Translating Natural Language Queries to SPARQL

Shreya Bhajikhaye

Table of Contents

Introduction

- Question Answering Systems can be considered an advanced form of Information Retrieval systems
- Answer questions posed by humans in natural language
- Search through a structured knowledge base or unstructured collection of documents
- Closed domain or Open domain
- Transform English language questions to SPARQL query for Wikidata

Background

Semantic Web & RDF

- “The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries.”
- The documents on the web are in various formats like XML, HTML, relational etc.
- The Resource Description Framework (RDF) models different formats of data to a machine-readable format.
- Resource descriptions in RDF are expressed as triples.

Background

RDF

- Example triple : <subject> <predicate> <object>

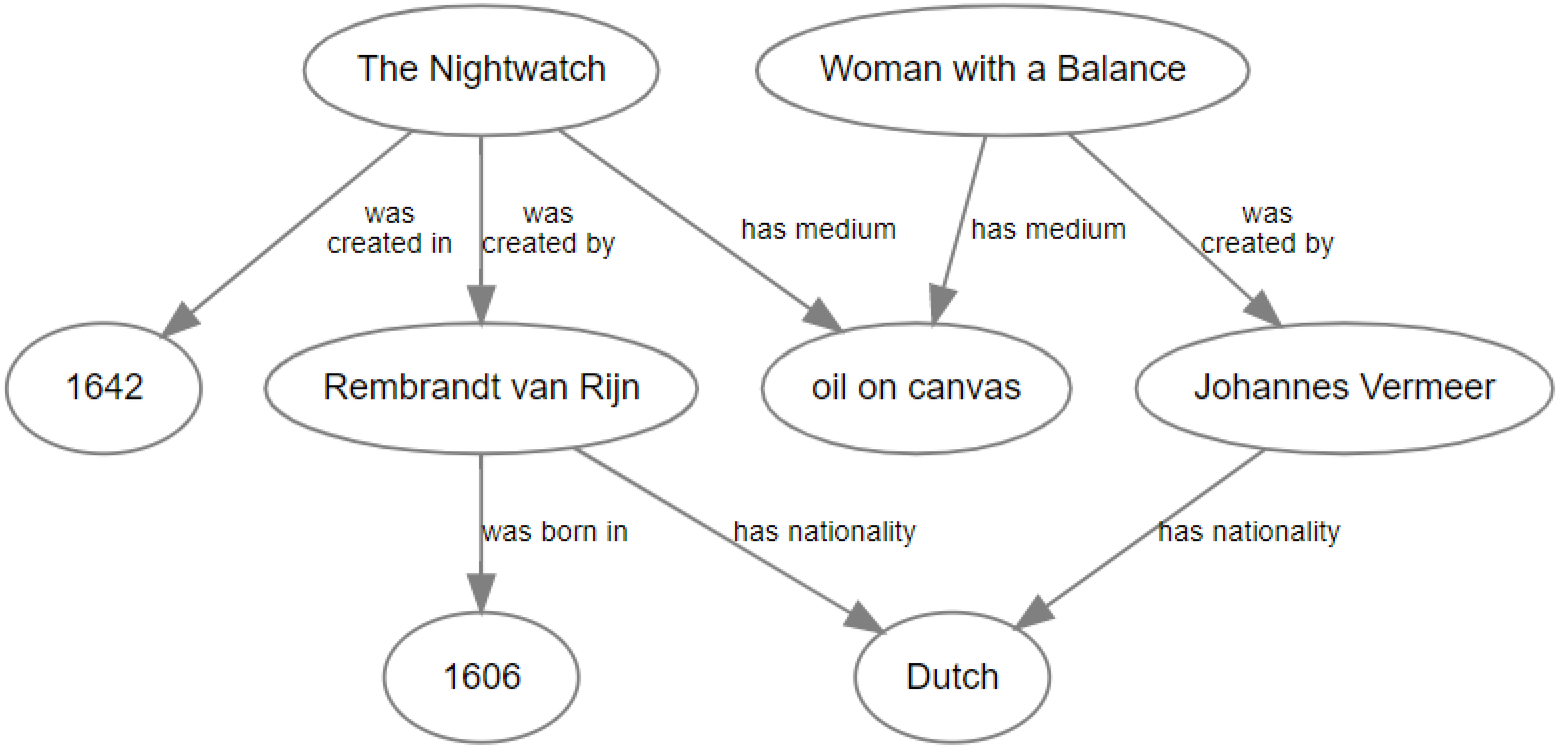
```
<The Nightwatch> <was created by> <Rembrandt van Rijn> .
```

- RDF database records

```
...
<The Nightwatch> <was created by> <Rembrandt van Rijn> .
<The Nightwatch> <was created in> <1642> .
<The Nightwatch> <has medium> <oil on canvas> .
<Rembrandt van Rijn> <was born in> <1606> .
<Rembrandt van Rijn> <has nationality> <Dutch> .
<Johannes Vermeer> <has nationality> <Dutch> .
<Woman with a Balance> <was created by> <Johannes Vermeer> .
<Woman with a Balance> <has medium> <oil on canvas> .
...
```

Background

RDF



Background

SPARQL

- SPARQL is the standard language to query graph databases represented in the RDF format.
- Two components :
  - SELECT clause defines the output variables
  - WHERE clause provides basic graph pattern

```
SELECT <variables>
WHERE {
   <graph pattern>
}
```

Background

SPARQL

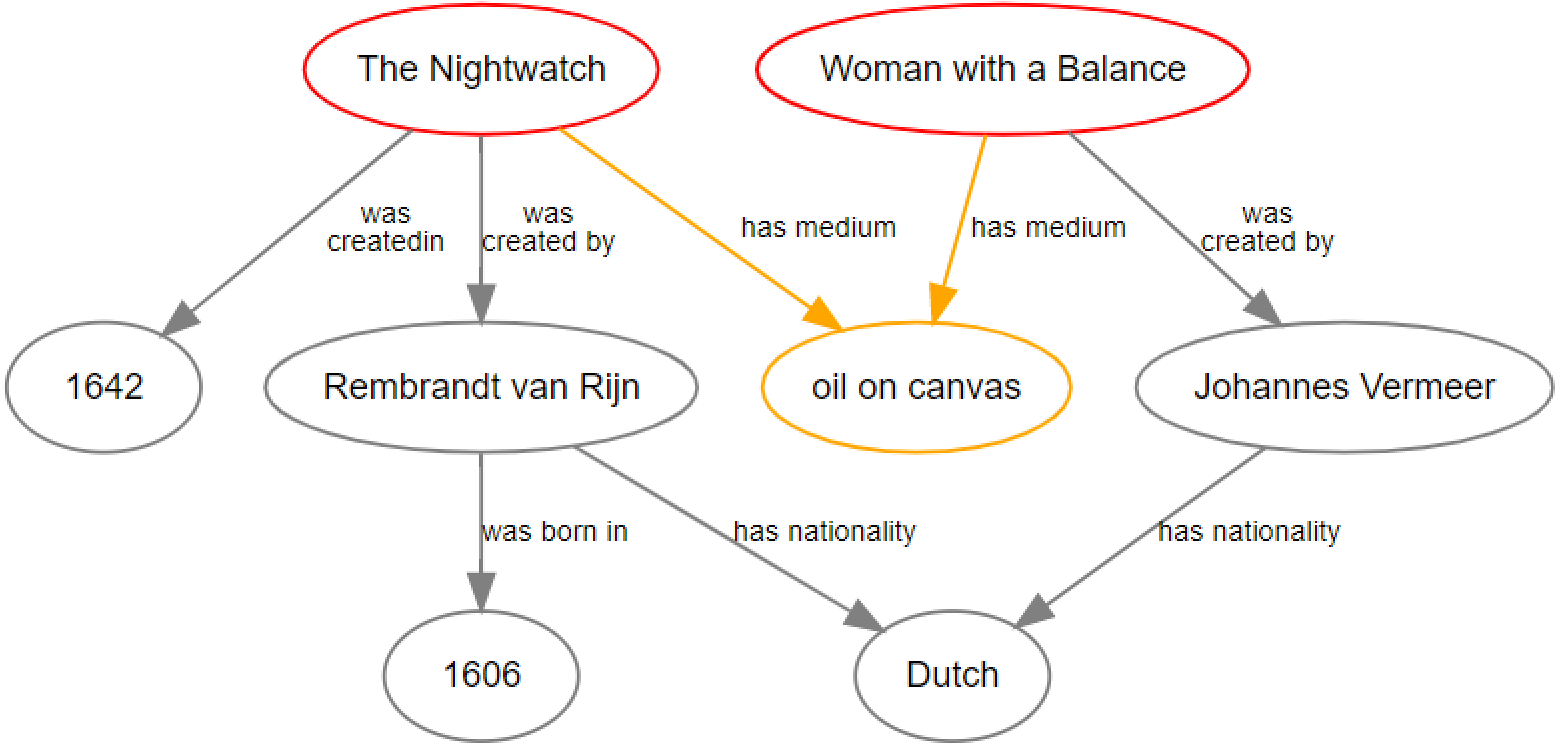
- Search for paintings that have medium as oil on canvas

```
SELECT ?painting
WHERE {
  ?painting <has medium> <oil on canvas> .
}
```

painting
The Nightwatch
Woman with a Balance

Background

SPARQL



Background

SPARQL

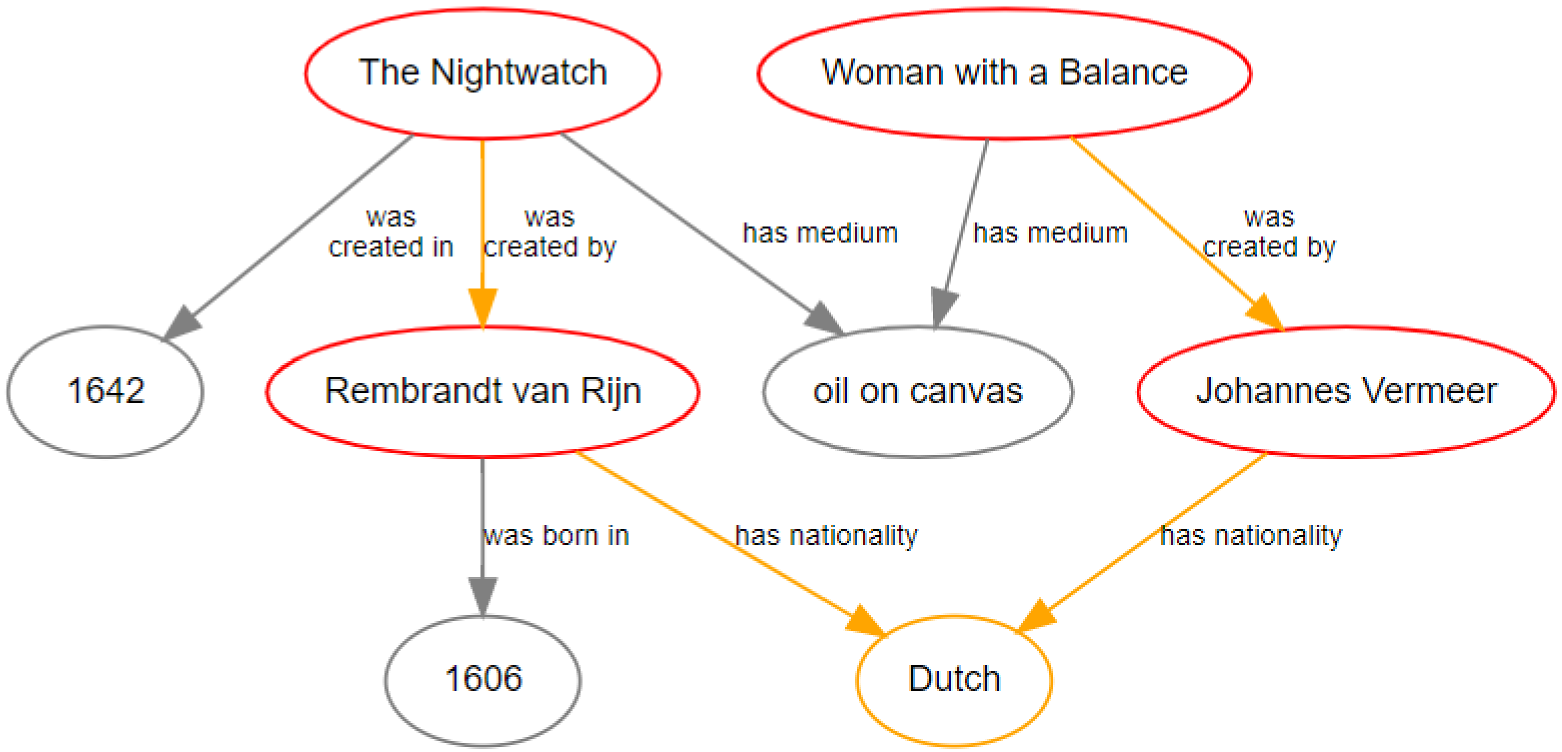
- Complex queries : Paintings by any artist who is Dutch

```
SELECT ?artist ?painting
WHERE {
  ?artist <has nationality> <Dutch> .
  ?painting <was created by> ?artist .
}
```

artist	painting
Rembrandt van Rijn	The Nightwatch
Johannes Vermeer	Woman with a Balance

Background

SPARQL



Background

SPARQL Wikidata Query

- Refer to every document and relation by its IRI
- Namespace : wd and wdt

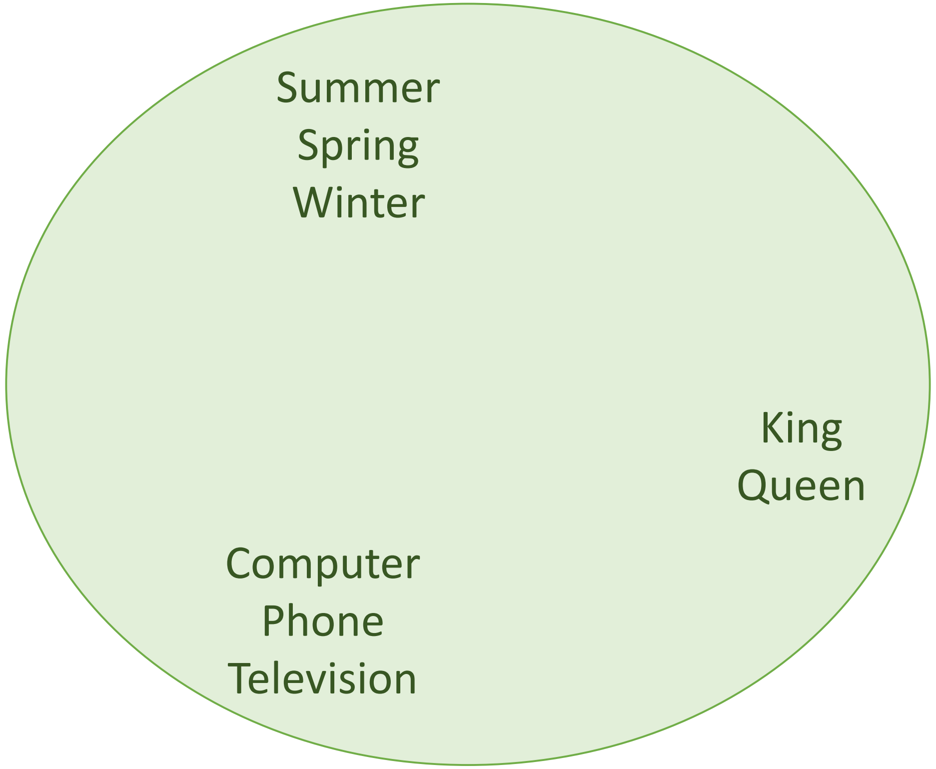
```
SELECT ?artist ?painting
WHERE {
  ?artist <has nationality> <Dutch> .
  ?painting <was created by> ?artist .
}
```

```
SELECT ?artist ?painting
WHERE {
  ?artist     wdt:P27      wd:Q170072 .
  ?painting   wdt:P170     wd:Q5598 .
}
```

Background

Word Vectorization

- One hot vector
  - Represent categorical data as a binary vector
  - [red, green, green] = [[1,0] ,[0,1] ,[0,1]]
- Word Embedding
  - Conveys the meaning of the word in a numerical format
  - Words with similar meaning lie closer to each other in the vector space



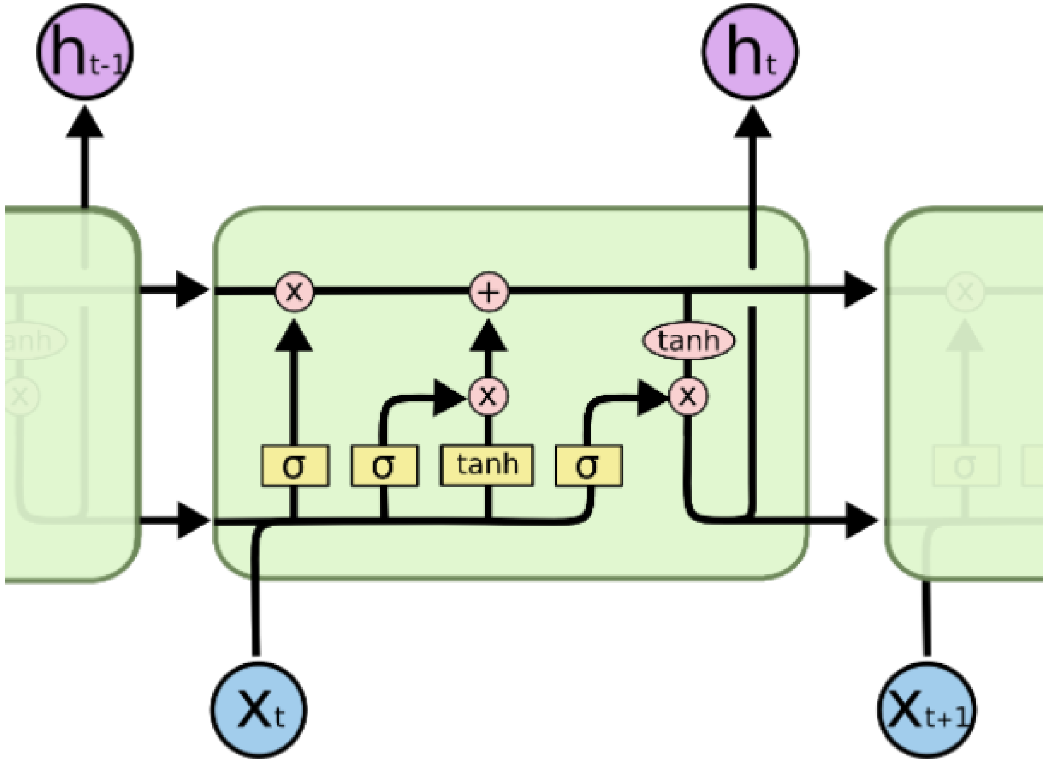
Background

Recurrent Neural Networks

- Allows its own output to be used as input
- Vanishing gradient problem

LSTM

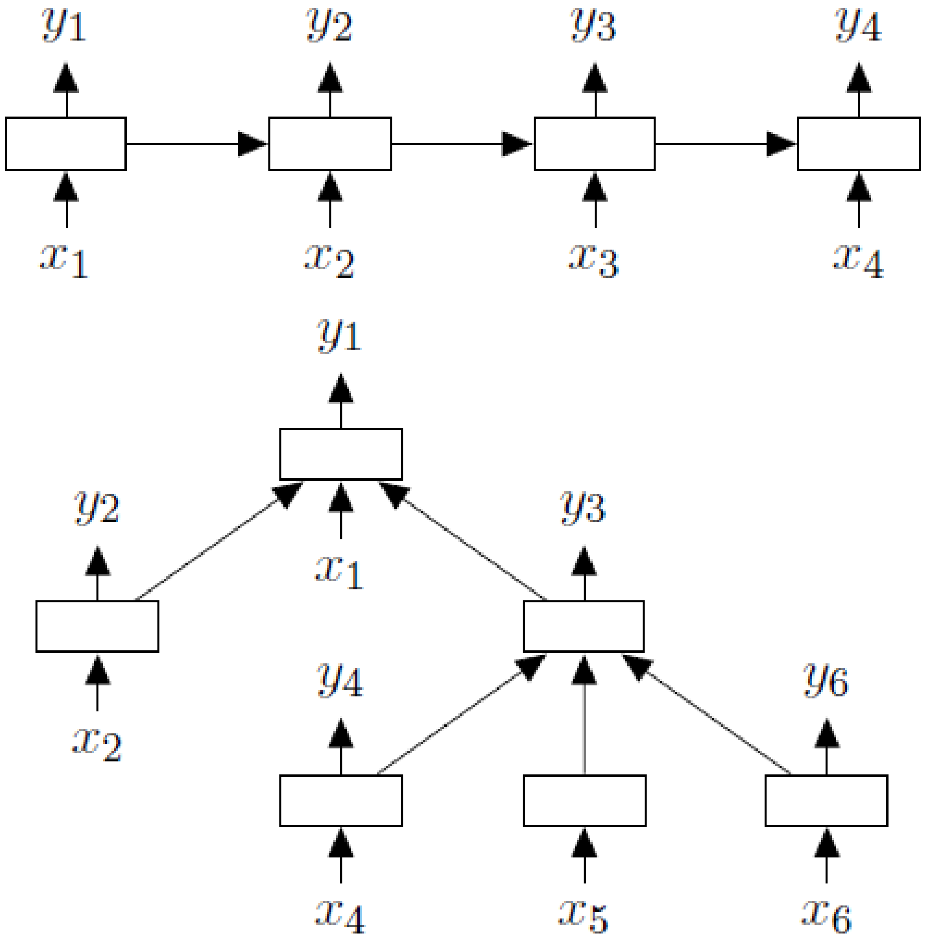
- Add memory cell to preserve long term dependencies
- Use gating to control information flow



Background

Recursive Neural Networks

- Apply the same set of weights recursively on a structured input
- Improve encoding of sentences using their structure



Tree-LSTM

- Generalization of the LSTM model
- Tree-structured input
- Useful in semantic relatedness and sentiment classification

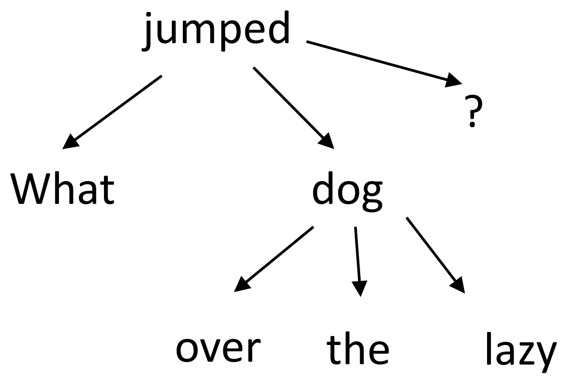
Background

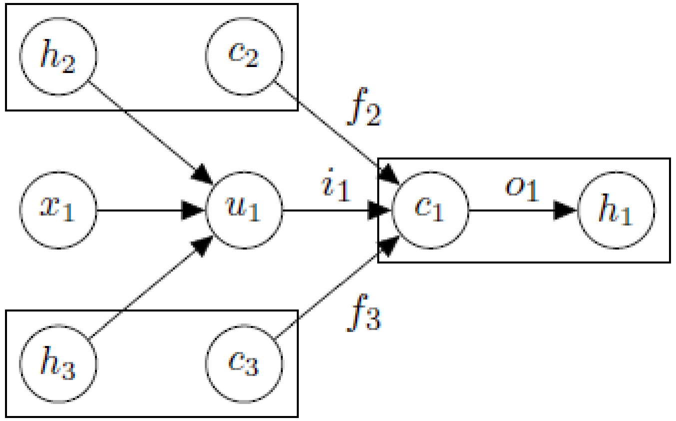
Child Sum Tree-LSTM

- Children output and memory cell are summed
- Does not take into account child order
- Works with variable number of children
- Shares gate weight between children
- Used in dependency Tree-LSTM

$$\widetilde{h_j} = \sum_{k \in child(j)} h_k$$

Example: What jumped over the lazy dog ?





Dataset Used

Lc-QuAD dataset

- 30,000 questions in English language across 38 templates
- Query types – list, boolean and count
- ~6500 list queries across 8 unique SPARQL template
- Dataset create with 600 questions across 3 templates


```
{
  "template": "E REF ?F",
  "template_id": "1",
  "question": "What is the capital of Denmark?",
  "NNQT_question": "What is <capital city> of <Denmark> ?",
  "sparql_wikidata":
  "select distinct ?answer where { wd:Q35 wdt:P36 ?answer}"
}
```


ID	SPARQL Query Template	Total Count
1	SELECT DISTINCT ?uri WHERE { <S> <P> ?uri }	3304
2	SELECT DISTINCT ?uri WHERE { ?uri <P> <O> }	740
3	SELECT DISTINCT ?uri WHERE { <S> <P1> ?uri . ?uri <P2> <O> }	2505

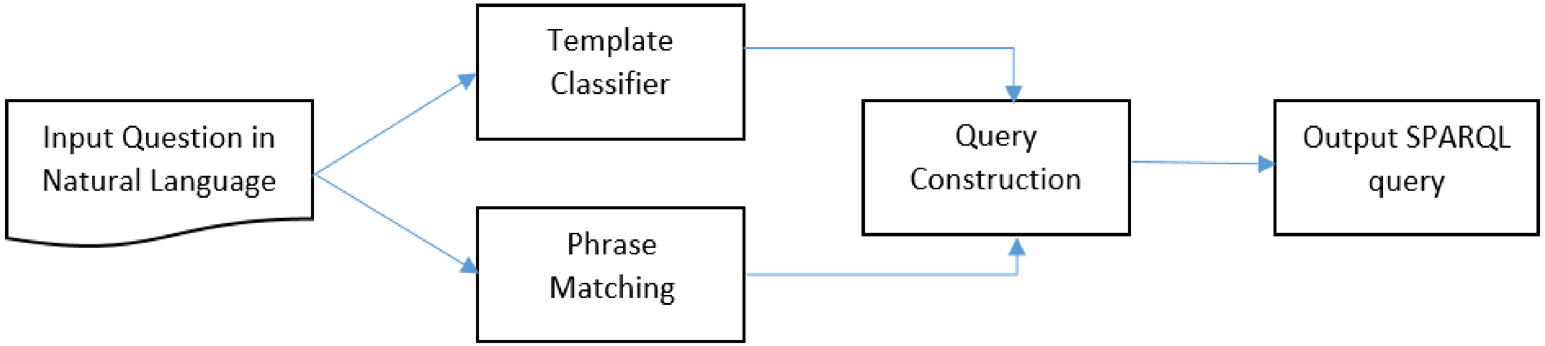
Dataset Used

Template ID	SPARQL Question Template	Total Count
1	SELECT DISTINCT ?uri WHERE { <S> <P> ?uri }	3304
2	SELECT DISTINCT ?uri WHERE { ?uri <P> <O> }	740
3	SELECT DISTINCT ?uri WHERE { <S> <P1> ?uri . ?uri <P2> <O> }	2505
4	SELECT DISTINCT ?uri WHERE { <S> <P1> <O> . <O> <P2> ?uri }	3713
5	SELECT DISTINCT ?uri WHERE { <S> <P1> ?obj . ?obj <P2> ?uri }	2969
6	SELECT DISTINCT ?uri WHERE { <S> <P1> <O> . <O> <P2> ?uri }	2943
7	SELECT DISTINCT ?uri WHERE { ?uri <P> <O> . ?uri rdf:instance <O> }	2042
8	SELECT DISTINCT ?uri WHERE { <S> <P> ?uri. ?uri rdf:instance <O> }	1872

System Design

Main components of the proposed system are as follows:

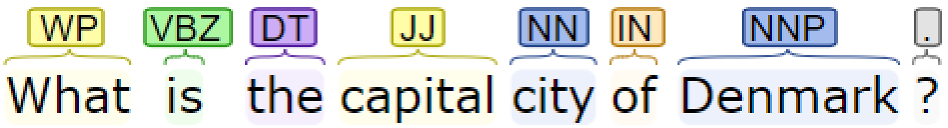
- Question Analysis
- Template Classification
- Phrase Matching
- Query Construction



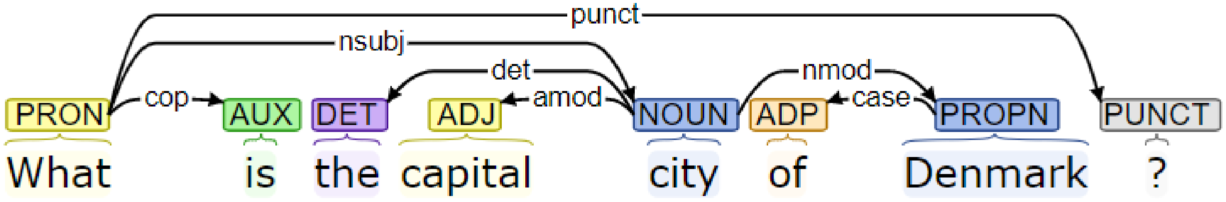
System Design

Question Analysis

- Stanza library for text analysis
- Part-of-speech tagging : annotate tokens



- Dependency parsing : build the dependency parse tree

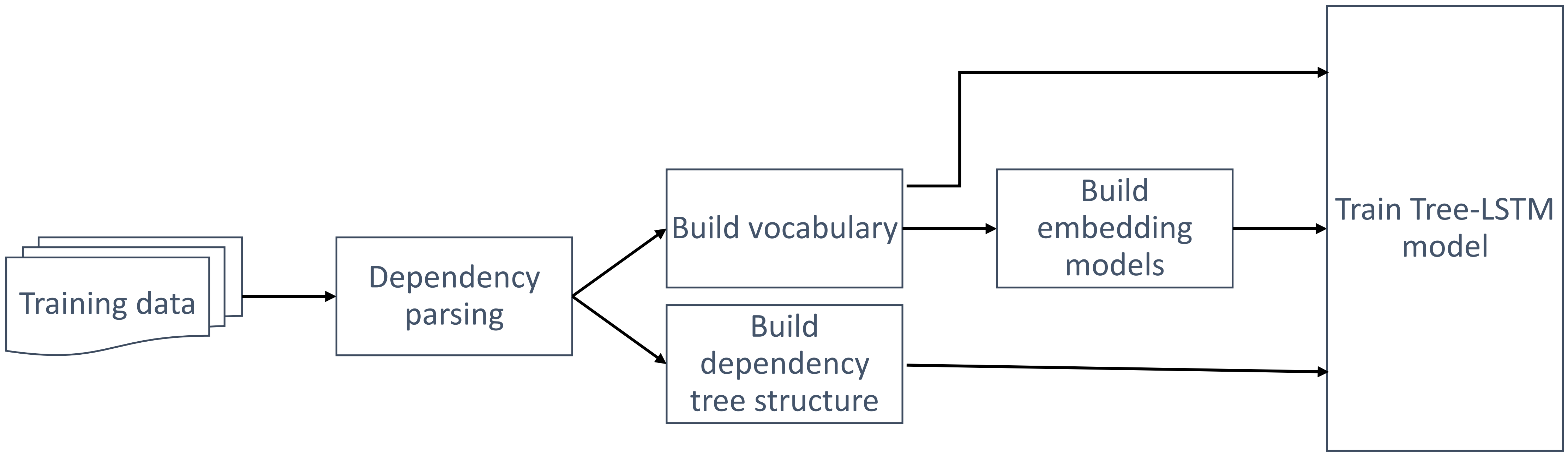


System Design

Template Classification

- Identify the type of SPARQL query equivalent to the input question
- Tree-LSTM model implemented with PyTorch library
- Feature set
  - Tokens
  - POS tags
  - Syntactic tree structure
  - Relationship dependency tags
  - Characters

System Design



System Design

Phrase Matching

- Named Entity Recognition
- Entity and Relation Linking with Wikidata
- Falcon 2.0 library

Example: What is the capital of Denmark?


```
{
  "entities_wikidata": [
    [
      "<http://www.wikidata.org/entity/Q35>",
      "Denmark"
    ]
  ],
  "relations_wikidata": [
    [
      "<http://www.wikidata.org/entity/P36>",
      "capital"
    ]
  ]
}
```

System Design

Query Construction

- Template classification captures the semantic structure of the user question with slots to be filled
- Entities and predicates to be filled from phrase matching phrase

Example: What is the capital of Denmark?

Identified SPARQL template :

```
SELECT DISTINCT ?answer WHERE { ?answer wdt:<P> wd:<R> }
```

Query constructed :

```
SELECT DISTINCT ?answer WHERE { ?answer wdt:P36 wd:Q35 }
```

Analysis and Results

Experiment Design

- The system was deployed on Google collaboratory.
- The Lc-QuaD dataset consisted of ~6000 English questions and their equivalent SPARQL query.
- To improve the dataset, 600 questions were cleaned by correcting their grammar and the SPARQL template id for 3 classes.
- This was separated into a training dataset of 480 questions and a testing dataset of 120 questions that was used to evaluate the Tree-LSTM classification model.
- The testing dataset of 120 questions was used to verify the query results of the system.
- The model was trained for 20 epochs for each experiment.

Analysis and Results

Template Classification

- Composition of dataset

	Composition of the training dataset	Correctly identified templates	Total data size(training / test)
Uncleaned full set records of three templates	Template 1 (3327) + Template 2 (740)+ Template 3 (2505)	56.5%	6572 (5258 /1314)
Cleaned dataset with subset of two templates	Template 1 (200) + Template 2 (200)	70%	400 (320 / 80)
Cleaned dataset with subset of three templates	Template 1 (200) + Template 2 (200) + Template 3 (200)	72.83%	600 (480 / 120)

Analysis and Results

Template Classification

- Feature Selection

Test data Composition	Feature List	Accuracy
Template 1 + Template 2 (320 training records / 80 test records)	Dependency Tree + Word Embedding	62.5%
	Dependency Tree + Parts-of-speech + Word embedding	65.5%
	Dependency Tree + Parts-of-speech + Relation tags + Character + Word Embedding	70%
Template 1 + Template 2 + Template 3 (480 training records / 120 test records)	Dependency Tree + Word Embedding	71.6%
	Dependency Tree + Parts-of-speech + Word embedding	72.5%
	Dependency Tree + Parts-of-speech + Relation tags + Character + Word Embedding	72.83%

Analysis and Results

Parameter Tuning

- Small dataset size of 600 questions
- Prevent overfitting with aggressive regularization and curtail learning rate
- Weight decay
  - Update weights every epoch with multiplicative factor less than 1
  - Prevent exploding gradient
- Dropout
  - Drop random units with their connections to prevent overfitting
- Adaptive learning rate
  - Accuracy stagnant after few epochs ; training loss increasing
  - Step scheduler periodically decreased learning rate

Analysis and Results

Parameter Tuning

- Final parameters of the model

Parameter	Value
Input dimensions	444 x 1
Tree-LSTM memory dimensions	150 x 1
Epochs	20
Batch size	25
Learning rate	1×10^{-1}
Weight decay	0.1×10^{-3}
Dropout	0.2
Loss function	Cross entropy loss
Optimizer	Adam optimizer
Scheduler	Stepwise learning rate decay
LR step size	once every 5 epochs
LR step decay	0.1

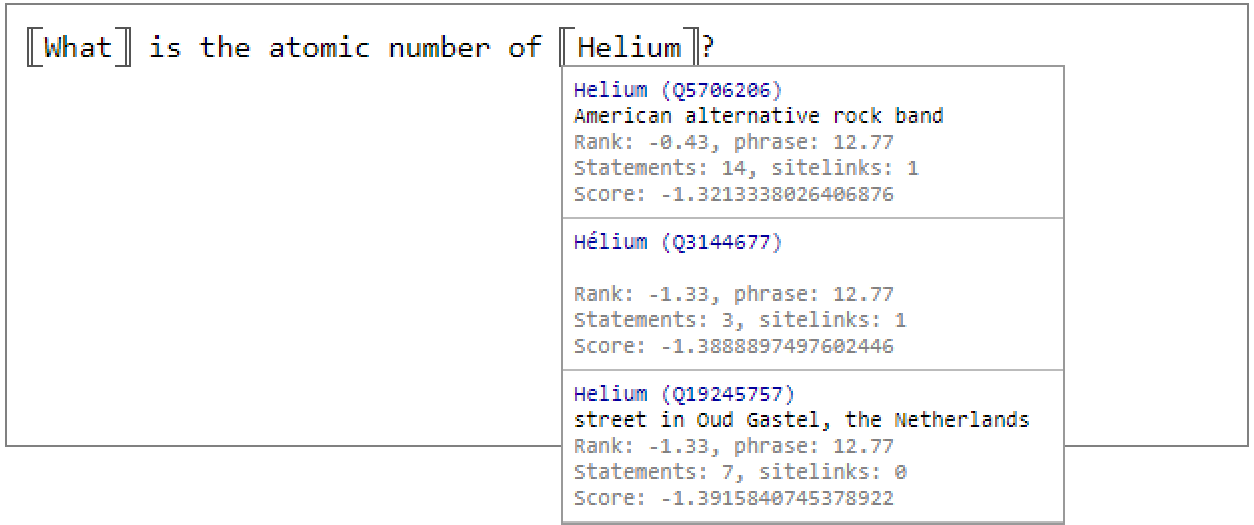
Analysis and Results

Phrase Matching

- Alternate entity linking library : OpenTapioca
- Works better than Falcon 2.0 for person, organization and location
- Falcon 2.0 is a joint entity and relation linking module

Example: What is the atomic number of Helium?


```
{
  "entities_wikidata": [
    [
      "<http://www.wikidata.org/entity/Q560>",
      "Helium"
    ]
  ],
  "relations_wikidata": [
    [
      "<http://www.wikidata.org/entity/P1086>",
      "atomic number"
    ]
  ]
}
```

Analysis and Results

Query Construction

- For a test dataset of 120 questions in English language, the highest accuracy achieved by the Tree-LSTM model was 72.83%. This model identifies the SPARQL query for the input question.
- The Falcon 2.0 API results the list of entities and relations that are combined with the classification results.
- The resultant queries are checked against manually generated queries. Queries are correct if they give the desired answer or are meaningfully correct.
- From the 120 questions, 60% queries were constructed correctly.

Analysis and Results

Query Construction

- Example Result

```
What is the total equity of Micron Technology?
['SELECT DISTINCT ?answer WHERE { wd:Q1197548 wdt:P2137
?answer}']
```

Micron Technology (Q1197548)

American multinational corporation based in Boise, Idaho which produce,
many forms of semiconductor devices.
Micron Technology, Inc.

total equity (P2137)

amount of equity value for an entity
equity | shareholder equity

Analysis and Results

Query Construction

- Example Result

```
What is Sanskrit's writing system?
['SELECT DISTINCT ?answer WHERE { wd:Q11059 wdt:P282 ?answer}',
'SELECT DISTINCT ?answer WHERE { wd:Q58778 wdt:P282 ?answer}']
```

Sanskrit (Q11059)

ancient Indian language
sa | Sunscrit | skt

writing system (P282)

alphabet, character set or other system of writing used by a language, supported by a typeface
alphabet | script

system (Q58778)

set of interacting or interdependent components

Analysis and Results

Query Construction

- Types of errors possible:
  - Misclassification of template
  - Incorrect entity and relation linking
  - Multiple triple candidates
  - Incorrect grammar of input user question

Conclusion

- Resultant query can be executed on Wikidata query service to get the desired answer
- The system has a correctness of 60% across 3 unique SPARQL templates
- Lack of ontology recognition. Can be improved with custom entity and relation linking module
- Extend training dataset for a larger template coverage as well as the number of questions under each template