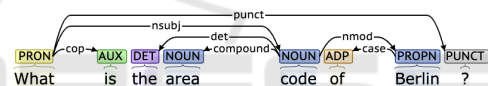


Figure 9: Dependency tree for Question 3.

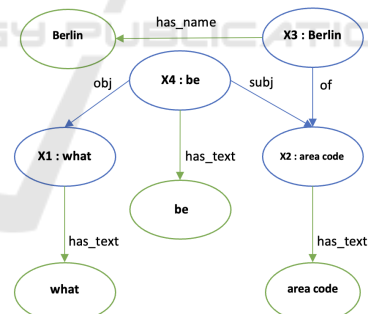


Figure 10: Partial DRS for Question 3.

Berlin, and X4-Be.

The verb 'to be' does not identify the relation but defines X1 and X2 as the same class. Since 'area code' implies a concept of place (Table 2), X3 is assigned as *ConceptPlace*. The relation X1-of-X2 is interpreted as the object property *conceptOfPlace*. The final solution is shown in Fig. 11.

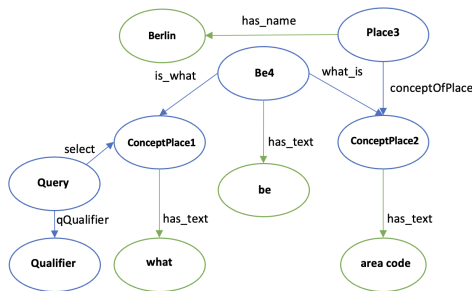


Figure 11: Query Ontology Solution for Question 3.

4 SPARQL QUERY BUILDER

The SPARQL Generator module receives as input the best solution selected by the previous module and through the application of mapping description rules generates the SPARQL query for the target KB. Mapping description rules are manually defined rules that allow each class and property of the solution to be translated into SPARQL. The SPARQL query is generated through the analysis of all triples in the solution, which will generate the query.

4.1 Entity Translation

The query ontology solution consists of a set of individuals related to each other via object properties and defined via data properties. Within the SPARQL query creation process, each individual is instantiated within the query as a variable having the name of the individual within the solution given by the query ontology, which is formed by the class it belongs to plus the index of the Discourse Reference, inherited from the partial DRS. Thus in the question 1 ‘Which country was Bill Gates born in?’, where in the solution were found the individuals *Place1* (*X1, country*), *Person2* (*X2, Bill Gates*) and *Born3* (*X3, Bear*), three different variables with these names (*Place1*, *Person2*, and *Born3*) will be instantiated within the generated query.

4.2 Resource Selection

The first set of properties that will be analysed are the data properties. They are used by the system to restrict the set of possible individuals of answers or to directly choose a particular resource. There are three main data properties:

- The first case is the *has_name* data property. The data property is created when, during the Stanza analysis, a name in the natural language query is identified as Proper Noun (PROPN). The mapping

description rule written to handle the *has_name* case takes into account two possible options: that the name is directly referred to a string or through all the individuals that contain within the string in relation with *rdfs:label* the string in the *has_name* relation. Below, the translation of *has_name* data property.

```
has_name(X1,nameX1) ->
{ FILTER (regex (?X1, nameX1)) . }
UNION
{ ?X1 rdfs:label ?nX1.
FILTER (regex (?nX1, nameX1)) . }
```

- The *has_text* data property holds the lemma for each individual in the solution. It accounts for cases where the lemma defines the object property, especially when there is no Action or when the Action cannot define the relationship (e.g., ‘to be’). This data property is not directly translated but is used in other Object Property translations. This mechanism prevents the system from always including *has_text* in the query body, excluding certain actions (like ‘Be’ or ‘Have’) and terms (e.g., wh-question words or ‘how’).

```
has_text(X1,textX1) ->
if (existType (hasLabel (textX1))) {
?X1 rdf:type ?tX1.
FILTER (regex (?tX1, textX1)) .
if (getClass (X1) equalTo Person) {
?X1 dbo:occupation ?tX1.
FILTER (regex (?tX1, textX1)) .
}
}
if (existProperty (hasLabel (textX1))) {
?X1 ?propX1 ?yX1.
?propX1 rdfs:label ?lPropX1.
FILTER (regex (?lPropX1, textX1)) .
}
```

- the last data property is *has_modifier*. Here are present the elements of the sentence marked by stanza as Adjectives (ADJ) associated with the respective DRs. Similar to the *has_text* data property, they are used to define the type of relationship in the object properties, the individual in the domain or the class of the responding entity and are not directly translated. The modifier is also analyzed outside the query body creation phase, to determine possible groupings or orderings of the answers, i.e. ‘ORDER BY DESC(?x) LIMIT 1’ expressed by keywords like ‘highest’.

```
has_modifier(X1,modTextX1) ->
for each m in has_modifier(X1,modTextX1)
if (existType (hasLabel (m.modTextX1))) {
?X1 rdf:type ?tX1m.
FILTER (regex (?tX1m, m.modTextX1)) .
}
if (existProperty (hasLabel (m.modTextX1))) {
{ ?X1 ?propX1m ?yX1m.
?propX1m rdfs:label ?lPropX1m.
FILTER (regex (?lPropX1m, m.modTextX1)) . }
UNION
{ ?yX1m ?propX1m ?X1.
?propX1m rdfs:label ?lPropX1m.
FILTER (regex (?lPropX1m, m.modTextX1)) . }
}
if (existIndividual (hasLabel (m.modTextX1))) {
```



```
{?X1 ?propX1m ?yX1m.
?yX1m rdfs:label ?lyX1m.
FILTER(regex(?lyX1m,m.modTextX1)).}
UNION
{?yX1m ?propX1m ?X1.
?yX1m rdfs:label ?lyX1m.
FILTER(regex(?lyX1m,m.modTextX1)).}
}
```

4.3 Entity Linking

Object properties relate different individuals. In the Query Ontology, actions (verbs) are marked as DRs during the Partial Semantic Representation phase, generating an individual of the Action subclass. In the solution, a ternary relation between the subject and the object complement (or place/date if the object complement is absent) is found through the action. In Example Question 1 can be seen that the subject *Person1* is related to *Place2* through the action *Born3*. In the target KB, i.e. DBpedia, unlike the Query Ontology, actions are often represented through Object Properties between objects. Therefore, two different approaches are used to correctly translate the pair of triples: the relation representing the subject of the action and the latter is translated as equality between the two individuals of the triple. In this way, the subject will be directly related to the second triple. the second triple, on the other hand, is translated using the lemma present in the Action (thus the verb related to it) to define the type of object property between the two entities. Below, the translation of two object properties in Example Question 2 representing the described case: *who_make* and *make_what*

```
who_make(X1,X2) -> replace(X2,X1)
has_text(X2,textX2)
has_mod(X2,textX2)
make_what(X1,X3) ->
{?X1 ?propM201 ?X3.
?propM201 rdfs:label ?nPropM201.
FILTER(regex(?nPropM201,textX1))}
UNION
{?X3 ?propO1M2 ?X1.
?propO1M2 rdfs:label ?nPropO1M2.
FILTER(regex(?nPropO1M2,textX1))}
```

Since the relation type is based on the lemma, it must include the possibility that the subject may be the domain or range of the relation in DBpedia, included by UNION. Similar to identifying the class when analyzing *has_text*, synonyms can define the object property in DBpedia. Currently, synonym mapping rules are defined manually.

There are two special cases: when the *Action* is 'to be' or 'to have'. For 'to have', the lemma text is not analyzed as it does not help identify the relation type. For 'to be', the variables defining three individuals in the Query Ontology are equated. The correct interpretation is determined by constraints on variables from other Query Ontology relations. Below are the mapping description rules for three object properties,

illustrated by Example Question 3.: *what_is(X1,X2)*, *is_what(X1,X3)*, and *conceptOfPlace(X2,X4)*.

```
what_is(X1,X2) ->replace(X2,X1)
has_text(X2,textX2)
has_mod(X2,textX2)
is_what(X1,X3) ->replace(X3,X1)
has_text(X2,textX3)
has_mod(X2,textX3)
conceptOfPlace(X2,X4) ->
has_text(X2,textX2)
if(not existType(hasLabel(textX2)) and not existProperty(
hasLabel(textX2))){
FILTER(X2 = X4)
}
```

4.4 Query Building Process

In the last section, the construction of a query will be shown step by step. Although the creation of the query will be described in steps to understand how it is done, in practice it is not necessary to use an order in the analysis of the solution triples.

4.4.1 SPARQL Query for Question 1

In the first step, the SELECT clause is constructed. The Query Ontology identifies *Place1* as the target via the object property *select*, with no modifier. Next, the data property *has_name* for *Person2* is translated in the query body by defining *Person2* as individuals labeled 'Bill Gates'. Next, the object properties *who_born* and *action_date* are found. *who_born* equates *?Person2* and *?Born3*, while *action_place* is defined by the class assigned to the property action, implying the translation *?birthProp* with rdfs:label 'birth place'. Since Place has 'country' as the value of *has_text*, it's assumed the country is specified. The translation adds a UNION to handle two triples in DBpedia: one for *?Person2*'s city of birth (*?birthProp*) and another linking the city to its country (*?country*) with rdfs:label 'country'.

```
who_born(X1,X2) -> replace(X2,X1)
action_place(X1,X3) ->
switch(getClass(X1))
case "Born":
?birthProp rdfs:label ?lbp.
FILTER(regex(?lbp,"birth place","i")).
{?X1 ?birthProp ?X3.
if(existType(hasLabel(textX3))){
?X3 rdf:type ?tX3.
?tX3 rdfs:label ?ltX3.
FILTER(regex(?ltX3,textX3,"i")).
}}
UNION
{?X1 ?birthProp ?cityX3.
?cityX3 ?cProp ?X3.
?cProp rdfs:label ?lcProp.
FILTER(regex(?lcProp,"country","i")).}
[...]
```

Below, the final result of the mapping through the mapping description rules of the Ontology query solution in a SPARQL query for DBpedia is presented.

```
SELECT DISTINCT ?Place1
WHERE {
?Born3 rdfs:label ?nPerson2.
```

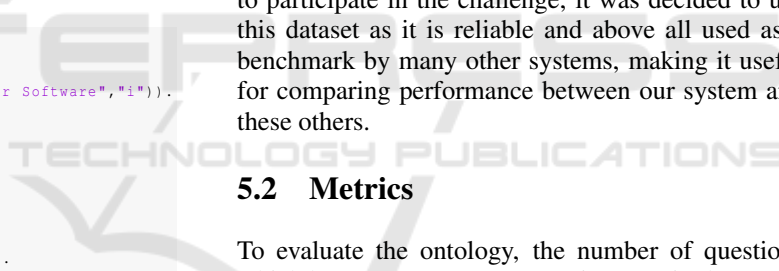
```

FILTER(regex(?nPerson2,"Bill Gates","i")).
?birthProp rdfs:label ?lbirthP.
FILTER(regex(?lbirthP,"birth place","i")).
{?Born3 ?birthProp ?Place1.
?Place1 rdfs:type ?tP1.
?tP1 rdfs:label ?lP1.
FILTER(regex(?lP1,"country","i"))}
UNION
{?Born3 ?birthProp ?cityPlace1.
?cityPlace1 ?cProp ?Place1.
?cProp rdfs:label ?lcP.
FILTER(regex(?lcP,"country","i"))}
}

```

Listing 1: SPARQL Query result from Query Ontology solution for Question 1.

4.4.2 SPARQL Query for Question 2

The SELECT clause is composed by the *Object1*. There is an *has_name* property for *X3*, and the individuals with label "Mean Hamster Software" will be matched for variable *X3*. As object properties, there are *who_make* and *make_what*, which are translated as seen before. In addition, the *has_text* for software is translated, to put a constraint on the *Object* type (software, in this case). As the translation of *who_make* and *make_who* was previously shown, below is presented the query created for the question 'What software has been published by Mean Hamster Software?'.


```

SELECT DISTINCT ?Object1
WHERE {
?Make2 rdfs:label ?nOrganisation3.
FILTER(regex(?nOrganisation3,"Mean Hamster Software","i")).
?Object1 ?htProp1 ?Make2.
?htProp1 rdfs:label ?lhtProp1.
FILTER(regex(?lhtProp1,"publish","i"))}
UNION
{?Make2 ?htProp1 ?Object1.
?htProp1 rdfs:label ?lhtProp1.
FILTER(regex(?lhtProp1,"publish","i"))}
?Object1 rdfs:type ?httObject1.
FILTER(regex(?httObject1,"software","i")).
}

```

Listing 2: SPARQL Query result from Query Ontology solution for Question 2.

4.4.3 SPARQL Query for Question 3

The last example is question 3 : 'What is the area code of Berlin?'. The SELECT Clause is formed by *ConceptPlace1*, which is in select relation. There is one *has_name* property, linked to *Place3*, whit value 'Berlin'. The triples fixing *Place3* with label 'Berlin' are added. The object property *conceptOfPlace* is translated as shown in subsection 4.3. As exist a property, the *has_text* translation for object property is added. At the end, *what_is* and *is_what* are used to replace the variables in the query, changing the body and the SELECT too. Below, it is shown the final query.

```

SELECT DISTINCT ?Be4
WHERE {
?Place3 rdfs:label ?nPlace3.

```

```

FILTER(regex(?nPlace3,"Berlin","i")).
?Place3 ?htProp3 ?Be4.
?htProp3 rdfs:label ?lhtProp3.
FILTER(regex(?lhtProp3,"area code","i")).
}

```

Listing 3: SPARQL Query result from Query Ontology solution for Question 3.

5 EXPERIMENTS ON DBpedia

In this section, it will described the experiments carried out to evaluate our system. As a target KB, DBpedia was chosen, as one of the best known KBs.

5.1 Dataset

The dataset chosen is the one provided by the QALD-9 Challenge (Ngomo, 2018). The QALD-9 dataset is divided into 2 parts: a training part (consisting of 408 questions) and the test part (consisting of 150 questions). The dataset is formatted in JSON and each question contains: the natural language question (up to 11 different translations), the SPARQL query and the answer to the query. Although our objective is not to participate in the challenge, it was decided to use this dataset as it is reliable and above all used as a benchmark by many other systems, making it useful for comparing performance between our system and these others.

5.2 Metrics

To evaluate the ontology, the number of questions which have a Correct Interpretation (CI) in the Query Ontology are used, which means that there is a possible correct interpretation of the question in Query Ontology by the correct assignation of Classes to DRs of partial DRS and correct assignation of object and data properties for the relations between DR. For this experiment, the CI evaluation was carried out manually. To evaluate the mapping description rules, the number of Equal Answers (EA) out of the dataset is chosen. For each question, the answer from the query generated with the methodology described is compared with the answer in the dataset: if they are equal, the question is marked as equal and counted in the final EA results, otherwise counted as incorrect. In addition, it is calculated the number of Correct Answers (CA). The calculation of this metric is defined as follow: if the resulting answer is coherent with the constraints and definitions in natural language question, it is evaluated as correct, otherwise not. This metric was chose as methodology presented in this paper use

a retrieving system based on lemma, which can bring to retrieve more individuals than the ones in dataset.

5.3 Results Evaluation

The Table 4 presents the results on 100 questions from test set of QALD-9. The subset includes most of the wh-questions in dataset, with question about places, dates, people and organisations. The full results of the experiment are available on a GitHub Repository⁴.

Table 4: Evaluation of Query Ontology and SPARQL Builder.

Correct Int.	Equal Ans.	Correct Ans.
1	0.37	0.56

As shown in (Ngomo, 2018), is presented the maximum precision achieved: 0.293 . Although the two scores are not directly comparable, as in the experiment presented in this article a portion of the entire dataset is tested and the precision shown in the cited article is calculated differently, can be stated that the results obtained are promising when compared with the other models.

The Table 4 shows a 20% discrepancy between Equal Answer and Correct Answer values. This is due to the methodology presented recovering additional answers and some QALD-9 answers being outdated. The main problems encountered are in the mapping of the solution given by the Query Ontology through the mapping description rules, as this is still a work in progress. This is due to the structure of DBpedia which includes different information, it is not always easy to create rules which can generalise certain links. For instance, many resources are often linked through the object property `dbo:wikiPageWikiLink`. It is often unclear to identify the domain and range of relationships through a given lemma and its properties. This issue is related to DBpedia's vocabulary for certain relations. Sometimes, terms in natural language can be directly mapped to DBpedia terms. For example, in the question 'Who write Harry Potter?', the action 'write' maps to the `dbo:author` property. This can be managed by mapping specific terms. For this experiment, the mapping was done manually, but a future module could automate this. A NER system, especially for proper names, can aid in interpreting DRs and defining resources in the target KB. Currently, a NER system uses manually constructed Gazetteers. In the future, a Gazetteer-based or different NER system may be tested.

⁴<https://github.com/dvaragnolo/NLP-QA-DBPEDIA>

6 CONCLUSIONS

A methodology for translating natural language queries into SPARQL queries for DBpedia was presented. The QALD-9 experiment showed promising results, highlighting the methodology's potential. However, current issues, especially in translating Ontology Queries to SPARQL, were identified. Suggested modifications could improve the methodology for future experiments.

REFERENCES

- Cornei, L.-M. and Trandabat, D. (2023). Dbspark: A system for natural language to sparql translation. In *International Conference on Research Challenges in Information Science*, pages 157–170. Springer.
- Hirschman, L. and Gaizauskas, R. (2001). Natural language question answering: the view from here. *natural language engineering*, 7(4):275–300.
- Hu, X., Shu, Y., Huang, X., and Qu, Y. (2021). Edg-based question decomposition for complex question answering over knowledge bases. In *The Semantic Web- ISWC 2021: 20th International Semantic Web Conference, ISWC 2021, Virtual Event, October 24–28, 2021, Proceedings 20*, pages 128–145. Springer.
- Kamp, H. and Reyle, U. (2013). *From discourse to logic: Introduction to modeltheoretic semantics of natural language, formal logic and discourse representation theory*, volume 42. Springer Science & Business Media.
- Ngomo, N. (2018). 9th challenge on question answering over linked data (qald-9). *language*, 7(1):58–64.
- Omar, R., Dhall, I., Kalnis, P., and Mansour, E. (2023). A universal question-answering platform for knowledge graphs. *Proceedings of the ACM on Management of Data*, 1(1):1–25.
- Silva, J. Q., Melo, D., Rodrigues, I. P., Seco, J. C., Ferreira, C., and Parreira, J. (2023). An ontology-based task-oriented dialogue to create outsystems applications. *SN Computer Science*, 4(12):1–17.
- Steinmetz, N., Arning, A.-K., and Sattler, K.-U. (2019). From natural language questions to sparql queries: a pattern-based approach.
- Varagnolo, D., Melo, D., and Rodrigues, I. (2023). An ontology-based question-answering, from natural language to sparql query. In *Proceedings of the 15th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 2: KEOD*, pages 174–181. INSTICC, SciTePress.
- Zlatareva, N. and Amin, D. (2021). Natural language to sparql query builder for semantic web applications. *Journal of Machine Intelligence and Data Science (JMIDS)*, 2:34.