SPINACH: SPARQL-Based Information Navigation for Challenging Real-World Questions

Shicheng Liu[†*] **Sina J. Semnani**[†*] **Harold Triedman**[‡§] **Jialiang Xu**[†]
Isaac Dan Zhao[†] **Monica S. Lam**[†]
[†] Stanford University [‡] Cornell Tech
`{shicheng, sinaj, xjl, ikezhao, lam}@cs.stanford.edu`
`hjt36@cornell.edu`

Abstract

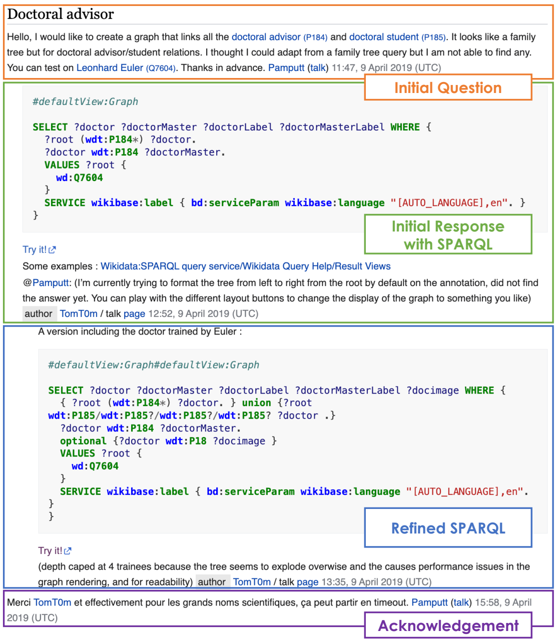
Large Language Models (LLMs) have led to significant improvements in the Knowledge Base Question Answering (KBQA) task. However, datasets used in KBQA studies do not capture the true complexity of KBQA tasks. They either have simple questions, use synthetically generated logical forms, or are based on small knowledge base (KB) schemas.

We introduce the SPINACH dataset, an expert-annotated KBQA dataset collected from discussions on Wikidata's "Request a Query" forum with 320 decontextualized question-SPARQL pairs. The complexity of these in-the-wild queries calls for a KBQA system that can dynamically *explore* large and often incomplete schemas and *reason* about them, as it is infeasible to create a comprehensive training dataset.

We also introduce an in-context learning KBQA agent, also called SPINACH, that mimics how a human expert would write SPARQLs to handle challenging questions. SPINACH achieves a new state of the art on the QALD-7, QALD-9 Plus and QALD-10 datasets by 31.0%, 27.0%, and 10.0% in F_1, respectively, and coming within 1.6% of the fine-tuned LLaMA SOTA model on WikiWebQuestions. On our new SPINACH dataset, the SPINACH agent outperforms all baselines, including the best GPT-4-based KBQA agent, by at least 38.1% in F_1.[1]

1 Introduction

Wikidata (Vrandečić and Krötzsch, 2014), one of the largest publicly available knowledge bases, currently contains 15 billion facts and is estimated to grow at a rate of 1 billion triples per year (MediaWiki, 2024b). It is of significant value to many scientific communities, including Mathematics (Scharpf et al., 2021), Biology (Mitraka et al., 2015; Pfundner et al., 2015), Education (Sigalov and Nachmias, 2023), Linguistics (Turki et al., 2017; lu Yu and Qiao, 2017), and the Social Sciences (Kapsalis, 2019; Zhao, 2022), among many others (Farda-Sarbas and Müller-Birn, 2019; Turki



Doctoral advisor

Hello, I would like to create a graph that links all the doctoral advisor (P184) and doctoral student (P185). It looks like a family tree but for doctoral advisor/student relations. I thought I could adapt from a family tree query but I am not able to find any. You can test on Leonhard Euler (Q7604). Thanks in advance. Pamputt (talk) 11:47, 9 April 2019 (UTC)

```
#defaultView:Graph

SELECT ?doctor ?doctorMaster ?doctorLabel ?doctorMasterLabel WHERE {
  ?root (wdt:P184*) ?doctor.
  ?doctor wdt:P184 ?doctorMaster.
  VALUES ?root {
    wd:Q7604
  }
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en". }
}
```

Some examples : Wikidata:SPARQL query service/Wikidata Query Help/Result Views

@Pamputt: (I'm currently trying to format the tree from left to right from the root by default on the annotation, did not find the answer yet. You can play with the different layout buttons to change the display of the graph to something you like) author TomT0m / talk page 12:52, 9 April 2019 (UTC)

A version including the doctor trained by Euler :

```
#defaultView:Graph#defaultView:Graph

SELECT ?doctor ?doctorMaster ?doctorLabel ?doctorMasterLabel ?docimage WHERE {
  { ?root (wdt:P184*) ?doctor. } union {?root
wdt:P185/wdt:P185?/wdt:P185?/wdt:P185? ?doctor .}
  ?doctor wdt:P184 ?doctorMaster.
  optional {?doctor wdt:P18 ?docimage }
  VALUES ?root {
    wd:Q7604
  }
  SERVICE wikibase:label { bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en".
}
}
```

(depth caped at 4 trainees because the tree seems to explode overwise and the causes performance issues in the graph rendering, and for readability) author TomT0m / talk page 13:35, 9 April 2019 (UTC)

Merci TomT0m et effectivement pour les grands noms scientifiques, ça peut partir en timeout. Pamputt (talk) 15:58, 9 April 2019 (UTC)

Figure 1(a): An example forum discussion

Question: Who are the doctoral advisors of Leonhard Euler,
and their advisors, and so on? In addition, who are his doctoral student,
grand-student, great-grand-student, and great-great-grand-students?
Each tuple in the result should contain both the student and the doctoral advisor.

SPARQL: **SELECT** ?doctor ?doctorMaster **WHERE** {
  { ?root (wdt:P184*) ?doctor. }
  UNION
  { ?root (wdt:P185/(wdt:P185?)/(wdt:P185?)/(wdt:P185?)) ?doctor. }
  ?doctor wdt:P184 ?doctorMaster.
  VALUES ?root {
    wd:Q7604
  }
}

Figure 1(b): The corresponding decontextualized example included in the validation set of the SPINACH dataset. The SPARQL is based on the "Refined SPARQL" proposed by the user `TomT0m`. Projection fields asking for labels and the optional image clause are removed in accordance with our methodology in Section 3.1.

---

[*] Equal contribution

[§] Work conducted while at the Wikimedia Foundation

[1] Code and data available at `https://github.com/stanford-oval/spinach`. The SPINACH agent is publicly available at `https://spinach.genie.stanford.edu/` and on Wikidata as SpinachBot.

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 15977–16001
November 12-16, 2024 ©2024 Association for Computational Linguistics

et al., 2023). Effective access to Wikidata data can be challenging. To address this, the Wikidata Request a Query forum[2] has been created so that users can ask questions, propose solutions, and participate in follow-up conversations. Figure 1(a) shows one such conversation, in which a user wants to obtain the academic genealogy of Leonhard Euler. Note that in SPARQL, entities and properties are uniquely identified by QIDs and PIDs, respectively. Here P184 and P185 represent the "doctoral advisor" and "doctoral student", respectively, and Q7604 is Leonhard Euler. Readers are referred to Wikidata (2024) for more information.

Although numerous datasets have been proposed for Knowledge Base Question Answering (KBQA) task, they either contain only simple questions (Yih et al., 2016; Bordes et al., 2015; Usbeck et al., 2017; Perevalov et al., 2022; Usbeck et al., 2023; Xu et al., 2023; Korablinov and Braslavski, 2020; Rybin et al., 2021) or synthetically generated complex logical forms (Bao et al., 2016; Gu et al., 2021; Talmor and Berant, 2018; Keysers et al., 2020; Cao et al., 2022a; Dubey et al., 2019). Datasets with synthetically generated logical forms often lead to an overestimate of the performance of KBQA systems; performing well on them does not translate to real-world queries (Oren et al., 2021; Campagna et al., 2022). Thus, the community needs a high-quality dataset with organic, real-world queries that capture the true complexity of KBQA tasks.

In this paper, we propose using the Wikidata Request a Query forum to build and evaluate next-generation KBQA systems. **We introduce the 🥬 Spinach dataset, a new, expert-annotated KBQA dataset featuring decontextualized question-SPARQL pairs derived from complex discussions on a real-world forum.** Figure 1(b) is the example obtained from the discussion in Figure 1(a).

As we show in this paper, current KBQA approaches (Xu et al., 2023; Sun et al., 2024a) cannot handle the complexity of these real-world queries. The state-of-the-art (SOTA) approach, ToG (Sun et al., 2024a), which integrates LLMs with KG reasoning, scores only 1.8 EM and 7.2 F1 on this data set. Therefore, **we propose a new LLM-augmented KBQA approach, 🥬 Spinach**: **SP**ARQL-Based **I**nformation **Na**vigation for **Ch**allenging Real-World Questions, which is designed with the primary goal of mimicking how an expert would tackle the task. We show that Spinach establishes a new state of the art on popular datasets such as QALD-7, QALD-9, and QALD-10 and comes within 1.6% of the fine-tuned SOTA on WikiWebQuestions. On the Spinach dataset, our agent outperforms all baselines, including the best GPT-4-based KBQA agent, by at least 38.1% in F1.

2 Related Work

2.1 KBQA Benchmarks

Over the past decade, numerous Knowledge Base Question Answering (KBQA) benchmarks with logical forms have been introduced and can be classified into two categories:

Datasets with natural questions originally collected through search engines or crowdsourcing. This includes the popular dataset WebQuestionSP (Yih et al., 2016), along with the QALD datasets (Usbeck et al., 2017, 2018, 2023; Perevalov et al., 2022), RuBQ (Korablinov and Braslavski, 2020), and SimpleQuestions (Bordes et al., 2015), among others;

Datasets with synthetically generated logical forms and questions, where some datasets paraphrase synthetically generated questions via crowdsourcing. This category includes ComplexWebQuestions (Talmor and Berant, 2018), GrailQA (Gu et al., 2021), KQA Pro (Cao et al., 2022a), CFQ (Keysers et al., 2020), and LC-QuAD-2 (Dubey et al., 2019), among others.

Some of the aforementioned datasets were originally based on Freebase (Bollacker et al., 2008) or DBPedia (Lehmann et al., 2015). Recently, however, the KBQA community has shifted toward using Wikidata as the underlying knowledge base for KBQA datasets, due to its larger size and continuous updates from community contributions.[3] Several datasets have been converted to use Wikidata: WebQuestionSP has been converted to WikiWebQuestions (Xu et al., 2023); SimpleQuestions has a Wikidata version (Diefenbach et al., 2017b); and CFQ has been converted to MCWQ (Cui et al., 2022).

In Table 1, we present quantitative statistics on these datasets. Existing datasets with natural questions typically involve relatively simple queries (e.g., 2.63 clauses per query for WikiWeb-

---

[2] `https://www.wikidata.org/wiki/Wikidata:Request_a_query`

[3] For more on why Wikidata is better suited for KBQA benchmarks, refer to Section 1.1 of Usbeck et al. (2023).

	Avg. Clauses	Avg. Projs	Avg. Rels	Avg. Subjs	Avg. Preds	Avg. Objs	Avg. Lits
Natural questions w/ annotated logical forms							
WikiWebQuestion (Xu et al., 2023)	2.63	1	1.53	1.25	1.52	1.53	0.04
QALD-9 Plus (Perevalov et al., 2022)	3.14	1	1.77	1.26	1.70	1.78	0.05
QALD-10 (Usbeck et al., 2023)	2.38	1	1.27	1.19	1.17	1.32	0.05
RuBQ (Korablinov and Braslavski, 2020)	2.17	1	1.12	1.03	1.11	1.07	0.01
SimpleQuestionsWikidata (Diefenbach et al., 2017b)	2.00	1	1.00	1.00	1.00	1.00	0.00
Synthetic logical forms w/ synthetic or paraphrased questions							
CWQ (Talmor and Berant, 2018)	5.19	1	2.80	1.87	2.62	3.38	0.11
GrailQA (Gu et al., 2021)	7.10	1	3.02	1.97	2.43	3.90	0.08
KQA Pro (Cao et al., 2022a)	6.34	1	5.01	2.77	3.94	2.43	2.37
MCWQ (Cui et al., 2022)	6.34	1	5.09	2.67	3.53	3.37	0.00
LC-QuAD-2 (Dubey et al., 2019)	3.65	1	2.07	1.51	2.05	2.07	0.22
Natural logical form w/ annotated questions							
🥬 SPINACH **(Ours)**	8.89	2.50	4.03	1.76	3.55	4.53	0.46

Table 1: Quantitative comparison of KBQA datasets on average number of Clauses, Projections, Relations, Subjects, Predicates, Objects, and Literals. For datasets originally based on Freebase, we calculate their corresponding datasets in Wikidata, if available. Refer to Appendix A for definitions of each metric.

Questions). In contrast, synthetically generated datasets generally contain more complex queries by design. However, due to the limited natural language variety between training and evaluation data, models can achieve artificially high accuracy. For instance, a simple semantic parser based on the BART (Lewis et al., 2020) model can achieve an accuracy of over 90% on KQA-Pro even without doing entity linking (Cao et al., 2022a). Furthermore, systems that excel at synthetic datasets do not necessarily perform well in natural, complex tasks (Oren et al., 2021; Campagna et al., 2022). Additional details on the number of properties and unique query patterns of existing datasets and SPINACH are presented in Appendix B. This shows the need for a high-quality dataset with both natural questions and natural complex logical forms in the KBQA community.

2.2 KBQA Approaches

Current KBQA systems can be classified into three categories: (1) subgraph retrieval using vector embeddings (Sun et al., 2018, 2019; Sen et al., 2021; Verga et al., 2021; Mavromatis and Karypis, 2022), (2) semantic parsing (Yih et al., 2015, 2016; Luo et al., 2018; Lan and Jiang, 2020; Das et al., 2021; Ye et al., 2022; Cao et al., 2022b; Gu and Su, 2022; Xu et al., 2023), and more recently, (3) LLM-based graph exploration (Sun et al., 2024a; Xiong et al., 2024). Some works use a combination of these techniques (Yu et al., 2023; Luo et al., 2024a,b).

Due to the sheer size of Wikidata, embedding its entire graph is prohibitively expensive. Subgraph retrieval approaches therefore limit the problem to a small fixed subset of Wikidata. Xiong et al. (2024), for instance, create and search through embeddings of the graph patterns (triples) for relevant patterns and conduct experiments on a tiny subgraph covering only 0.01% of Wikidata (i.e. containing 17K QIDs instead of the full 111,568K). This limitation renders them inapplicable to real-world scenarios as it leaves much of the knowledge graph unaccessible to users. Among semantic parsing systems, Xu et al. (2023) fine-tune LLaMA (Touvron et al., 2023) with a modified SPARQL syntax and achieve state-of-the-art results on 2 KBQA benchmarks. Among LLM-based approaches, Sun et al. (2024a) instruct an LLM to dynamically explore the graph to fetch answers, achieving SOTA on 6 KBQA datasets. In Section 5, we evaluate these two systems on our new SPINACH dataset.

3 The SPINACH Dataset

The archives of the Wikidata Request a Query from July 2016 to the present date are available[4]. These conversations are real and organic, reflecting the kind of queries that practitioners are interested in. Additional details on the forum and analysis of the conversations can be found in Appendix C.1.

Out of the discussions up to May 2024, 2780 discussions include at least one valid SPARQL query. We keep conversations whose last-mentioned SPARQL query returns non-empty results under 10 MB. Of the 2171 discussions left, we randomly sample 920 conversations spanning many domains for consideration. These conversations typically do not specify the exact natural language corresponding to the SPARQLs. Thus, manual processing is

---

[4] `https://www.wikidata.org/wiki/Wikidata:Request_a_query/Archive`

required to convert them to a KBQA dataset. Three Wikidata experts among the authors of this paper manually inspected these conversations to produce a dataset with 155 examples in the validation and 165 examples in the test set, as described below.

Additional discussions on the size and statistical power of SPINACH dataset can be found in Appendix C.2. More information on the process of converting source conversations and SPARQLs into a KBQA dataset can be found in Appendix C.3.

3.1 Choosing and Trimming the Queries

We design the dataset to focus on end-users rather than Wikipedia and Wikidata contributors interested in obscure optimizations or formatting. As such, we manually remove the following SPARQL clauses:

Wikimedia presentation queries: We filter out clauses for analyzing or improving Wikimedia projects themselves, such as asking which Wikipedia articles exist in one language but not another.

Questions on complex SPARQL code. We exclude conversations where users ask for help with debugging overly complicated SPARQL queries, when their meaning is already difficult to accurately convey in natural language.

Queries obscured by optimizations: Because Wikidata restricts query runtime to 60 seconds (MediaWiki, 2024a), users may include clauses to optimize the performance, which are often not directly relevant to answering the question. We remove these clauses.

Formatting clauses: We remove clauses that only format the results.

Appendix C.4 and C.5 show specific examples of these modifications and exclusions.

3.2 Annotating Natural Questions

For each conversation, the experts annotate a self-contained, decontextualized natural language question that *accurately* captures the meaning of the user-written SPARQL. We follow these steps:

Disambiguate entities and properties: Entities and properties are important parts of a query. For ambiguous entities, the verbalization includes additional information to resolve the ambiguity. As for the properties, the verbalization should distinguish between similar properties. For example, instead of asking “where a movie takes place”, we distinguish between the “narrative location” and the “filming location”; instead of asking “where a person comes from”, we distinguish between the “country of citizenship” vs. “country of birth”.

Natural verbalizations: Whenever possible, to the extent that no ambiguities are introduced, the question should refrain from directly using entity and property names, instead using a more natural way to express the meaning. For instance, instead of asking “what is the point of time of the goal?”, a more natural question with the same level of accuracy like “when does the goal take place?” should be used.

Accurately capturing optional clauses and projections: In SPARQL, `OPTIONAL` clauses are used to include additional data that may or may not exist for queried items without excluding results lacking this optional data.[5] The verbalization should specify which fields are optional with clauses such as “if available”. Similarly, the verbalization should accurately capture what is requested in the projections, using phrases like: “For each result, return their name and location” if necessary.

As illustrated in Table 1, the SPINACH dataset is the first dataset with both natural and complex logical forms, which represents the real-world KBQA needs of Wikidata users. Relative to previous datasets, there are more unique properties and unique logical forms per query. Additional details on the popularity of the query topics in the SPINACH dataset can be found in Appendix C.6.

4 The SPINACH Agent

The SPINACH agent is an LLM-augmented knowledge graph exploration agent, where an LLM is instructed to explore the knowledge graph until an appropriate answer is found or another stop condition is met. However, unlike prior work, we design SPINACH with the primary goal of mimicking a human expert writing a SPARQL query. An expert starts by writing simple queries and looking up Wikidata entity or property pages when needed, all to understand the structure of the knowledge graph and what connections exist. This is especially important for Wikidata due to its anomalous structure (Shenoy et al., 2022). An expert then might add new SPARQL clauses to build towards the final SPARQL, checking their work along the way by executing intermediate queries and eyeballing the results. Potentially, they might go back to the drawing board and double-check their assumptions

---

[5] `https://www.wikidata.org/wiki/Wikidata:SPARQL_tutorial#OPTIONAL`

if a query fails.

Following this design principle, SPINACH improves upon the following aspects of previous agent-based KBQA systems:

First, SPINACH uses the full expressiveness of SPARQL for exploration. This contrasts with Sun et al. (2024a); Xiong et al. (2024); Sun et al. (2024b), which explore the knowledge graph one edge at a time. That is, SPINACH weaves together the exploration and the writing of SPARQL. It is instructed to *try many SPARQLs and fail early*. It learns about the structure of the knowledge graph from the results of its queries (or lack thereof), or realizes its incorrect assumptions, and decides to revisit them. This is a key difference from Sun et al. (2024b,a), where the answer is generated only when exploration is done.

Second, during its exploration, SPINACH does not keep track of a subgraph as its state. Instead, we define the state to be the full history of actions and their results so far. Limiting the state to a subgraph (Sun et al., 2024b,a) means that the agent would categorically not support questions whose answer contains a large number of entities (e.g. "Which actors graduated from Harvard?") or computation (e.g. "What is the tallest mountain?").

Third, while most KBQA agents start their exploration from entities they detect from the question, SPINACH does not assume access to entities. This, for example, gives the agent the flexibility to start by searching for relevant relations if that is a more suitable way to explore.

Concretely, SPINACH agent runs for N time steps. At time step n, it takes the current *state* as input and outputs a *thought* (Yao et al., 2023) t_n and one of the possible actions a_n from the set of all possible actions. The action is then executed outside of the agent, and the results are returned to the agent as observation o_n. In the rest of this section, we go into the details of the SPINACH agent. Figure 2 shows an example of how SPINACH answers a question.

4.1 Exploration State

The exploration state at time step n is $\mathcal{S}_n = \{t_1, a_1, o_1, ..., t_{n-1}, a_{n-1}, o_{n-1}\}$, that is, the list of all thoughts, actions, and observations up to and including the previous time step. $\mathcal{S}_1$ is set to $\emptyset$.

Complex KBQA tasks require high reasoning ability. One promising approach to improve the reasoning capabilities of LLMs is ReAct prompting (Yao et al., 2023), in which the LLM is instructed to output its reasoning trace in natural language first, before selecting an action. We require the agent to output a *thought* at the beginning of each time step.

At time step n, the SPINACH agent takes a_n, one of the following five actions, and receives the result of the execution as observation o_n. These actions are selected for their similarity to what a human expert can do.

`search_wikidata(string)` is equivalent to using the text search box at `wikidata.org`. This action searches Wikidata for items (entities or properties) that match a given string using the Wikidata API[6] `wbsearchentities`. This API finds matches using human-readable labels and aliases. The main use for this action is to find relevant QIDs and PIDs.

`get_wikidata_entry(QID)` is equivalent to visiting the Wikidata page for an entity, e.g. `https://www.wikidata.org/wiki/Q219563`. This action retrieves all outgoing edges, i.e. linked entities, properties, and qualifiers of a specified Wikidata entity using its QID.

`get_property_examples(PID)` provides a few examples demonstrating the use of the specified property in Wikidata. The main use of this action is to understand how a property is used in Wikidata, if it is not clear from its label alone.

`execute_sparql(SPARQL)` is equivalent to using the Wikidata Query Service at `https://query.wikidata.org/`. It runs a SPARQL query on Wikidata and returns its results. Executing this action gives the agent one of the following four types of feedback signal:

1. *The query returns non-empty results.* We have observed that LLM is capable of reasoning over these results to choose the next action.
2. *The query returns empty results.* We have observed that the agent often attempts to debug the query clause-by-clause or attempts to take alternative actions.
3. *The query has a syntax error.* We show the specific syntax error message that the Wikidata Query API returns to the agent. We have observed that the agent can use the syntax error message to correct its previous syntactically incorrect SPARQL.
4. *The query execution times out.* The Wikidata Query API caps the execution at 60 seconds. When a timeout occurs, we have observed that

---

[6] `https://www.wikidata.org/w/api.php`

the agent attempts to simplify the SPARQL query in some way to avoid timeouts

`stop()` marks the last executed SPARQL query as the final answer and ends the process.

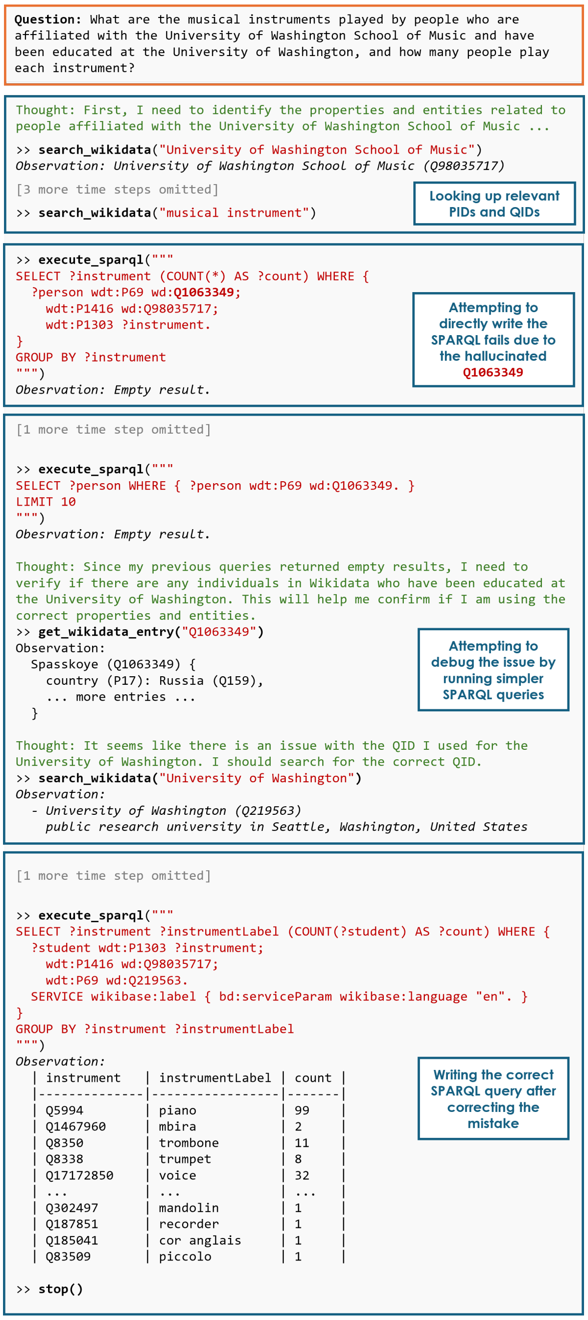
Importantly, the results of the execution of each action are put in a human-readable format to make it easier for the LLM to process. To limit the amount of information that the agent has to process, we limit the output of search results to at most 8 entities and 4 properties, and limit large results of SPARQL queries to the first and last 5 rows. The outputs of `get_wikidata_entry()` are pruned using a separate few-shot prompt (Prompt 13) to only keep entries that are relevant to the question.

4.2 Policy

The policy of SPINACH is implemented using a zero-shot prompt that only includes high-level instructions such as “start by constructing very simple queries and gradually build towards the complete query” and “confirm all your assumptions about the structure of Wikidata before proceeding” (Prompt 12). The decision of selecting the action at each time step is left to the LLM. This decision is made after experimenting with more restrictive policies, but we empirically observe that accounting for all various edge cases might confuse the agent and hurt the performance.

In each round of exploration, the agent constructs the exploration state and uses the policy prompt to ask LLM to output a thought and an action. It then executes the action and adds the observation to the state. In practice, the LLM can occasionally (1) fall into a cycle of repeating the same action and argument over and over again, or (2) deviate from its instructions and call **`stop`** even though the last SPARQL execution did not return any results. To remedy this, the agent checks for these potential problems and resets the exploration state to the one before the repetition or the deviation, and continues from there.

Note that the policy LLM is run using sampling with high temperature of 1.0 to encourage exploration, so rerunning from a previous state does not repeat the exact same sequence of actions. The agent loop continues until either (1) the `stop()` action is selected, or (2) 15 actions is taken after taking “rollbacks” into account, or (3) a total of 30 actions.[7].



```
Question: What are the musical instruments played by people who are
affiliated with the University of Washington School of Music and have
been educated at the University of Washington, and how many people play
each instrument?
```

```
Thought: First, I need to identify the properties and entities related to
people affiliated with the University of Washington School of Music ...
>> search_wikidata("University of Washington School of Music")
Observation: University of Washington School of Music (Q98035717)
[3 more time steps omitted]
>> search_wikidata("musical instrument")
```

Looking up relevant PIDs and QIDs

```
>> execute_sparql("""
SELECT ?instrument (COUNT(*) AS ?count) WHERE {
  ?person wdt:P69 wd:Q1063349;
    wdt:P1416 wd:Q98035717;
    wdt:P1303 ?instrument.
}
GROUP BY ?instrument
""")
Obesrvation: Empty result.
```

Attempting to directly write the SPARQL fails due to the hallucinated Q1063349

```
[1 more time step omitted]


>> execute_sparql("""
SELECT ?person WHERE { ?person wdt:P69 wd:Q1063349. }
LIMIT 10
""")
Obesrvation: Empty result.

Thought: Since my previous queries returned empty results, I need to
verify if there are any individuals in Wikidata who have been educated at
the University of Washington. This will help me confirm if I am using the
correct properties and entities.
>> get_wikidata_entry("Q1063349")
Observation:
  Spasskoye (Q1063349) {
    country (P17): Russia (Q159),
    ... more entries ...
  }

Thought: It seems like there is an issue with the QID I used for the
University of Washington. I should search for the correct QID.
>> search_wikidata("University of Washington")
Observation:
  - University of Washington (Q219563)
    public research university in Seattle, Washington, United States
```

Attempting to debug the issue by running simpler SPARQL queries

```
[1 more time step omitted]


>> execute_sparql("""
SELECT ?instrument ?instrumentLabel (COUNT(?student) AS ?count) WHERE {
  ?student wdt:P1303 ?instrument;
    wdt:P1416 wd:Q98035717;
    wdt:P69 wd:Q219563.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
}
GROUP BY ?instrument ?instrumentLabel
""")
Observation:
  | instrument    | instrumentLabel | count |
  |--------------|----------------|-------|
  | Q5994        | piano          | 99    |
  | Q1467960     | mbira          | 2     |
  | Q8350        | trombone       | 11    |
  | Q8338        | trumpet        | 8     |
  | Q17172850    | voice          | 32    |
  | ...          | ...            | ...   |
  | Q302497      | mandolin       | 1     |
  | Q187851      | recorder       | 1     |
  | Q185041      | cor anglais    | 1     |
  | Q83509       | piccolo        | 1     |

>> stop()
```

Writing the correct SPARQL query after correcting the mistake

Figure 2: The sequence of 13 actions that the SPINACH agent takes to answer a sample question from the SPINACH validation set. Here, the agent goes through several distinct phases, only with the high-level instruction in Section 4.2. Note that every step includes a thought, action and observation, but some are omitted here for brevity. Full version available in Listing 1.

In the example in Figure 2, SPINACH goes through the following four phases, just by following the high-level instruction we provided: it (1) looks up relevant PIDs and QIDs, (2) attempts to write the SPARQL in one go, which fails due to a

---

[7] The agent has a budget of 15 actions to take, and an extra 15 actions to spend on these “rollbacks”, to a total of 30 actions. For instance, the agent can take 9 actions, get rolled back by 4 actions, then take another 10 actions and stop. This means the final solution has $9 - 4 + 10 = 15 \leq 15$ actions, but the total number of actions is $9 + 4 + 10 = 23 \leq 30$

hallucinated QID that the agent did not previously check, (3) starts the debugging process by executing simpler SPARQL queries, and double-checking its assumptions. It realizes its mistake after looking at the result of a `get_wikidata_entry`, and finally (4) moves on to write the correct SPARQL query.

Discussion In our initial experiments with imposing low-level control over the selected actions, we found that it does not improve the accuracy of the agent. There are many possible states that the agent can encounter, and programming the best action in each one is challenging, especially in our zero-shot setting and without methods that can directly learn from large amounts of policy data.

Therefore, we leave most of the action selection to the LLM, and only impose have high-level control via the "policy prompt" (Prompt 12) with instructions like "Confirm all your assumptions about the structure of Wikidata before proceeding.", which encourages **`search_wikidata`** and **`get_wikidata_entry`** in the beginning, before executing any SPARQLs via **`execute_sparql`**. This way, we leverage the underlying LLM's common sense and world knowledge to reason over its observations. For example, we saw empirically that the LLM is capable of determining if the result of a SPARQL query appears to be implausible (e.g. returning a long list of entities when the question is about "the oldest").

5 Experiments

5.1 Evaluation Metrics

Prior works mostly use two metrics: Exact Match (EM) and F_1 (Yih et al., 2016; Usbeck et al., 2023). As shown in Table 1, the SPINACH dataset contains 2.5 projections on average for each query, whereas prior datasets all only contain only one field in projection. This means that naively applying EM and F_1 is not possible; therefore, we propose a generalization of these metrics.

From the user's perspective, answers containing more than the minimum information which help clarify or enrich the answer are welcome. For the query "what is the county with most people in South Dakota?", a LLM-based system can choose to return the top county along with its population, even though the gold answer contains only the county. In other words, answers that contain additional projection columns from the gold should not be penalized. To reflect this, we introduce a **row-major** generalization of EM and F1 to handle matrix-wise comparisons in real-life KBQA tasks, where each row is andled such that extra columns are not penalized.

Consider the general case where the answer of a query consists of m projections of n results. Let

$$\mathbf{y} = \begin{pmatrix} y_{11} & \cdots & y_{1m} \\ \vdots & \ddots & \vdots \\ y_{n1} & \cdots & y_{nm} \end{pmatrix} \quad \mathbf{y}' = \begin{pmatrix} y'_{11} & \cdots & y'_{1m'} \\ \vdots & \ddots & \vdots \\ y'_{n'1} & \cdots & y'_{n'm'} \end{pmatrix}$$

be the gold and predicted result, respectively.

We first define the recall between a row in gold and a row in predicted such that additional columns in the predicted result are not penalized. Let y_i and y'_j be the i and j rows in the gold and predicted results, respectively.

$$\text{recall}(y_i, y'_j) = \frac{|y_i \cap y'_j|}{|y_i|}$$

To calculate the recall between the full gold and the predicted answer, we first find assignment (SciPy, 2024) $A(\mathbf{y}, \mathbf{y}') = \{(i_1, j_1), ..., (i_r, j_r)\}$ where row y_{i_k} is matched with y'_{j_k} such that the sum of the recall between the matched rows is maximized. (Matching rows with 0 recall is disallowed.)

For the calculation of F_1, given $\mathbf{y}$ with n rows, $\mathbf{y}'$ with n' rows, and $A(\mathbf{y}, \mathbf{y}') = \{(i_1, j_1), ..., (i_r, j_r)\}$, the number of true positives, false negatives, and false positives are, respectively:

$$tp = \sum_{(i,j) \in A(\mathbf{y}, \mathbf{y}')} \text{recall}(y_i, y'_j)$$

$$fn = n - r + \sum_{(i,j) \in A(\mathbf{y}, \mathbf{y}')} 1 - \text{recall}(y_i, y'_j)$$

$$fp = n' - r$$

The row-major F_1 between $\mathbf{y}$ and $\mathbf{y}'$ is:

$$\frac{2tp}{2tp + fp + fn}$$

EM is defined to be 1 if the row-major F_1 score is 1 and 0 otherwise. Note that with this definition, EM and F_1 are exactly equal to the tradition al definition of EM and F_1 when there is only one projection.

5.2 SPINACH Agent on Prior Datasets

We evaluate our approach on four previous Wikidata datasets. We experiment with three QALD challenges that have annotated Wikidata SPARQLs: QALD-7 (task 4 for Wikidata) (Usbeck et al.,

	QALD-7 (Task 4)		QALD-9 Plus (en)		QALD-10 (en)				WikiWebQuestions			
	Test		Test		Full Test Set		Subset in ToG		Dev		Test	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
STAGG (Yih et al., 2016)	-	19.0	-	-	-	-	-	-	-	-	-	-
GGNN (Sorokin and Gurevych, 2018)	-	21.3	-	-	-	-	-	-	-	-	-	-
LingTeQA (To and Reformat, 2020)	-	34.0	-	-	-	-	-	-	-	-	-	-
Baramiia et al. (2022)	-	-	-	-	-	42.8	-	-	-	-	-	-
Shivashankar et al. (2022)	-	-	-	-	-	49.1	-	-	-	-	-	-
QAnswer (Diefenbach et al., 2017a)	-	40.0	-	44.6	-	57.8	-	-	-	-	-	-
SPARQL-QA (Borroto et al., 2022)	-	-	-	-	-	59.5	-	-	-	-	-	-
Liu et al. (2024)	-	-	-	-	56.5	-	-	-	-	-	-	-
0-shot ToG (GPT-4) (Sun et al., 2024a)	-	-	-	-	-	-	54.7	-	-	-	-	-
Fine-tuned WikiSP (Xu et al., 2023)	38.0	43.6	-	-	-	-	-	-	**75.6**	**76.9**	**65.5**	**71.9**
0-shot SPINACH agent (GPT-4o) **(Ours)**	**62.2**	**74.6**	**58.3**	**71.6**	**63.1**	**69.5**	**64.7**	**72.4**	61.2	72.3	59.9	70.3

Table 2: Performance of the SPINACH agent and prior works on 4 prior datasets. Sun et al. (2024a) only evaluated on the subset of non-boolean questions of QALD-10 test set, which we denote as "Subset in ToG". SPINACH achieves the new SOTA on QALD-7, QALD-9 Plus, and QALD-10. On WikiWebQuestions, it comes within 1.6% F1 to the SOTA WikiSP fine-tuned on the dataset.

	Dev		Test	
	EM	F1	EM	F1
Direct GPT-4o Question Answering	0.0	3.9	0.0	4.0
GPT-4o Generating SPARQL	1.3	5.4	0.6	3.9
Fine-tuned WikiSP (Xu et al., 2023)	1.3	3.5	1.2	7.1
0-shot ToG (GPT-4) (Sun et al., 2024a)	3.9	9.8	1.8	7.2
0-shot SPINACH agent (GPT-4o) **(Ours)**	**21.4**	**46.4**	**16.4**	**45.3**

Table 3: Evaluation of the SPINACH agent and prior works on the SPINACH dataset.

2017), QALD-9 Plus (Perevalov et al., 2022), and the English subset of QALD-10 (Usbeck et al., 2023). In addition, we also evaluate with WikiWebQuestions (Xu et al., 2023), the Wikidata version of the popular WebQuestionSP dataset.

As shown in Table 2, the SPINACH agent achieves new SOTA on the three QALD datasets and comes within 1.6% F1 of the fine-tuned SOTA (WikiSP) on WikiWebQuestion. In particular, our 0-shot agent outperforms WikiSP by 24.2% EM and 31.0% F1 on Qald-7 (Task 4), showing its impressive cross-dataset generalization capability compared to a fine-tuned model. Our approach also outperforms the GPT-4-based ToG (Sun et al., 2024a) by 10.0% EM on the same subset of QALD-10 used for evaluation in its paper.

5.3 Results on the SPINACH Dataset

We benchmark the performance of four baseline systems on the SPINACH dataset. These baselines are: (1) directly asking GPT-4o to answer the question without writing SPARQL, (2) directly asking GPT-4o to write a SPARQL query, (3) WikiSP (Xu et al., 2023) which is a fine-tuned LLaMA (7B-parameter) model, and (4) the GPT-4-based ToG agent (Sun et al., 2024a).

WikiSP expects a Named Entity Disambiguation (NED) module to predict the relevant entities given a question. The original NED module was based on a fine-tuned version of ReFinED (Ayoola et al., 2022). To make the task easier for WikiSP, we directly use the gold entities that appear in the annotated SPARQL as inputs. Similarly, the ToG (Sun et al., 2024a) pipeline uses LLM to extract "topic entities" from the input question as the starting point of its knowledge graph exploration. We observed that using all the gold entities as topic entities, the model was unable to return any exploration paths for the first 80 questions in the validation set and simply defaulted to GPT-4. To make the task even easier for ToG, we sample up to 20 entities from the results of the gold SPARQL queries as the topic entities. Additional details on our setup and comparison of baseline systems can be found in Appendix D.

As shown in Table 3, the SPINACH agent drastically outperforms all baselines. In particular, we highlight that the low accuracy of the GPT-4o QA system shows that this dataset mainly consists of long-tail knowledge unfamiliar to LLMs. For reference, GPT-4 was able to achieve 90.5% EM on WebQuestionSP (Tan et al., 2023). Although SPINACH was able to achieve the SOTA on this dataset, the relatively lower performance compared to those achieved on previous dataset indicates that there is a lot of room for improving KBQA systems in the future.

5.4 Error Analysis

We randomly sample 20 cases on the SPINACH dataset where the SPINACH agent achieves less than 0.05 F_1 and conduct an error analysis. We observe that:

- **Property-related problems**: 40% of errors are due to problems with properties. This includes cases when the SPINACH agent fails to fetch the correct property or incorrectly uses a property (e.g. using it as a `wdt:` relation as opposed to the correct `ps:` or `pq:` qualifier).
- **Complicated SPARQL**: 30% of errors are due to the failure of the SPINACH agent to write complex SPARQL to fetch results, e.g., a complicated filter for finding people born in a specific month.[8]
- **Not enough exploration**: 15% of errors are due to insufficient exploration performed by the LLM after reaching the maximum allowed number of actions.
- **Inaccurate semantic parsing**: 10% of errors are due to the LLM injecting an extra clause. For instance, when asked "what items were published in ...", the LLM assumes that the results have to be an instance of the domain entity "book", but there are many other types of results returned by the gold query (e.g. "printed matter" and "legal act").
- **Formatting issues**: 5% of errors are due to format errors. An example is the LLM returning the date in full ("June 23 2021") when only the year is requested ("2021").

5.5 Ablation Study and Analysis

We created the set of agent actions from our own experience in writing SPARQL. To understand how each action contributes to the performance of the SPINACH agent, we remove the actions one at a time (except **`execute_sparql`** and **`stop`**, without which the agent would never return a SPARQL query). The results on the SPINACH dev set are reported in Table 4. This ablation shows that all actions together contribute to the performance of the SPINACH agent. We also report the distribution of the number of actions the SPINACH agent took to answer questions from the SPINACH dev set in Table 5 and the distribution of the number of tokens in Appendix D.1.

	EM	F1
SPINACH agent	**21.4**	**46.4**
w/o **`get_wikidata_entry`**	11.7	36.4
w/o **`get_property_examples`**	10.4	29.4
w/o **`search_wikidata`**	4.6	25.3

Table 4: Ablation study of SPINACH agent without each of its actions on the SPINACH dev set.

# of actions	Percentage
3-5	28.6%
6-8	33.8%
9-11	11.0%
12-14	7.1%
15	19.5%

Table 5: Distribution of the number of actions (without counting rollbacked actions) the SPINACH agent takes to answer a question from the SPINACH dev set.

6 Conclusion

We propose the expert-annotated SPINACH dataset drawn from real-world queries and introduce the SPINACH agent that mimics how a human expert writes SPARQL queries. Experiments on prior works show that the SPINACH agent achieves the new SOTA on 3 QALD datasets and comes within 1.6% F1 to the fine-tuned SOTA model on WikiWebQuestions. On the new SPINACH dataset, our agent outperforms the best LLM agent approach, ToG, by 35.68% F_1 on the test set. The performance of 16.4% EM and 45.3% F_1 on the test set suggests ample opportunity for further improvement. Fortunately, as our SPINACH agent diaplays all its intermediate steps, users can continue the conversation and revise their queries to help the agent derive the answer. We have deployed the SPINACH agent at `https://spinach.genie.stanford.edu/` and on Wikidata at `https://www.wikidata.org/wiki/User:SpinachBot` as a community resource to facilitate the access of data in Wikidata.

Limitations

Since SPINACH agent makes multiple LLM calls for each question, its latency and cost are higher

---

[8] An interesting observation is that in 2 cases, GPT-4o ended the reasoning with "I will fetch all related records and proceed to filter the results in Python" after observing its previous SPARQL returned no results, suggesting there may be benefits to leveraging programming languages that are more familar to LLMs.

compared to simpler systems. Other multi-stage KBQA pipelines with LLMs like Sun et al. (2024a) suffer from the same limitations. This seems to be the price for a more accurate KBQA system. However, recently, model distillation has been shown to be effective in improving model efficiency (Semnani et al., 2023). We leave the exploration of this direction for future work.

As observed in recent works (Kamalloo et al., 2023; Zhang et al., 2024), automatic metrics such as EM and F_1 often fail to accurately measure the capabilities of LLM-based systems. By verbalizing the SPARQL queries as accurately as possible, we aim to minimize the difference between automatic metrics and manual inspection. We thus expect these automatic metrics to still play a pivotal role in the evaluation of future work on the SPINACH dataset.

Annotated datasets can also suffer from annotation errors and mistakes (Zang et al., 2020). We address this limitation by having experts annotate the dataset and independently cross-validating each other's annotations, as discussed in Appendix C.3.

Ethical Considerations

We release the SPINACH dataset in this paper. This dataset is built using publicly available data in the Wikidata Request Query forum, which is licensed under the Creative Commons CC0 License. We don't expect any harm being produced from the usage of this dataset. The generation and annotation of the dataset were done by the authors of this paper.

No GPU-based training was carried out in our experiments, as we mainly used the OpenAI API to call the GPT-4o model. To evaluate the LLaMA baseline from Xu et al. (2023), we used a Linux server with one NVIDIA A100 GPU for <1 hour.

To facilitate further research, we release our code and data for the SPINACH agent and the SPINACH dataset. The SPINACH agent is released under Apache License, version 2.0. The SPINACH dataset, derived from the Wikidata Request a Query forum, is released under the CC BY-SA 4.0 license, the same license that covers the forum.

Acknowledgements

We thank Isaac Johnson at the Wikimedia Foundation for helpful discussions about this work, Harshit Joshi for his help with the project, the ACL ARR reviewers for their valuable comments and suggestions, and members of the Stanford Open Virtual Assistant Lab (OVAL) and NLP group for their feedback and support. This work is supported in part by the Verdant Foundation, Microsoft Azure AI credits, KDDI, and the Stanford Human-Centered Artificial Intelligence (HAI) Institute.

References

Tom Ayoola, Shubhi Tyagi, Joseph Fisher, Christos Christodoulopoulos, and Andrea Pierleoni. 2022. ReFinED: An efficient zero-shot-capable approach to end-to-end entity linking. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 209–220, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514, Osaka, Japan. The COLING 2016 Organizing Committee.

Nikita Baramiia, Alina Rogulina, Sergey Petrakov, Valerii Kornilov, and Anton Razzhigaev. 2022. Ranking approach to monolingual question answering over knowledge graphs. In *NLIWoD@ ESWC*, pages 32–37.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA. Association for Computing Machinery.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *Preprint*, arXiv:1506.02075.

Manuel Borroto, Francesco Ricca, Bernardo Cuteri, and Vito Barbara. 2022. Sparql-qa enters the qald challenge. In *Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference, Hersonissos, Greece*, volume 3196, pages 25–31.

Giovanni Campagna, Sina Semnani, Ryan Kearns, Lucas Jun Koba Sato, Silei Xu, and Monica Lam. 2022. A few-shot semantic parser for Wizard-of-Oz dialogues with the precise ThingTalk representation. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 4021–4034, Dublin, Ireland. Association for Computational Linguistics.

Shulin Cao, Jiaxin Shi, Liangming Pan, Lunyiu Nie, Yutong Xiang, Lei Hou, Juanzi Li, Bin He, and Hanwang Zhang. 2022a. KQA pro: A dataset with explicit compositional programs for complex question answering over knowledge base. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6101–6119, Dublin, Ireland. Association for Computational Linguistics.

Shulin Cao, Jiaxin Shi, Zijun Yao, Xin Lv, Jifan Yu, Lei Hou, Juanzi Li, Zhiyuan Liu, and Jinghui Xiao. 2022b. Program transfer for answering complex questions over knowledge bases. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8128–8140, Dublin, Ireland. Association for Computational Linguistics.

Dallas Card, Peter Henderson, Urvashi Khandelwal, Robin Jia, Kyle Mahowald, and Dan Jurafsky. 2020. With little power comes great responsibility. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9263–9274, Online. Association for Computational Linguistics.

Ruixiang Cui, Rahul Aralikatte, Heather Lent, and Daniel Hershcovich. 2022. Compositional generalization in multilingual semantic parsing over Wikidata. *Transactions of the Association for Computational Linguistics*, 10:937–955.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dennis Diefenbach, Kamal Singh, and Pierre Maret. 2017a. Wdaqua-core0: A question answering component for the research community. In *Semantic Web Challenges: 4th SemWebEval Challenge at ESWC 2017, Portoroz, Slovenia, May 28-June 1, 2017, Revised Selected Papers*, pages 84–89. Springer.

Dennis Diefenbach, Thomas Pellissier Tanon, Kamal Deep Singh, and Pierre Maret. 2017b. Question answering benchmarks for wikidata. In *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017.*

Mohnish Dubey, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. 2019. Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia. In *The Semantic Web – ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II*, page 69–78, Berlin, Heidelberg. Springer-Verlag.

Wolfgang Fahl, Tim Holzheim, Andrea Westerinen, Christoph Lange, and Stefan Decker. 2022. Getting and hosting your own copy of wikidata.

Mariam Farda-Sarbas and Claudia Müller-Birn. 2019. Wikidata from a research perspective – a systematic mapping study of wikidata. *Preprint*, arXiv:1908.11153.

Catherine Finegan-Dollak, Jonathan K. Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-SQL evaluation methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 351–360, Melbourne, Australia. Association for Computational Linguistics.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. 2021. Beyond i.i.d.: Three levels of generalization for question answering on knowledge bases. In *Proceedings of the Web Conference 2021*, WWW '21, page 3477–3488, New York, NY, USA. Association for Computing Machinery.

Yu Gu and Yu Su. 2022. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1718–1731, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5591–5606, Toronto, Canada. Association for Computational Linguistics.

Effie Kapsalis. 2019. Wikidata: Recruiting the crowd to power access to digital archives. *Journal of Radio & Audio Media*, 26(1):134–142.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, Dmitry Tsarkov, Xiao Wang, Marc van Zee, and Olivier Bousquet. 2020. Measuring compositional generalization: A comprehensive method on realistic data. In *International Conference on Learning Representations*.

Vladislav Korablinov and Pavel Braslavski. 2020. Rubq: A russian dataset for question answering over wikidata. In *The Semantic Web – ISWC 2020*, pages 97–110, Cham. Springer International Publishing.

Yunshi Lan and Jing Jiang. 2020. Query graph generation for answering multi-hop complex questions from knowledge bases. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 969–974, Online. Association for Computational Linguistics.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jiaxiang Liu, Tong Zhou, Yubo Chen, Kang Liu, and Jun Zhao. 2024. Enhancing large language models with pseudo- and multisource- knowledge graphs for open-ended question answering. *Preprint*, arXiv:2402.09911.

Xue lu Yu and Lin Qiao. 2017. Meronymy relation extraction based on 3-motif in wikidata. *DEStech Transactions on Computer Science and Engineering*.

Haoran Luo, Haihong E, Zichen Tang, Shiyao Peng, Yikai Guo, Wentai Zhang, Chenghao Ma, Guanting Dong, Meina Song, and Wei Lin. 2024a. ChatKBQA: A generate-then-retrieve framework for knowledge base question answering with fine-tuned large language models.

Kangqi Luo, Fengli Lin, Xusheng Luo, and Kenny Zhu. 2018. Knowledge base question answering via encoding of complex query graphs. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2185–2194, Brussels, Belgium. Association for Computational Linguistics.

Linhao Luo, Yuan-Fang Li, Reza Haf, and Shirui Pan. 2024b. Reasoning on graphs: Faithful and interpretable large language model reasoning. In *The Twelfth International Conference on Learning Representations*.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Costas Mavromatis and George Karypis. 2022. ReaRev: Adaptive reasoning for question answering over knowledge graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2447–2458, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

MediaWiki. 2024a. Wikidata query service/user manual — mediawiki,. [Online; accessed 13-June-2024].

MediaWiki. 2024b. Wikidata: Sparql query service/wdqs graph split. Accessed: 2024-06-12.

Elvira Mitraka, Andra Waagmeester, Sebastian Burgstaller-Muehlbacher, Lynn M. Schriml, Andrew I. Su, and Benjamin M. Good. 2015. Wikidata: A platform for data integration and dissemination for the life sciences and beyond. *bioRxiv*.

Inbar Oren, Jonathan Herzig, and Jonathan Berant. 2021. Finding needles in a haystack: Sampling structurally-diverse training sets from synthetic data for compositional generalization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10793–10809, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Aleksandr Perevalov, Dennis Diefenbach, Ricardo Usbeck, and Andreas Both. 2022. Qald-9-plus: A multilingual dataset for question answering over dbpedia and wikidata translated by native speakers. In *2022 IEEE 16th International Conference on Semantic Computing (ICSC)*, pages 229–234.

Alexander Pfundner, Tobias Schönberg, John Horn, Richard D Boyce, and Matthias Samwald. 2015. Utilizing the wikidata system to improve the quality of medical content in wikipedia in diverse languages: a pilot study. *Journal of Medical Internet Research*, 17(5):e110.

Ivan Rybin, Vladislav Korablinov, Pavel Efimov, and Pavel Braslavski. 2021. Rubq 2.0: An innovated russian question answering dataset. In *The Semantic Web*, pages 532–547, Cham. Springer International Publishing.

Philipp Scharpf, Moritz Schubotz, and Bela Gipp. 2021. Mathematics in wikidata. In *International Semantic Web Conference (ISWC) 2021*. Virtual event.

SciPy. 2024. *scipy.optimize.linear_sum_assignment*. Accessed: 2024-07-11.

Sina Semnani, Violet Yao, Heidi Zhang, and Monica Lam. 2023. WikiChat: Stopping the hallucination of large language model chatbots by few-shot grounding on Wikipedia. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2387–2413, Singapore. Association for Computational Linguistics.

Priyanka Sen, Armin Oliya, and Amir Saffari. 2021. Expanding end-to-end question answering on differentiable knowledge graphs with intersection. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8805–8812, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kartik Shenoy, Filip Ilievski, Daniel Garijo, Daniel Schwabe, and Pedro Szekely. 2022. A study of the quality of wikidata. *Journal of Web Semantics*, 72:100679.

Kanchan Shivashankar, Khaoula Benmaarouf, and Nadine Steinmetz. 2022. From graph to graph: Amr to sparql. In *Proceedings of the 7th Natural Language Interfaces for the Web of Data (NLIWoD) co-located with the 19th European Semantic Web Conference (ESWC 2022), Hersonissos, Greece, 29th May*.

Shani Evenstein Sigalov and Rafi Nachmias. 2023. Investigating the potential of the semantic web for education: Exploring wikidata as a learning platform. *Education and Information Technologies*, pages 1 – 50.

Daniil Sorokin and Iryna Gurevych. 2018. Modeling semantics with gated graph neural networks for knowledge base question answering. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3306–3317, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2380–2390, Hong Kong, China. Association for Computational Linguistics.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242, Brussels, Belgium. Association for Computational Linguistics.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2024a. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Lei Sun, Zhengwei Tao, Youdi Li, and Hiroshi Arakawa. 2024b. Oda: Observation-driven agent for integrating llms and knowledge graphs. *Preprint*, arXiv:2404.07677.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 641–651, New Orleans, Louisiana. Association for Computational Linguistics.

Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. 2023. Can chatgpt replace traditional kbqa models? an in-depth analysis of the question answering performance of the gpt llm family. *Preprint*, arXiv:2303.07992.

Nhuan D. To and Marek Reformat. 2020. Question-answering system with linguistic terms over rdf knowledge graphs. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 4236–4243.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *Preprint*, arXiv:2302.13971.

Houcemeddine Turki, Mohamed Ali Hadj Taieb, Mohamed Ben Aouicha, Lane Rasberry, and Daniel Mietchen. 2023. [novel] ten years of wikidata: A bibliometric study. In *The 4th Wikidata Workshop*.

Houcemeddine Turki, Denny Vrandecic, Helmi Hamdi, and Imed Adel. 2017. Using wikidata as a multilingual multi-dialectal dictionary for arabic dialects. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 437–442.

Ricardo Usbeck, Ria Hari Gusmita, Axel-Cyrille Ngonga Ngomo, and Muhammad Saleem. 2018. 9th challenge on question answering over linked data (qald-9) (invited paper). In *Semdeep/NLIWoD@ISWC*.

Ricardo Usbeck, Axel-Cyrille Ngonga Ngomo, Bastian Haarmann, Anastasia Krithara, Michael Röder, and Giulio Napolitano. 2017. 7th open challenge on question answering over linked data (qald-7). In *Semantic web evaluation challenge*, pages 59–69. Springer.

Ricardo Usbeck, Xi Yan, Aleksandr Perevalov, Longquan Jiang, Julius Schulz, Angelie Kraft, Cedric Möller, Junbo Huang, Jan Reineke, Axel-Cyrille Ngonga Ngomo, Muhammad Saleem, and Andreas Both. 2023. Qald-10 – the 10th challenge on question answering over linked data. *Semantic Web*, semantic-web/sw233471.

Pat Verga, Haitian Sun, Livio Baldini Soares, and William Cohen. 2021. Adaptable and interpretable neural MemoryOver symbolic knowledge. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3678–3691, Online. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Wikidata. 2024. Wikidata: Sparql tutorial. Accessed: 2024-10-04.

Guanming Xiong, Junwei Bao, and Wen Zhao. 2024. Interactive-kbqa: Multi-turn interactions for knowledge base question answering with large language models. *Preprint*, arXiv:2402.15131.

Silei Xu, Shicheng Liu, Theo Culhane, Elizaveta Pertseva, Meng-Hsi Wu, Sina Semnani, and Monica Lam. 2023. Fine-tuned LLMs know more, hallucinate less with few-shot sequence-to-sequence semantic parsing over Wikidata. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5778–5791, Singapore. Association for Computational Linguistics.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Xi Ye, Semih Yavuz, Kazuma Hashimoto, Yingbo Zhou, and Caiming Xiong. 2022. RNG-KBQA: Generation augmented iterative ranking for knowledge base question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6032–6043, Dublin, Ireland. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Wen-tau Yih, Matthew Richardson, Chris Meek, Ming-Wei Chang, and Jina Suh. 2016. The value of semantic parse labeling for knowledge base question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–206, Berlin, Germany. Association for Computational Linguistics.

Donghan Yu, Sheng Zhang, Patrick Ng, Henghui Zhu, Alexander Hanbo Li, Jun Wang, Yiqun Hu, William Yang Wang, Zhiguo Wang, and Bing Xiang. 2023. DecAF: Joint decoding of answers and logical forms for question answering over knowledge bases. In *The Eleventh International Conference on Learning Representations*.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. MultiWOZ 2.2 : A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117, Online. Association for Computational Linguistics.

Heidi C. Zhang, Sina J. Semnani, Farhad Ghassemi, Jialiang Xu, Shicheng Liu, and Monica S. Lam. 2024. Spaghetti: Open-domain question answering from heterogeneous data sources with retrieval and semantic parsing. *Preprint*, arXiv:2406.00562.

Fudie Zhao. 2022. A systematic review of Wikidata in Digital Humanities projects. *Digital Scholarship in the Humanities*, 38(2):852–874.

A Definition of metrics used in Table 1

For each query, we define the number of clauses as the number of atomic nodes of a SPARQL abstract syntax tree (AST), where an atomic node is defined as one of: a projection clause (i.e., `SELECT ... WHERE`), a single subject-relation-object clause, a group by clause, a having by clause, a filter clause, a sorting clause, a `MINUS` clause, and a join clause. The number of projections is defined as the number of fields in the final `SELECT ... WHERE` clause. The number of relations is defined as the number of subject-relation-object clauses. The number of subjects is defined to be the number of uniquely occuring subjects (either variables or entities) in each subject-relation-object clause. The number of predicates is defined to be the unique number of properties (i.e. a PID of the form `P...`). The number of objects is defined to be the number of uniquely occuring objects (either variables or entities) in each subject-relation-object clause, where we also count the number of `y` occuring in each `VALUES` clause of the form `VALUES ?x {y}`. The number of literals is defined to be the unique number of strings (e.g. “`wikidata`”) or numerical numbers (e.g. `123.45`). The number of unique properties is the number of Wikidata properties across all queries in a dataset (e.g. `P123`). The number of unique forms is the number of unique query patterns that occur in a dataset when one ignores all unique query variables, property IDs (PIDs), entity IDs (QIDs), numbers, and string literals and simply counts the resulting patterns.

Similar to prior work (Gu et al., 2021), we use hand-crafted rules (including regular expressions) to retrieve these metrics.

B Additional Comparison between SPINACH and Prior KBQA Datasets

Datasets with synthetically generated SPARQL queries typically attempt to cover the space of possible SPARQLs using heuristics, resulting in relatively easy SPARQL structures repeated over and over again. To quantitatively show this difference, we calculate the number of unique query patterns similar to what was proposed for SQL in Finegan-Dollak et al. (2018). Specifically, we de-duplicate SPARQL queries after ignoring all query variables, property IDs (PIDs), entity IDs (QIDs), numbers, and string literals and count the resulting patterns. Table 6 shows the comparison of the number of unique query patterns for SPINACH and several other KBQA datasets. As shown here, even though synthetic datasets such as GrailQA, KQA-Pro, and CWQ contain many more data points, a small portion of query patterns are repeated over and over again. SPINACH, on the other hand, features organic queries found in the wild where each query is structurally unique and poses a new challenge.

Dataset	Size	UQPs	UQPs / Size
GrailQA (train+dev)	51100	116	0.227%
KQA-Pro	117970	1689	1.432%
CWQ	34689	402	1.159%
WikiWebQuestions	4316	176	0.041%
SPINACH	320	320	100.0%

Table 6: Comparison of Unique Query Patterns (UQPs) in SPINACH and prior works.

We also report the number of unique properties in SPARQL queries of the SPINACH dataset and prior Wikidata datasets in Table 7. Datasets that repeat the same few properties over and over again have a low diversity of logical forms. Datasets that synthesize their logical forms like MCWQ especially have a low count of unique properties and do not properly test the ability of systems to explore the knowledge graph as they can simply memorize all of them during training. As shown here, SPINACH contains the most unique properties, and is quite diverse in this sense.

Dataset	# UPs	Dataset Size
MCWQ	27	124187
QALD 9	158	507
QALD 10	177	394
RUBQ	251	2910
WikiWebQuestions	189	4316
SPINACH	298	323

Table 7: Comparison of number of Unique Properties (UPs) in SPINACH and prior works.

C Additional Details on SPINACH Dataset

C.1 Additional Details on the Request a Query Forum

We have found that posters on the Request a Query forum generally come to the forum with a clear understanding of what they would like to query, or even have a sample SPARQL query in a different field that they would like to adapt (e.g. converting a query about basketball statistics to baseball statistics, or converting a query about museums in the San Francisco area to operas in the Paris). In other words, most questions are already very complex in the first post on the forum.

We have analyzed the source conversations of data points in the SPINACH dataset to derive the distribution of conversation lengths, where longer conversations may have more back-and-forth between users who are asking questions and users who are attempting to respond to those questions. We exclude conversations from the Request a Query forum that have no responses, as well as conversations that do not contain any SPARQL queries as there is no good candidate SPARQL to use for annotation. Shorter conversations (1-3 responses) comprise 57% of the dataset, medium length conversations (4-6 responses) comprise 31.9% of the dataset, and longer conversations (7 or more responses) comprise 11.1% of the dataset. A detailed breakdown is available in Table 8. The length of a conversation is defined to be the number of responses posted.

length of conv.	percentage
1	13.9%
2	14.9%
3	28.2%
4	13.3%
5	13.0%
6	5.6%
7	4.0%
8	1.9%
9	2.2%
>10	3.1%

Table 8: Distribution of the length of conversations for each of data point in the SPINACH.

C.2 Discussion on Size and Statistical Power of SPINACH

NLP datasets are often used to differentiate between two proposed systems, for example, to determine if a new system outperforms the state-of-the-art or not. The size of the validation/test set determines the minimum detectable effect (MDE) (Card et al., 2020), i.e. the minimum improvement in a metric (like exact match) that will yield sufficiently powered comparisons. Following the approach in Card et al. (2020), we estimate that the SPINACH dev set is sufficiently large to differentiate between the SPINACH agent and systems that are 9.0% or more better than it. Meaning, that if a future paper presents a system scoring 30.4% EM or higher on the dev set of SPINACH, their experiment will be statistically significant with the commonly used significance level (alpha) of 5%. Separately, we also note that all the improvements we report compared to our baselines in Table 3 are statistically significant.

Many influential datasets in the LLM era contain fewer examples than traditional ML datasets, often because they are only meant to serve as a validation and test set instead of a training set. The SPINACH dataset calls for systems that can dynamically explore large and often incomplete KB schemas and reason about them, as opposed to relying on training data. The poor performance of Xu et al. (2023), a model fine-tuned on a larger training set, showcases that models fine-tuned on often larger datasets do not necessarily generalize well (Table 3). Furthermore, many recent LLM-based systems (e.g. one of our baselines, ToG (Sun et al., 2024a)) only evaluate on smaller datasets, or small subsets of larger datasets, due to the high cost of LLM APIs.

For these reasons, we believe the SPINACH dataset is a good middle ground that keeps the cost of our expert annotation and running multiple experiments in future papers manageably low, while being reasonably powered to differentiate between and track the progress of future systems that use this dataset.

C.3 Additional Details on Annotation

Expert annotations: The SPINACH dataset is annotated by 3 authors of this paper who are experts with extensive knowledge in SPARQLs and Wikidata. The 3 experts first engaged in a long discussion with specific examples to standardize the annotation procedure. 2 experts first annotated the dev and test sets, with the 3rd expert double-checking and validating the annotations.

LLM suggestions: To facilitate the annotation process, we used GPT-4o to generate preliminary annotation suggestions to the experts. For each example, the experts are shown with (1) the original SPARQL query and with properties & entities substituted with labels, (2) link to the specific forum discussion, (3) two LLM-suggested verbalizations (one more verbose and one more natural), and (4) an LLM suggestion of whether to modify, include, or exclude the query with its reasonings. For each data example, the experts executed numerous intermediate SPARQL queries on the Wikidata site to verify the query's validity and finalize the annotated SPARQL.

Automatic exclusion of mwapi in queries: Some SPARQL queries on the forum make use of `mwapi`[9]. The usage of these APIs are mostly for optimization or are otherwise very specific to the structure of Wikimedia sites. We note that sometimes, removing these APIs would lead to small changes in the results (most likely due to the outdated discrepancy between the API outputs and Wikidata-direct outputs). To standardize the dataset and avoid issues during evaluation, we use regex to delete all occurrences of `mwapis` in the SPARQLs before executing them.

C.4 Examples of queries modified

The following are examples of how we modify target SPARQLs.

Wikimedia presentation queries: This dicsussion, titled "Two-letter genera, on Wikispecies" in July of 2020, involves the following snippets of retrieving information from `species.wikimedia.org` for certain taxa:

```
?wikispecies schema:about ?item .
?wikispecies schema:isPartOf
    <https://species.wikimedia.org/> .
```

This, however, is very specific to the structure of different Wikimedia sites. Removing these two clauses does not modify the core parts of the meaning. We thus remove these two clauses and the corresponding target `?wikispecies` in the projection. The resulting SPARQL is then included in the SPINACH validation set.

Queries obscured by optimizations: This discussion, titled "Persons from a certain time period and country" in Septempter of 2020, contains a query that makes use of the following optimization:

```
int:Prior hint:rangeSafe true .
```

which is used to speed up the succeeding filter. Removing this optimization in this case does not significantly increase SPARQL executing time. We

---

[9] https://en.wikibooks.org/wiki/SPARQL/SERVICE_-_mwapi

thus remove this clause and include the example in the SPINACH validation set.

Formatting clauses: This discussion, titled "Query all taxons which are trees", results in a SPARQL of the following structure:

```
SELECT
  ?taxon ?sample (GROUP_CONCAT(DISTINCT
      str(?commonname); separator =
      "//") as ?commonnames)
WHERE
{
...
}
GROUP BY ?taxon ?sample
```

Using a group by, it is trying to concatenate all `commonnames` into one single string, separated by `"//"`. However, it is very difficult to accurately capture this in natural language, and a system could return the results in different orders, raising issues for evaluation. We thus exclude this concatenation and instead change it to a counting operation:

```
SELECT
  ?taxon ?sample (COUNT(?commonname) as
      ?commonnamecount)
WHERE
{
...
}
GROUP BY ?taxon ?sample
```

The resulting SPARQL is then included in the SPINACH validation set.

C.5 Examples of queries excluded

The following are examples of when, after modifications, no meaningful part of the SPARQL remains. The discussions are thus excluded from our dataset.

Wikimedia presentation queries: The discussion, titled "?item =wd:QXXX" in April of 2020, results in a SPARQL that is only trying to fetch wikipedia pages that are about the item "Gambling, Gods And LSD". The usage of `schema:about` property is very specific to Wikimedia and could raise confusion on the meaning of "about". It is thus excluded. The discussion, titled "Bengali Wikipedia articles with no Wikidata statement " in March of 2018, results in a SPARQL trying to fetch Wikidata items with no statements that are the topic of discussion of Bengali Wikipedia articles. The usage of `wikibase:statements`, `schema:about`, `schema:isPartOf`, and `wikibase:sitelinks` properties are all very specific to the structure of Wikimedia sites and can create confusions for KBQA systems. It is thus excluded.

Questions on complex SPARQL code: This discussion, titled "Scatterplot query" in March of 2021, is an example where the original requester comes in with a SPARQL that is complicated whose meaning is already difficult to acurately express in English. The final SPARQL from the conversation (hyperlinked in " a bit more baroque, with axes") only adds to the complexity.

Queries obscured by optimizations: This discussion, titled "Section "P31wdt:P279 wd:Q16917 in wdt:P131wdt:P131 wd:Q25 (query optimization)", involves only the refactoring of a query into using two sub-queries which are then joined together to avoid timing outs. The two queries are semantically equivalent. Due to reproducibility challenges from this refactoring, this discussion is excluded.

Formatting clauses: The discussion, titled "preferred format for id" in Feburary of 2022, only contains discussions on differnt ways for string processing in SPARQL and is thus excluded. The discussion, titled "Custom link formatting in WDQS results" in June of 2018, only contains results in a SPARQL that focuses on converting string formats after minimal use of Wikidata (fetching the sitelinks of one item) and is thus excluded.

C.6 Analysis of the Popularity of Queries

Following the method in Semnani et al. (2023) and Mallen et al. (2023), we estimate and report the popularity of queries in the SPINACH dataset. Specifically, for each unique entity in the gold SPARQL queries of the SPINACH dataset, we obtain the number of page views for its English Wikipedia page in the past year (July 2023 - June 2024), counting those without an English Wikipedia page as 0. We show in Table 9 a distribution of the page views, sorted from higher to lower page views, where the gold entity is shown in bold. As shown here, the dataset contains a wide range of entities, from popular ones like **Norway** at 3,606,300 page views per year to tail entities like **Administrative divisions of Moscow** at 44 page views per year.

D System and Evaluation Details

Unless otherwise specified, LLMs are used with greedy decoding, i.e. with $temperature = 0$, with the exception of SPINACH agent's policy prompt, which is run with $temperature = 1$ and nucleus sampling (Holtzman et al., 2019) with $p = 0.9$

The ToG agent by default builds a local version

Percentile	Page Views Per Year	Example question in the SPINACH dev set
10th	3,499,391	Who are citizens of **Norway** that have held a position as a teacher or a subclass of teacher, along with their labels, dates of birth (if available), and Norwegian historical register of persons IDs (if available)?
25th	873,067	What are the **Nazi concentration camps** or subcamps? Include the following way of finding such a camp: (1) those that are classified as a **Nazi concentration camp**, (2) those that are considered the **subcamps**, and (3) **subsidiaries of Nazi concentration camps**. For each one, also find its coordinates (e.g. literals such as Point(9.1978 49.1686)). Each pair of camp and coordinate should only appear once.
50th	131,131	What are the religious buildings located in the region of **Molise, Italy**?
75th	44	What are the current municipal districts, urban districts in Russia, or **administrative districts of Moscow** located within **Perm Krai**?
90th	0	Who are the members of the historic **Lower House** and Upper House **of the Parliament of Sweden**, whose party's Swedish label contains the string "vilde" in their party name? For each person, also return the associated party, and the start (if available) and end (if available) times of the party.

Table 9: Distribution of page views for the entity's English Wikipedia page in July 2023 - June 2024 for each entity in SPINACH's gold SPARQL queries.

of Wikidata using the `simple-wikidata-db` library[10]. However, building this index from scratch is extremely computationally expensive. Sun et al. (2024b) report that they need to deploy the Wikidata dump across five AWS EC2 instances, each consisting of a 768GB machine with 48 cores. Fahl et al. (2022) reported various other methods of hosting Wikidata locally. We attempted using Qlever as the SPARQL Engine but failed to do so on the latest Wikidata dump. For the evaluation of ToG, we re-implemented the same logic using dynamic Wikidata API calls.

We adopted the same hyperparameters for ToG as used in the original paper. For the experiment in Section 5.3, we used D=3 and T=3 as the hyperparameters for graph exploration.

The GPT-4 and ToG systems return results in strings of the entities instead of entity IDs; we convert the gold SPARQL output to only contain their English labels and calculate EM and F1 metrics.

D.1 Token Distribution for the SPINACH Agent

Table 10 and Table 11 show the distribution of input and output tokens needed to answer questions from the SPINACH dev set, respectively. On average, running SPINACH agent on each question costs \$0.1253 using GPT-4o.

E Prompts used in the SPINACH system

The policy prompt and the prompt used to prune the output of **`get_wikidata_entry()`** are shown in Table 12 and Table 13, repsectively.

# of Tokens	Percentage
4912.50 - 10330.50	16.23%
10330.50 - 15748.50	24.03%
15748.50 - 21166.50	18.18%
21166.50 - 26584.50	8.44%
26584.50 - 32002.50	6.49%
32002.50 - 37420.50	5.84%
37420.50 - 42838.50	10.39%
42838.50 - 48256.50	7.14%
48256.50 - 53674.50	1.30%
53674.50 - 59092.50	1.95%

Table 10: Distribution of total input tokens needed to answer a question (including all intermediate steps) from the SPINACH dev set.

# of Tokens	Percentage
164.00 - 445.30	33.77%
445.30 - 726.60	27.92%
726.60 - 1007.90	14.29%
1007.90 - 1289.20	11.04%
1289.20 - 1570.50	6.49%
1570.50 - 1851.80	3.90%
1851.80 - 2133.10	1.30%
2133.10 - 2414.40	0.65%
2414.40 - 2695.70	0.00%
2695.70 - 2977.00	0.65%

Table 11: Distribution of total output tokens needed to answer a question (including all intermediate steps) from the SPINACH dev set.

F Full Example Trace of the SPINACH Agent

Listing 1 contains a full sequence of thoughts, actions and observations of the SPINACH agent for

---

[10] Github: neelguha/simple-wikidata-db

```
# instruction
Your task is to write a Wikidata SPARQL query to answer the given question. Follow a step-by-step process:

1. Start by constructing very simple fragments of the SPARQL query.
2. Execute each fragment to verify its correctness. Adjust as needed based on your the observations.
3. Confirm all your assumptions about the structure of Wikidata before proceeding.
4. Gradually build the complete SPARQL query by adding one piece at a time.
5. Do NOT repeat the same action, as the results will be the same.
6. The question is guaranteed to have an answer in Wikidata, so continue until you find it.
7. If the user is asking a True/False question with only one answer, use ASK WHERE to fetch a True/False answer at the very
     end.
8. In the final SPARQL projections, do not only ask for labels. Ask for the actual entities whenever needed (e.g. instead of
     doing `SELECT xLabel`, do `SELECT x`).
9. If the final result was contained in last round's `get_wikidata_entry` and you are ready to stop, use `execute_sparql` and
     generate a SPARQL to retrieve that results.

Form exactly one "Thought" and perform exactly one "Action", then wait for the "Observation".

Possible actions are:

- get_wikidata_entry(QID): Retrieves all outgoing edges (linked entities, properties, and qualifiers) of a specified Wikidata
     entity using its QID.
- search_wikidata(string): Searches Wikidata for entities or properties matching the given string.
- get_property_examples(PID): Provides a few examples demonstrating the use of the specified property (PID) in Wikidata.
- execute_sparql(SPARQL): Runs a SPARQL query on Wikidata and returns a truncated result set for brevity.
- stop(): Marks the last executed SPARQL query as the final answer and ends the process.


# input
Question: {{ question }}

{% if action_history %}
{% for i in range(0, action_history|length) %}

{{ action_history[i] }}
{% endfor %}
{% endif %}

Output one "Thought" and one "Action":
```

Table 12: The policy prompt of the SPINACH agent.

the given input question.

```
# instruction
At each turn, you are given a Wikidata entry and a question.
You want to write a SPARQL query that answers the question.
As the first step, remove the parts of the Wikidata entry that could not be potentially helpful when writing the SPARQL.
The output should be a json object containing part of the Wikidata entry.

# few-shot example 1, input
Wikidata entry for OneRepublic (Q1438730, 'OneRepublic' is an American pop rock band formed in Colorado Springs, Colorado, in 2002):
{
  "instance of (P31)": "musical group (Q215380)",
  ...
  "social media followers (P8687)": {
    "3134158": {
      "Qualifiers": [
        {
          "point in time (P585)": "4 February 2023"
        }
      ]
    },
    "3276596": {
      "Qualifiers": [
        {
          "point in time (P585)": "6 January 2021"
        }
      ]
    },
    "3178896": {
      "Qualifiers": [
        {
          "point in time (P585)": "2 March 2022"
        }
      ]
    },
    "3720919": {
      "Qualifiers": [
        {
          "point in time (P585)": "10 May 2018"
        }
      ]
    }
  }
}

Question: "What is the title of the second single on OneRepublic's third album Native?"

# few-shot example 1, output
{
  "instance of (P31)": "musical group (Q215380)",
  "record label (P264)": "Columbia Records (Q183387)",
  "discography (P358)": "OneRepublic discography (Q935670)",
}



# few-shot example 2, input
Wikidata entry for Barack Obama Sr. (Q649593, Barack Obama Sr. was an economist and the father of Barack Obama jr.):
{
  "place of birth (P19)": {
    "Nyang'oma Kogelo (Q1027168)": {
      "Qualifiers": [
        {
          "country (P17)": "Kenya Colony"
        }
      ]
    }
  },
  ...
}

Question: "where did barack obama sr. attend school?"

# few-shot example 2, output
{
  "place of birth (P19)": {
    "Nyang'oma Kogelo (Q1027168)": {
      "Qualifiers": [
        {
          "country (P17)": "Kenya Colony"
        }
      ]
    }
  },
  "place of death (P20)": "Nairobi (Q3870)",
 ...
}


# input
Wikidata entry for "{{ entity_and_description }}":
{{ outgoing_edges }}

Question: "{{ question }}"
```

Table 13: The 2-shot prompt used to prune the output of `get_wikidata_entry()` action. Here we omit parts of the few-shot example inputs for space.

```
"""
Question: SteplanI hav
"""

Thought: First, I need to identify the properties and entities related to
    people affiliated with the University of Washington School of Music,
    their educational background, and the musical instruments they play.
    This will help in formulating the final SPARQL query.
>> search_wikidata("University of Washington School of Music")
Observation: University of Washington School of Music (Q98035717): school of
    music

Thought: Now that I have identified the entity for the University of
    Washington School of Music, I need to find the properties that can help
    me identify people affiliated with it and the properties related to
    their education and musical instruments.
>> get_wikidata_entry("Q98035717")
Observation:
  {
    instance of (P31): {
      faculty (Q180958): {
        Qualifiers: [
          {
            of (P642): University of Washington (Q219563)
          }
        ]
      },
      music school (Q1021290): {}
    },
    field of work (P101): [
      higher education (Q136822),
      postgraduate education (Q141212),
      musicology (Q164204),
      music (Q638),
      research (Q42240),
      ethnomusicology (Q208365)
    ],
    located in the administrative territorial entity (P131): Seattle (Q5083),
    location (P276): Music Building (Q98690890),
    has subsidiary (P355): University of Washington Laboratory for Music
        Cognition, Culture & Learning (Q101157561),
    part of (P361): University of Washington College of Arts and Sciences
        (Q7896566),
    has part(s) (P527): University of Washington Laboratory for Music
        Cognition, Culture & Learning (Q101157561),
    parent organization (P749): University of Washington College of Arts and
        Sciences (Q7896566)
  }

Thought: To find people affiliated with the University of Washington School
    of Music, I need to look for properties that define such an affiliation.
    Additionally, I need to identify properties related to their education
    at the University of Washington and the musical instruments they play. I
    will start by searching for the property that links people to their
    affiliations.
>> search_wikidata("affiliation")
Observation:
  - affiliation (Q46135267)
    connection, formally or informally, with an organization or intellectual
        movement
  - filiation (Q5861944)
  system for recognizing the lineage or kinship of a person in anthropology
  - affiliation (Q2620373)
    archaic term for the legal establishment of paternity
  - affiliated entity (Q69467106)
    inverse property label item for P1416
  - Affiliation (Q93774359)
    scholarly article
  - Affiliations Committee (Q46815761)
    Wikimedia community-run committee
  - Wikimedia Affiliations Committee wiki (Q33120941)
    Wikimedia private wiki
  - Affiliation and participation (Q107433952)
    master's thesis of Hannah Marie Mayfield
  - affiliation (P1416)
    organization that a person or organization is affiliated with (not
        necessarily member of or employed by)
    Data Type: wikibase-item
  - affiliation string (P6424)
    qualifier to provide the published string form of affiliation attached
        to an author
    Data Type: string
  - Olympedia affiliations ID (P9447)
    identifier for the affiliation of athletes in Olympedia athlete database
    Data Type: external-id (Represents an identifier used in an external
        system.)

Thought: The property `affiliation (P1416)' seems to be the most relevant
    for linking people to the University of Washington School of Music.
    Next, I need to identify the property for their education at the
    University of Washington and the property that links people to the
    musical instruments they play.
>> search_wikidata("educated at")
Observation:
  - educated at (P69)
    educational institution attended by subject
    Data Type: wikibase-item

Thought: I have identified the property `educated at (P69)' to link people
    to the University of Washington. Next, I need to find the property that
    links people to the musical instruments they play.
>> search_wikidata("musical instrument")
Observation:
  - musical instrument (Q34379)
    device created or adapted to make musical sounds
  - heraldic musical instrument (Q56877088)
    category of heraldic charges
  - Musical instrument (Q102413357)
    Oil of canvas by Alla Grigoryan
  - Musical Instrument (Q102413736)
    painting by Musical instrument
  - Musical instrument (Q86150836)
    sculpture in the church Brother Klaus in the city of Bern, Switzerland
  - Musical Instrument (Q80002888)
    musical instrument by unknown artist (1929.322)
  - Musical instrument (Q123046258)
    US patent 11482197
  - musical instrument maker (Q1955150)
    manufacturer of musical instruments
  - instrument (P1303)
    musical instrument that a person plays or teaches or used in a music
        occupation
    Data Type: wikibase-item
  - MIMO instrument ID (P3763)
    identifier for a musical instrument, on the instrument keyword thesaurus
        from Musical Instruments Museums Online
    Data Type: external-id (Represents an identifier used in an external
        system.)

Thought: I have identified the relevant properties: `affiliation (P1416)' to
    link people to the University of Washington School of Music, `educated
    at (P69)' to link people to the University of Washington, and `musical
    instrument (P1303)' to identify the instruments they play. I will now
    construct a SPARQL query that combines these properties to get the
    required information.
>> execute_sparql("""
    SELECT ?instrument (COUNT(*) AS ?count) WHERE {
      ?person wdt:P69 wd:Q1063349;
              wdt:P1416 wd:Q98035717;
              wdt:P1303 ?instrument.
    }
    GROUP BY ?instrument
    """
)
Observation:
  Query returned empty result.

Thought: It appears that the query returned an empty result. This could be
    due to several reasons: either there are no people matching all
    criteria, or the properties might not be used exactly as I assumed. To
    ensure accuracy, I will individually verify if there are people
    affiliated with the University of Washington School of Music who have
    also been educated at the University of Washington.
>> execute_sparql(
    """
    SELECT ?person WHERE {
      ?person wdt:P69 wd:Q1063349;
        wdt:P1416 wd:Q98035717.
    }
    LIMIT 10
    """
)
Observation: Query returned empty result.

Thought: The absence of results could indicate that the properties or the
    specific entities used in the query might not be correct or commonly
    utilized in this context. I will check if there are people who have been
    educated at the University of Washington (Q1063349) and then see if I
    can refine the affiliation aspect.
>> execute_sparql(
    """
    SELECT ?person WHERE { ?person wdt:P69 wd:Q1063349. }
    LIMIT 10
    """
)
Observation: Query returned empty result.

Thought: Since my previous queries returned empty results, I need to verify
    if there are any individuals in Wikidata who have been educated at the
    University of Washington (Q1063349). This will help me confirm if I am
    using the correct properties and entities.
>> get_wikidata_entry("Q1063349")
Observation:
  {
    country (P17): Russia (Q159),
    instance of (P31): hamlet (Q5084),
    located in the administrative territorial entity (P131): Tavtimanovsky
        selsoviet (Q4449505),
    population (P1082): {
      11: {
        Qualifiers: [
          {
            point in time (P585): 1 January 2010,
            determination method (P459): census (Q39825)
          },
          {
            determination method (P459): census (Q39825),
            point in time (P585): 1 January 2002
          }
        ]
      },
      9: {
        Qualifiers: [
          {
            point in time (P585): 1 January 2009,
            determination method (P459): census (Q39825)
          }
        ]
      }
    }
  }

Thought: The query results indicate that there may be an issue with the QID
    I used for the University of Washington (Q1063349). It seems to
    correspond to a location in Russia rather than the university. I should
    search for the correct QID for the University of Washington.
>> search_wikidata("University of Washington")
Observation:
  - University of Washington (Q219563)
    public research university in Seattle, Washington, United States
  - University of Washington Collection (NAID 931) (Q59502962)
    collection in the National Archives and Records Administration's holdings
  - University of Washington Department of Chemistry (Q97958839)
    academic department at the University of Washington
  - University of Washington Department of Architecture (Q98844905)
    academic department at the University of Washington
  - University of Washington Department of Civil and Environmental
      Engineering (Q97990078)
    academic department at the University of Washington
  - University of Washington Department of Chemical Engineering (Q99196105)
    academic department at the University of Washington
  - University of Washington Paul G. Allen School of Computer Science &
      Engineering (Q98186807)
    school of the University of Washington
  - University of Washington School of Music (Q98035717)
    school of music

Thought: The correct QID for the University of Washington is Q219563. I
    should now update my SPARQL query to use this correct QID for
    identifying people who have been educated at the University of
    Washington and who are affiliated with the University of Washington
    School of Music.
>> execute_sparql(
    """
    SELECT ?person WHERE {
      ?person wdt:P69 wd:Q219563;
        wdt:P1416 wd:Q98035717.
    }
    LIMIT 10
    """
)
Observation:
  | Person         |
  |---------------|
  | Q107692821    |
  | Q107705263    |
  | Q107706107    |
  | Q107707997    |
  | Q107708199    |
  | Q107708456    |
  | Q107720948    |
  | Q107773759    |
  | Q107782800    |
  | Q107789859    |

Thought: Now that I have identified individuals who have been educated at
    the University of Washington (Q219563) and are affiliated with the
    University of Washington School of Music (Q98035717), I will update my
    SPARQL query to find the musical instruments played by these individuals
    and count the occurrences of each instrument.
>> execute_sparql(
    """
    SELECT ?instrument ?instrumentLabel (COUNT(?student) AS ?count) WHERE {
      ?student wdt:P1303 ?instrument;
        wdt:P1416 wd:Q98035717;
        wdt:P69 wd:Q219563.
      SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
    }
    GROUP BY ?instrument ?instrumentLabel
    """
```

```
)
Observation:
  | instrument   | instrumentLabel | count |
  |--------------|-----------------|-------|
  | Q5994        | piano           | 99    |
  | Q1467960     | mbira           | 2     |
  | Q8350        | trombone        | 11    |
  | Q8338        | trumpet         | 8     |
  | Q17172850    | voice           | 32    |
  | ...          | ...             | ...   |
  | Q302497      | mandolin        | 1     |
  | Q187851      | recorder        | 1     |
  | Q185041      | cor anglais     | 1     |
  | Q83509       | piccolo         | 1     |

Thought: The query successfully returned the musical instruments played by
    individuals who are affiliated with the University of Washington School
    of Music and have been educated at the University of Washington, along
    with the counts of each instrument. I will mark this query as complete.
>> stop()
```

Listing 1: Full version of the example in Figure 2