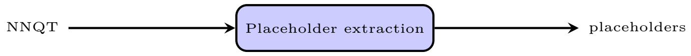
1 Base Model

Starting point is the LC-QuAD 2.0 dataset, whose creation is explained in section 6.

Process Diagram 1: Placeholder extraction



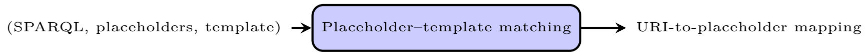
Process Description:
From a Normalized Natural Question Template, extract the placeholders with regex pattern matching.

Example:

Input:
```
What is the {periodical literature} for {mouthpiece} of {Delta
Air Lines}.
```

Output:
```
[periodical literature, mouthpiece, Delta Air Lines]
```

Process Diagram 2: Placeholder–template matching


(SPARQL, placeholders, template)
Placeholder–template matching
URI-to-placeholder mapping

Process Description:
Both the queries and NNQTs follow a specific pattern depending on the sample's template. Hence, a mapping from URI to resp. placeholder can be determined.

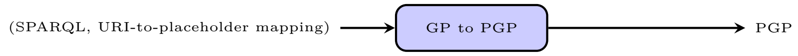
Example:

Input:
```
(select distinct ?obj where { wd:Q188920 wdt:P2813 ?obj . ?obj
wdt:P31 wd:Q1002697 }, [periodical literature, mouthpiece, Delta
Air Lines], <S P ?O ; ?O instanceOf Type>)
```

Output:

```
{ "wd:Q188920": "Delta Air Lines",
"wdt:P2813": "mouthpiece",
"wd:81002697": "periodical literature",
"wdt:P31": "instanceOf" }
```

Process Diagram 3: GP to PGP



Process Description:
With the mapping, any URI in the original query can be replaced with its resp. placeholder, i.e., Graph Pattern to Placeholder Graph Pattern.

Example:

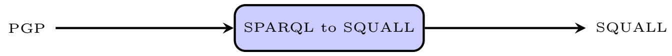
Input:

```
(select distinct ?obj where { wd:Q188920 wdt:P2813 ?obj . ?obj
wdt:P31 wd:Q1002697 }, { "wd:Q188920": "Delta Air Lines",
"wdt:P2813": "mouthpiece",
"wd:81002697": "periodical literature",
"wdt:P31": "instanceOf" }
```

Output:

```
select distinct ?obj where { <Delta Air Lines> <mouthpiece> ?obj
. ?obj <instance of> <periodical literature> . }
```

Process Diagram 4: SPARQL to SQUALL



Process Description:
SPARQL can be mapped to SQUALL, regardless if it contains URIs (GP) or placeholders (PGP), thus a SQUALL expression is obtained for the PGP.

Example:

Input:

```
select distinct ?obj where { <Delta Air Lines> <mouthpiece> ?obj
. ?obj <instance of> <periodical literature> . }
```

Output:

```
What is a <mouthpiece> of <Delta Air Lines> and has <instance of>
<periodical literature>?
```

Process Diagram 5: Prompt function

(question, SQUALL)
Prompt function
Prompt

Process Description:
Instantiates a prompt for a specific question–SQUALL pair.

Example:

Input:

```
(What periodical literature does Delta Air Lines use as a
mouthpiece?, What is a <mouthpiece> of <Delta Air Lines> and has
<instance of> <periodical literature>?)
```

Output:

```
 Below is an instruction that describes a task, paired with an
input that provides further context. Write a response that
appropriately completes the request.

### Instruction:
Given the following sentence, your job is to generate SQUALL for
it in the JSON format.

### Input:
What periodical literature does Delta Air Lines use as a
mouthpiece?

### Response:
{ "input": "What periodical literature does Delta Air Lines use
as a mouthpiece?", "squall": "What is a <mouthpiece> of <Delta
Air Lines> and has <instance of> <periodical literature>?" }
```

Process Diagram 6: LLM

prompt
LLM
generated output

Process Description:
Given the prompt, the LLM is trained to “replicate” it.

2 Index Creation

The ontology is mapped to an RDF dataset consisting of triples (subject, predicate, object), from an index is created.

Process Diagram 7: Get rdfs:label

RDF triple
Get rdfs:label
textual triple

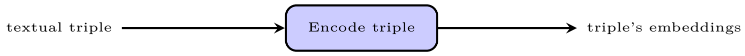
Process Description:
Get the rdfs:label of the subject, predicate and object. For triple elements that are literals instead of URIs, the literal itself is returned, since it is already a string.

Example:

Input:
`(http://orkg.org/orkg/resource/R57674, http://orkg.org/orkg/predicate/P27, "Woo E")`

Output:
`("The interaction between ...", "author", "Woo E."`

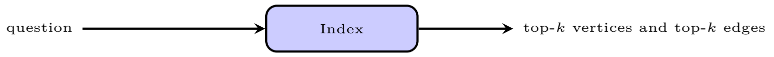
Process Diagram 8: Encode triple



Process Description:
Use a text encoder model to map the subject, predicate and object text to embeddings.

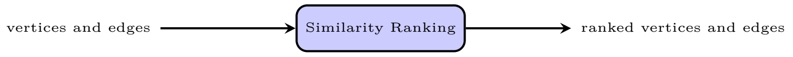
3 RAG Model

Process Diagram 9: Index



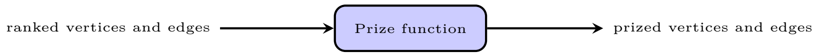
Process Description:
For a given question string, retrieve the top-k vertices and top-k edges with the highest similarity between the string and their respective rdfs:labels.

Process Diagram 10: Similarity Ranking



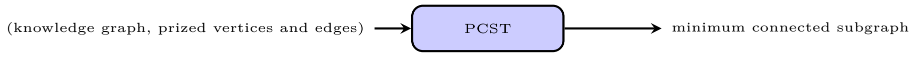
Process Description:
Rank a set of vertices and edges with related similarity scores from high to low.

Process Diagram 11: Prize function



Process Description:
Assign prizes to vertices and edges based on rank: the higher its rank the higher its prize.

Process Diagram 12: PCST


(knowledge graph, prized vertices and edges)
PCST
minimum connected subgraph

Process Description:
Given the knowledge graph, use the PCST algorithm to find the connected subgraph of the KG that maximizes the total prize minus its cost (proportional to its size).

Complete example from question to subgraph.

Example:

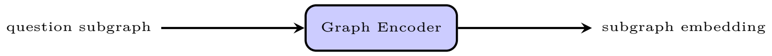
Input:

```
Provide a list of papers that have utilized the Depth DDPPO model
and include the links to their code?
```

Output:

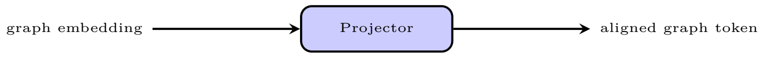
```
 node_id,node_attr,node_uri
346,model,http://orkg.org/orkg/class/Model
26311,has model,http://orkg.org/orkg/predicate/HAS_MODEL
52733,has source
code,http://orkg.org/orkg/predicate/HAS_SOURCE_CODE
63664,contribution depth
ddppo,http://orkg.org/orkg/resource/R131294
200736,depth ddppo,http://orkg.org/orkg/resource/R123481
...

src,edge_attr,dst,edge_uri
111,Has Datasets,5274,http://orkg.org/orkg/predicate/P41003
26311,type,226,http://www.w3.org/1999/02/22-rdf-syntax-ns#type
63664,has model,200736,http://orkg.org/orkg/predicate/HAS_MODEL
...
```

Process Diagram 13: Graph Encoder



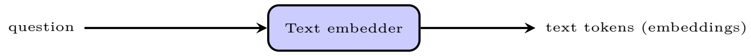
Process Description:
Make a graph embedding of the minimum connected subgraph using a graph encoder.

Process Diagram 14: Projector



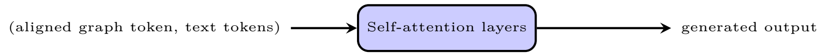
Process Description:
The graph embedding is aligned with the text embedding space using a projector which is a trainable alignement layer.

Process Diagram 15: Text embedder



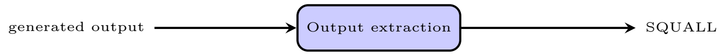
Process Description:
Using the text embedder of the frozen and finetuned base model, produce the text tokens.

Process Diagram 16: Self-attention layers



Process Description:
The self-attention layers receive extra context from the graph token and the task from the text tokens, to generate output containing SQUALL.

Process Diagram 17: Output extraction



Process Description:
Using regex, extract SQUALL from the generated output.

Example:

Input:

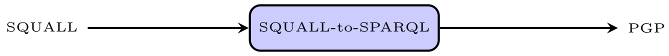
```
 Below is an instruction that describes a task, paired with an
input that provides further context. Write a response that
appropriately completes the request.

### Instruction:
Given the following sentence, your job is to generate SQUALL for
it in the JSON format.

### Input:
Provide a list of papers that have utilized the Depth DDPPO model
and include the links to their code?
```

```
### Response:
{ "input": "Provide a list of papers that have utilized the Depth
DDPPO model and include the links to their code?", "squall":
"What is a <has source code> of a thing that has <has model> a
<Model> X and has a <has benchmark> that has a <has dataset> and
X has a rdfs:label whose string matches 'Depth DDPPO'?" }
```

Output:

```
What is a <has source code> of a thing that has <has model> a
<Model> X and has a <has benchmark> that has a <has dataset> and
X has a rdfs:label whose string matches 'Depth DDPPO'?
```

Process Diagram 18: SQUALL-to-SPARQL



Process Description:
Using the SQUALL-to-SPARQL translator, map SQUALL to PGP.

Example:

Input:

```
What is a <has source code> of a thing that has <has model> a
<Model> X and has a <has benchmark> that has a <has dataset> and
X has a rdfs:label whose string matches 'Depth DDPPO'?
```

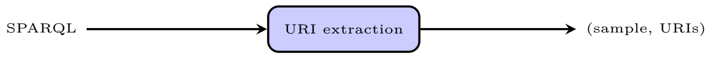
Output:

```
select distinct ?code where { ?model a <Model>; rdfs:label
?model_lbl . filter(str(?model_lbl) = 'Depth DDPPO') ?benchmark
<has dataset> ?dataset . ?cont <has benchmark> ?benchmark . ?cont
<has model> ?model; <has source code> ?code . }
```

4 Training RAG Model

To train the RAG model, we need ground truth SQUALL, this is created as follows. The loss between the generated and ground truth SQUALL is used to update the weights of the non-frozen layers of the RAG model: graph encoder and projector. Starting point is the SciQA dataset, whose creation is explained in section 7.

Process Diagram 19: URI extraction



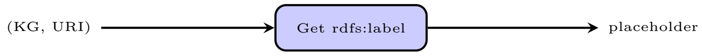
Process Description:
Extract the URIs from a SPARQL query.

Example:

Input:
`select distinct ?code where { ?model a orkgc:Model; rdfs:label ?model_lbl . filter (str(?model_lbl) = 'Depth DDPPO') ?benchmark orkgp:HAS_DATASET ?dataset . ?cont orkgp:HAS_BENCHMARK ?benchmark . ?cont orkgp:HAS_MODEL ?model; orkgp:HAS_SOURCE_CODE ?code . }`

Output:
`[a, orkgc:Model, rdfs:label, orkgp:HAS_DATASET, orkgp:HAS_BENCHMARK, orkgp:HAS_MODEL, orkgp:HAS_SOURCE_CODE]`

Process Diagram 20: Get rdfs:label



Process Description:
Query the knowledge graph for the rdfs:label of a URI, either from a vertex or edge.

Example:

Input:
`(ORKG, orkgp:HAS_DATASET)`

Output:

```
has dataset
```

Process Diagram 21: GP to PGP

(KG, GP)
GP to PGP
PGP

Process Description:
Use “URI extraction”, followed by “Get rdfs:label” for each URI, and replace URI with placeholder, to map each URI to a placeholder: GP to PGP.

Example:

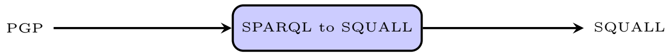
Input:

```
(ORKG, select distinct ?code where { ?model a orkgc:Model;
rdfs:label ?model_lbl . filter (str(?model_lbl) = 'Depth DDPPO')
?benchmark orkgp:HAS_DATASET ?dataset . ?cont orkgp:HAS_BENCHMARK
?benchmark . ?cont orkgp:HAS_MODEL ?model; orkgp:HAS_SOURCE_CODE
?code . },
```

Output:

```
select distinct ?code where { ?model a <Model>; rdfs:label
?model_lbl . filter(str(?model_lbl) = 'Depth DDPPO') ?benchmark
<has dataset> ?dataset . ?cont <has benchmark> ?benchmark . ?cont
<has model> ?model; <has source code> ?code . }
```

Process Diagram 22: SPARQL to SQUALL



Process Description:
Map a PGP to a SQUALL expression using the SPARQL to SQUALL translator or by manual construction.

Example:

Input:

```
select distinct ?code where { ?model a <Model>; rdfs:label
?model_lbl . filter(str(?model_lbl) = 'Depth DDPPO') ?benchmark
<has dataset> ?dataset . ?cont <has benchmark> ?benchmark . ?cont
<has model> ?model; <has source code> ?code . }
```

Output:

```
What is a <has source code> of a thing that has <has model> a
<Model> X and has a <has benchmark> that has a <has dataset> and
X has a rdfs:label whose string matches 'Depth DDPPO'?
```

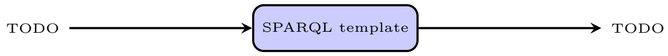
5 Linking

6 LC-QuAD 2.0 Dataset Creation

Process Diagram 23: Wikipedia vital entity list

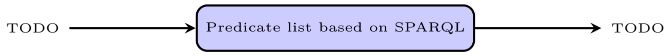
Process Description:
TODO

Process Diagram 24: SPARQL template



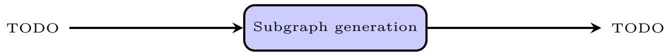
Process Description:
TODO

Process Diagram 25: Predicate list based on SPARQL



Process Description:
TODO

Process Diagram 26: Subgraph generation



Process Description:
TODO

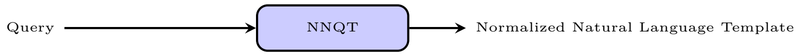
Process Diagram 27: Template fitting

Process Description:
TODO

Process Diagram 28: SPARQL

Process Description:
TODO

Process Diagram 29: NNQT

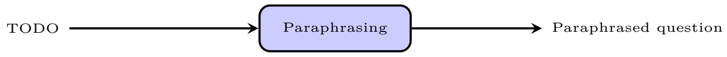


Process Description:
TODO

Process Diagram 30: Verbalization

Process Description:
TODO

Process Diagram 31: Paraphrasing



Process Description:
TODO

7 Sci-QA