universität freiburg

Porting ChatKBQA to Wikidata

Bachelorarbeit

Chair of Algorithms and Data Structures
Luis Drayer
2. Oktober 2026

Problem - KGQA

- Provide grounded answers to arbitrary natural language questions:
  - "Who is the brother of Justin Bieber?"
  - "Which countries have produced actors who have won an Academy Award?"
- Retrieving a correct answer is not trivial

→ Rely on structured representation of knowledge like RDF

What is ChatKBQA?

- KGQA-framework: Given a natural language question generates a SPARQL query

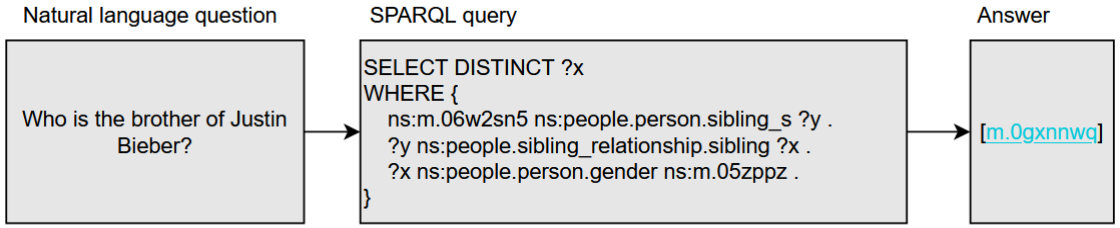

Figure 1: ChatKBQA input and output example

- Utilizes LLMs
- Can be decomposed into 3 steps
  - Training format conversion
  - Model finetuning and generation
  - Unsupervised Retrieval

Why ChatKBQA?

- State-of-the-art results on WebQuestionsSP (WebQSP) and ComplexWebQuestions (CWQ)
- Exclusive to these two benchmarks
- Exclusive to Freebase

→ How well does the method transfer to other Knowledge Graphs?

Training Format Conversion - Original

- Uses own training format based on logical chain
- Makes intermediate variables obsolete
- Replace entity identifiers with respective label

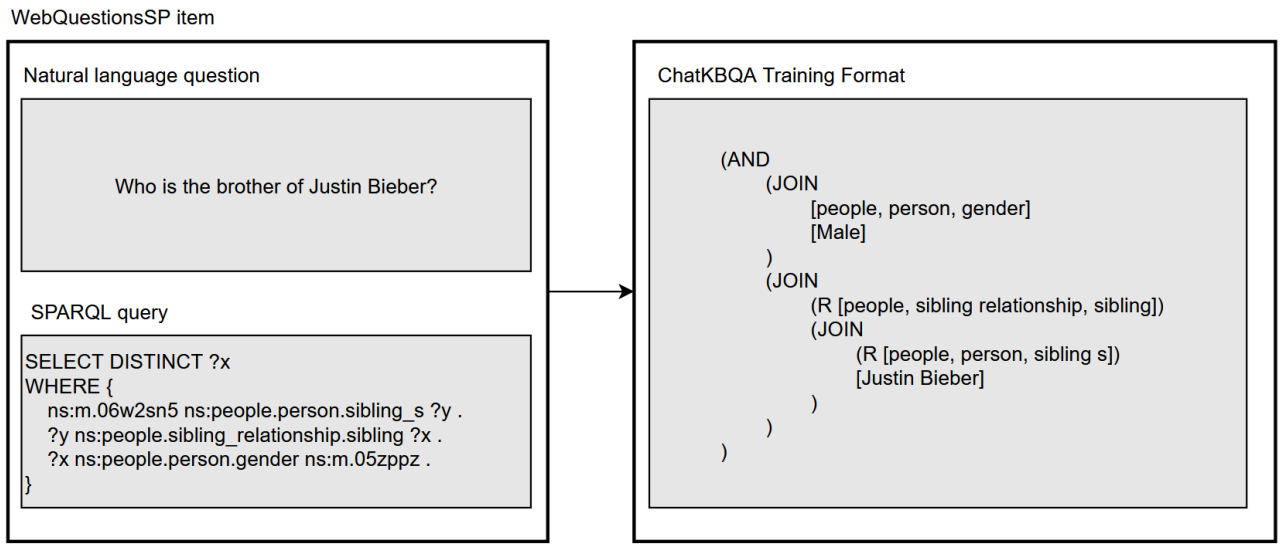

Figure 2: ChatKBQA Training Format example

→ Reduce complexity for LLM generation

Training Format Conversion - Limitations

- Limited expressiveness
  - Fixed set of supported operators
  - Cannot express multi-variable SELECT clause
- Requires complete parser from and to arbitrary SPARQL
  - ChatKBQA includes separate parsers for WebQSP and CWQ, but no general parser

→ Does not work for arbitrary benchmark queries

Training Format Conversion – Extension

- Generate two alternative training formats using Apache Jena
- Preserve full expressiveness of SPARQL

Jena Syntax-Expression

```
(distinct
  (project (?x)
    (filter
      (exprlist
        (!= ?x fb:Justin_Bieber)
        (||
          (|| (! (isLiteral ?x)) (= (lang ?x) ""))
          (langMatches (lang ?x) "en")
        )
      )
      (bgp
        (triple fb:Justin_Bieber fbp:people.person.sibling_s ?y)
        (triple ?y fbp:people.sibling_relationship.sibling ?x)
        (triple ?x fbp:people.person.gender fb:Male)
      )
    )
  )
)
```

Normalized SPARQL

```
SELECT DISTINCT ?x
WHERE {
   fb:Justin_Bieber fbp:people.person.sibling_s ?y .
   ?y fbp:people.sibling_relationship.sibling ?x .
   ?x fbp:people.person.gender fb:Male .
   FILTER (?x != fb:Justin_Bieber)
   FILTER (((!isLiteral(?x)) || (lang(?x) = "")) || langMatches(lang(?x), "en"))
}
```

Figure 3: Extension Training Formats using Apache Jena

Finetuning and Generation

- Original used LLMTuner for finetuning and beam search inference
- Extension uses Llamafactory, the successor to LLMTuner

→ No significant methodical change

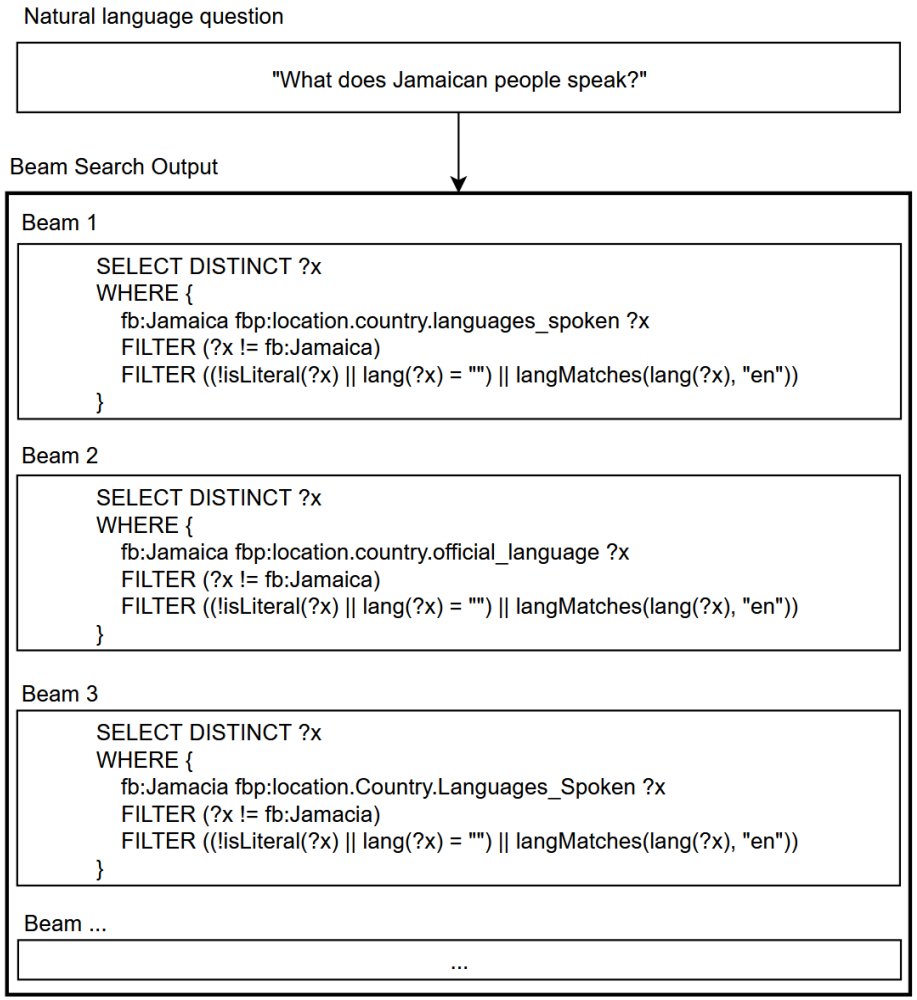

Figure 4: Sample beam search inference output

Retrieval of Entities and Predicates - Original

- Centerpiece of ChatKBQA
- Iterate over prediction beams in 2 separate passes
- Combinatorial exploration of SPARQL query candidates
- First executable query is considered final

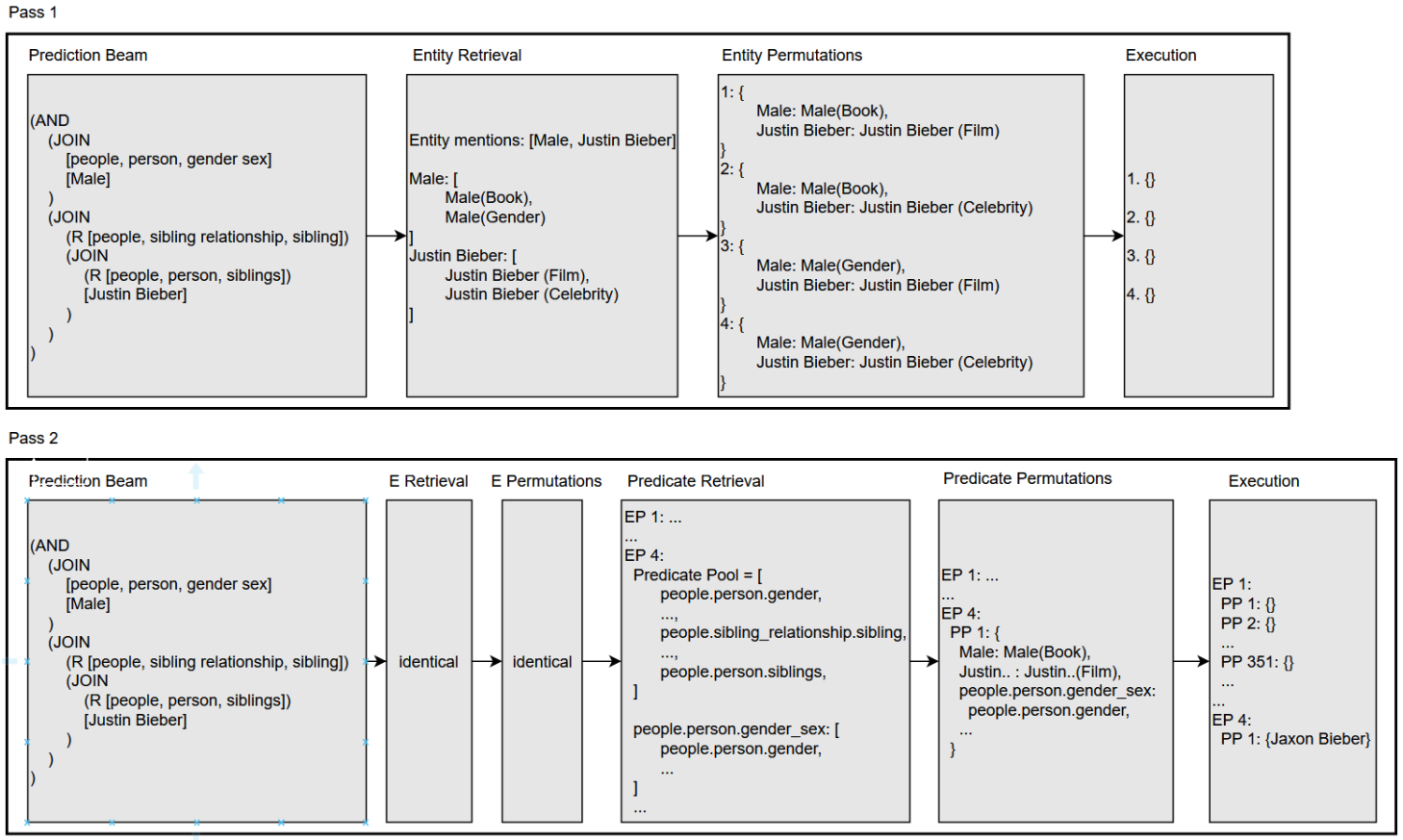

Figure 5: Simple ChatKBQA retrieval example

Retrieval – Entity Retrieval

1. Type Map lookup
2. FACC1 lookup

Retrieval – Predicate Retrieval

- Implicit in pass 1
- Explicit 2-Hop neighborhood expansion in pass 2
  - Based on the current entity permutation's entity assigment

$$v_2 \xrightarrow{p_1} v_1 \xrightarrow{p_0} e \qquad v_2 \xleftarrow{p_1} v_1 \xrightarrow{p_0} e \qquad e \xrightarrow{p_0} v_1 \xleftarrow{p_1} v_2 \qquad e \xrightarrow{p_0} v_1 \xrightarrow{p_1} v_2$$

Figure 6: Retrieval neighborhood expansion schema

Retrieval – Limitations and Extensions

- Rigid hyperparameter configuration
  - ➢ Fully customizeable YAML run config
- Entity and Predicate linking behavior rigid and embedded in algorithm
  - ➢ Extract behavior into predefined entity and predicate linkers
  - ➢ Retrieval algorithm takes list of entity and predicate linkers
- Freebase-specific implementation
  - ➢ Extract Knowledge-Graph-specific operations into Knowledge-Graph-Module

Retrieval – Wikidata Entity Linkers

1. Type Map lookup
    - Knowledge-graph-agnostic
2. FACC1 lookup
    - Freebase specific index

→ Use an entity linker based on Wikibase's searchentities API instead of FACC1

Retrieval – Wikidata Predicate Linkers

1. Implicit (Simple) Linker
    - Only works for Freebase, because Wikidata predicate identifiers do not carry semantic meaning
2. Neighborhood Linker
    - Works on Wikidata with some small adaptations

→ Use a predicate linker based on simple label search via SPARQL query instead of implicitly returning the input predicate mention.

Results - Metrics

- Exact Match (EM): 1 if the result rows are identical (independant of order), 0 otherwise
- Assignment F1 (F1):
  - Find overlap of gold to predicted rows.
  - Missing columns are penalized, additional columns are tolerated.
- Hit@1: 1 if F1 > 0, otherwise 0

?x	?label
Q40	'Austria'
Q183	'Germany'

?x
Q40
Q183
Q39

Table 1: Example result rows

Results - Experimental Setup

Resource	Allocation
CPU	1 CPU core per job
Memory	130–200 GB for Freebase; 32 GB for Wikidata
GPU	NVIDIA L40S for SimCSE-based experiments, fine-tuning and inference
Cluster	University computing cluster
Knowledge base	Locally hosted Virtuoso for Freebase; externally hosted QLever instance for Wikidata

Table 2: Experimental Setup

Results - Validating Retrieval Performance

Run	EM	F1	Hit@1	Note
ChatKBQA CWQ	73.3	77.8	82.7	Official metrics reported in paper
Reproduction CWQ	74.3	78.3	82.7	Adapted original predictions
ChatKBQA CWQ Gold	76.8	81.3	86.0	Official metrics reported in the paper, using gold entities
Reproduction CWQ Gold	78.3	82.2	86.3	Adapted original predictions, using gold entities
ChatKBQA WebQSP	73.8	79.8	83.2	Official metrics reported in paper
Reproduction WebQSP	73.3	77.9	82.4	Adapted original predictions
ChatKBQA WebQSP Gold	77.8	83.5	86.4	Official metrics reported in paper, using gold entities
Reproduction WebQSP Gold	76.7	81.1	85.3	Adapted original predictions, using gold entities

Table 3: Comparison of the reproduced ChatKBQA results with the reported results

Results - Retraining Experiments

Run	EM	F1	Hit@1	time	Note
Reproduction CWQ	71.7	75.8	80.0	15.75s	Adapted original predictions
Llama-2-13b_CWQ-sparql	72.8	76.5	80.5	18.28s	SPARQL retrain
Reproduction CWQ Gold	78.3	82.2	86.3	4.59s	Adapted original predictions, using gold entities
Llama-2-13b_CWQ-sparql Gold	80.5	83.8	87.5	4.33s	SPARQL retrain, using gold entities
Reproduction WebQSP	73.0	77.5	82.0	18.00s	Adapted original predictions
Llama-2-7b_WebQSP-sparql	72.0	76.3	80.5	15.56s	SPARQL retrain
Reproduction WebQSP Gold	76.5	81.0	85.3	14.19s	Adapted original predictions, using gold entities
Llama-2-7b_WebQSP-sparql Gold	77.0	81.0	85.3	5.95s	SPARQL retrain, using gold entities

Table 4: Comparison of adapted prediction results with the retrain results

Results – Comparing Training Formats

Run	Exec %	F1	time	Note
Llama-2-13b_CWQ-sparql	94.0%	76.5	18.28 s	SPARQL retrain
CWQ sparql	85.3%	74.4	5.55 s	Reduced config
CWQ jena	84.7%	74.0	8.75 s	Reduced config
Llama-2-13b_CWQ-sparql Gold	96.3%	83.8	4.33 s	SPARQL retrain
CWQ sparql Gold	92.4%	82.2	2.31 s	Reduced config
CWQ jena Gold	91.9%	81.8	2.58 s	Reduced config
Llama-2-7b_WebQSP-sparql	97.9%	76.3	15.56 s	SPARQL retrain
WebQSP sparql	97.5%	76.1	1.76 s	Reduced config
WebQSP jena	95.8%	74.5	2.70 s	Reduced config
Llama-2-7b_WebQSP-sparql Gold	99.1%	81.0	5.95 s	SPARQL retrain
WebQSP sparql Gold	97.8%	80.3	1.02 s	Reduced config
WebQSP jena Gold	96.9%	78.9	1.70 s	Reduced config

Table 5: Comparison of training format performance impact under otherwise equal conditions. Retrain experiments are shown for reference values.

Results – Wikidata Experiments

Run	Exec %	EM	F1	Hit@1	time
QALD-7 sparql	90.0%	30.4	35.0	45.6	47.31 s
QALD-10 sparql	91.9%	33.9	34.9	36.0	46.94 s
WWQ sparql	97.7%	66.5	72.9	79.6	5.28 s
LC-QuAD 2.0 sparql	81.0%	59.7	62.0	63.7	30.46 s

Table 6: Performance of ChatKBQA extension on various Wikidata benchmarks

Conclusions and Contribution

- Transfer the ChatKBQA method to Wikidata
- Establish first baseline values for Wikidata
- Transform ChatKBQA into a flexible framework for different linking behavior
- Enable compatibility for any knowledge base

Extras – Hyperparameter Setup

		Simple		Neighborhood	
Run	$\mathbf{k}_b$	k_e	k_p	k_e	k_p
CWQ	8	500	1	30	300
WebQSP	15	500	1	50	4000

Table 7: Hyperparamter configuration derived from original ChatKBQA implementation

	Pass 1		Pass 2	
$\mathbf{k}_b$	k_e	k_p	k_e	k_p
8	100	5	5	15

Table 8: Reduced hyperparamter configuration

Extras – Label Fallback

```
PREFIX ns: http://rdf.freebase.com/ns/
PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema

SELECT DISTINCT ?x
WHERE {
  FILTER (!isLiteral(?x) || lang(?x) = "" || langMatches(lang(?x), "en")
     )
  ns:m.05xyz12 ns:people.person.place_of_birth ?x .
}
```

```
PREFIX ns: http://rdf.freebase.com/ns/
PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema

SELECT DISTINCT ?x
WHERE {
  FILTER (!isLiteral(?x) || lang(?x) = "" || langMatches(lang(?x), "en")
     )

  ns:m.05xyz12 rdfs:label ?en0 .
  ?ei0 rdfs:label ?en0 .
  FILTER(langMatches(lang(?en0), "EN"))

  ?ei0 ns:people.person.place_of_birth ?x .
}
```

Figure 7: Label fallback example

Extras – WebQSP Evaluation


```
{
  "QuestionId": "WebQTest-0",
  "RawQuestion": "what does jamaican people speak?",
  "ProcessedQuestion": "what does jamaican people speak",
  "Parses": [
```

```
{
  "ParseId": "WebQTest-0.P0",
  "AnnotatorId": 0,
  "AnnotatorComment": {
    "ParseQuality": "Complete",
    "QuestionQuality": "Good",
    "Confidence": "Normal",
    "FreeFormComment": "First-round parse verification"
  },
  "Sparql": "PREFIX ns: <http://rdf.freebase.com/ns/>\nS
  "PotentialTopicEntityMention": "jamaican",
  "TopicEntityName": "Jamaica",
  "TopicEntityMid": "m.03_r3",
  "InferentialChain": [
    "location.country.languages_spoken"
  ],
  "Constraints": [],
  "Time": null,
  "Order": null,
  "Answers": [
    {
      "AnswerType": "Entity",
      "AnswerArgument": "m.01428y",
      "EntityName": "Jamaican English"
    },
    {
      "AnswerType": "Entity",
      "AnswerArgument": "m.04ygk0",
      "EntityName": "Jamaican Creole English Language"
    }
  ]
},
```

```
{
  "ParseId": "WebQTest-0.P1",
  "AnnotatorId": 0,
  "AnnotatorComment": {
    "ParseQuality": "Complete",
    "QuestionQuality": "Good",
    "Confidence": "Normal",
    "FreeFormComment": "First-round parse verification"
  },
  "Sparql": "PREFIX ns: <http://rdf.freebase.com/ns/>\nS
  "PotentialTopicEntityMention": "jamaican",
  "TopicEntityName": "Jamaica",
  "TopicEntityMid": "m.03_r3",
  "InferentialChain": [
    "location.country.official_language"
  ],
  "Constraints": [],
  "Time": null,
  "Order": null,
  "Answers": [
    {
      "AnswerType": "Entity",
      "AnswerArgument": "m.01428y",
      "EntityName": "Jamaican English"
    }
  ]
}
```


Figure 8: Example WebQSP item