

814 A DB-style Input and Output Format

815 A.1 GrailQA

816 Schema Filtering Stage:

```
INPUT:
Identify the exact schema component from the given schema that would correctly answer the
↪ following question.

schema:

book.series_editor : book_edition_series_edited | book.book_edition : book_edition_series,
↪ place_of_publication | book.book_edition_series : editions_in_this_series, series_editor

question:

a people's history of christianity was edited by what series editor?

OUTPUT:
book.series_editor : book_edition_series_edited
```

817

818 Query Building Stage:

```
INPUT:
Generate an s-expression that can be used to find the answer to the following question using the
↪ knowledge graph schema items provided.

schema:

a people's history of christianity: m.012bphrj | book.series_editor : book_edition_series_edited |
↪ book.book_edition : book_edition_series, place_of_publication | book.book_edition_series :
↪ editions_in_this_series, series_editor

question:

a people's history of christianity was edited by what series editor?

OUTPUT:
(AND book.series_editor (JOIN book.series_editor.book_edition_series_edited m.012bphrj))
```

819

820 A.2 Spider

821 Schema Filtering Stage:

```
INPUT:
Identify the exact schema component from the given schema that would correctly answer the
↪ following question.

schema:

department : department_id , name , creation , ranking , budget_in_billions , num_employees | head
↪ : head_id , name , born_state , age | management : department_id , head_id , temporary_acting

question:

How many heads of the departments are older than 56 ?

OUTPUT:
head : age
```

822

823 Query Building Stage:

INPUT:
Your task is to convert the following question to an SQL query using the following database
↳ schema.

schema:
department : department_id , name , creation , ranking , budget_in_billions , num_employees | head
↳ : head_id , name , born_state , age | management : department_id , head_id , temporary_acting

question:
How many heads of the departments are older than 56 ?

OUTPUT:
select count(*) from head where age > 56

824

825 A.3 CWQ

826 Schema Filtering Stage:

INPUT:
Identify the exact schema component from the given schema that would correctly answer the
↳ following question.

schema:

type.object : type | location.location : time_zones, containedby, contains, people_born_here |
↳ common.topic : notable_types | geography.river : mouth | travel.tourist_attraction :
↳ near_travel_destination | kg.object_profile : prominent_type |
↳ education.educational_institution : colors | location.country : languages_spoken,
↳ administrative_divisions

question:What language is spoken in the country that has Southern Peninsular?

OUTPUT:
location.country : languages_spoken, administrative_divisions

827

828 Query Building Stage:

INPUT:
Given the list of schema items, write an SPARQL query that can be used to find the answer to the
↳ following question.

schema:
Southern Peninsula: m.08kmfj | type.object : type | location.location : time_zones, containedby,
↳ contains, people_born_here | common.topic : notable_types | geography.river : mouth |
↳ travel.tourist_attraction : near_travel_destination | kg.object_profile : prominent_type |
↳ education.educational_institution : colors | location.country : languages_spoken,
↳ administrative_divisions

question:
What language is spoken in the country that has Southern Peninsular?

OUTPUT:
PREFIX ns: <http://rdf.freebase.com/ns/>SELECT DISTINCT ?xWHERE {?c
↳ ns:location.country.administrative_divisions ns:m.08kmfj .?c
↳ ns:location.country.languages_spoken ?x .}

829

830 A.4 MTOP

831 Schema Filtering Stage:

INPUT:

Identify the exact schema component from the given schema that would correctly answer the
→ following question.

schema:

```
in:get : message, weather, alarm, info_recipes, stories_news, reminder, recipes, event, call_time,  
→ life_event, info_contact, contact, timer, reminder_date_time, age, sunrise, employer,  
→ education_time, job, availability, category_event, call, employment_time, call_contact,  
→ location, track_info_music, sunset, mutual_friends, undergrad, reminder_location,  
→ attendee_event, message_contact, reminder_amount, date_time_event, details_news,  
→ education_degree, major, contact_method, life_event_time, lyrics_music, airquality, language,  
→ gender, group | in:send_message | in:set : unavailable, rsvp_yes, available,  
→ default_provider_music, rsvp_interested, default_provider_calling, rsvp_no | in:delete :  
→ reminder, alarm, timer, playlist_music | in:create : alarm, reminder, call, playlist_music,  
→ timer | in:question : news, music | in:play : music, media | in:end_call | in:ignore_call |  
→ in:update_call | in:update_reminder_date_time | in:pause : music, timer | in:answer_call |  
→ in:snooze_alarm | in:update_reminder_todo | in:is_true_recipes | in:remove_from_playlist_music  
→ | in:add : time_timer, to_playlist_music | in:share_event | in:prefer | in:start_shuffle_music  
→ | in:silence_alarm | in:switch_call | in:subtract_time_timer | in:update_timer |  
→ in:previous_track_music | in:hold_call | in:skip_track_music | in:update_method_call |  
→ in:update_alarm | in:like_music | in:restart_timer | in:resume : timer, call, music |  
→ in:merge_call | in:replay_music | in:loop_music | in:stop : music, shuffle_music |  
→ in:unloop_music | in:update_reminder_location | in:cancel : message, call | in:update_reminder  
→ | in:rewind_music | in:repeat : all_music, all_off_music | in:fast_forward_music |  
→ in:dislike_music | in:disprefer | in:help_reminder | in:follow_music
```

question:

Has Angelika Kratzer video messaged me ?

OUTPUT:

IN:GET : MESSAGE

832

833 Query Building Stage:

INPUT:

Convert the following natural language query to an API call in Task Oriented Parsing (TOP)
→ representation using the following API specification.

API specification:

```
IN:GET : MESSAGE, WEATHER, ALARM, INFO_RECIPES, STORIES_NEWS, REMINDER, RECIPES, EVENT, CALL_TIME,  
→ LIFE_EVENT, INFO_CONTACT, CONTACT, TIMER, REMINDER_DATE_TIME, AGE, SUNRISE, EMPLOYER,  
→ EDUCATION_TIME, JOB, AVAILABILITY, CATEGORY_EVENT, CALL, EMPLOYMENT_TIME, CALL_CONTACT,  
→ LOCATION, TRACK_INFO_MUSIC, SUNSET, MUTUAL_FRIENDS, UNDERGRAD, REMINDER_LOCATION,  
→ ATTENDEE_EVENT, MESSAGE_CONTACT, REMINDER_AMOUNT, DATE_TIME_EVENT, DETAILS_NEWS,  
→ EDUCATION_DEGREE, MAJOR, CONTACT_METHOD, LIFE_EVENT_TIME, LYRICS_MUSIC, AIRQUALITY, LANGUAGE,  
→ GENDER, GROUP | IN:SEND_MESSAGE | IN:SET : UNAVAILABLE, RSVP_YES, AVAILABLE,  
→ DEFAULT_PROVIDER_MUSIC, RSVP_INTERESTED, DEFAULT_PROVIDER_CALLING, RSVP_NO | IN:DELETE :  
→ REMINDER, ALARM, TIMER, PLAYLIST_MUSIC | IN:CREATE : ALARM, REMINDER, CALL, PLAYLIST_MUSIC,  
→ TIMER | IN:QUESTION : NEWS, MUSIC | IN:PLAY : MUSIC, MEDIA | IN:END_CALL | IN:IGNORE_CALL |  
→ IN:UPDATE_CALL | IN:UPDATE_REMINDER_DATE_TIME | IN:PAUSE : MUSIC, TIMER | IN:ANSWER_CALL |  
→ IN:SNOOZE_ALARM | IN:UPDATE_REMINDER_TODO | IN:IS_TRUE_RECIPES | IN:REMOVE_FROM_PLAYLIST_MUSIC  
→ | IN:ADD : TIME_TIMER, TO_PLAYLIST_MUSIC | IN:SHARE_EVENT | IN:PREFER | IN:START_SHUFFLE_MUSIC  
→ | IN:SILENCE_ALARM | IN:SWITCH_CALL | IN:SUBTRACT_TIME_TIMER | IN:UPDATE_TIMER |  
→ IN:PREVIOUS_TRACK_MUSIC | IN:HOLD_CALL | IN:SKIP_TRACK_MUSIC | IN:UPDATE_METHOD_CALL |  
→ IN:UPDATE_ALARM | IN:LIKE_MUSIC | IN:RESTART_TIMER | IN:RESUME : TIMER, CALL, MUSIC |  
→ IN:MERGE_CALL | IN:REPLAY_MUSIC | IN:LOOP_MUSIC | IN:STOP : MUSIC, SHUFFLE_MUSIC |  
→ IN:UNLOOP_MUSIC | IN:UPDATE_REMINDER_LOCATION | IN:CANCEL : MESSAGE, CALL | IN:UPDATE_REMINDER  
→ | IN:REWIND_MUSIC | IN:REPEAT : ALL_MUSIC, ALL_OFF_MUSIC | IN:FAST_FORWARD_MUSIC |  
→ IN:DISLIKE_MUSIC | IN:DISPREFER | IN:HELP_REMINDER | IN:FOLLOW_MUSIC
```

question:

Has Angelika Kratzer video messaged me ?

OUTPUT:

[IN:GET_MESSAGE [SL:CONTACT Angelika Kratzer] [SL:TYPE_CONTENT video] [SL:RECIPIENT me]]

834

835 B Prompts for Synthesizing Query Structures

836 B.1 Prompt for Pseudo Structure Synthesis

You are an expert in logical query expression synthesis. Your task is to merge two simple query
↪ skeletons into a single, more complex skeleton that logically integrates their structure and
↪ meaning. The result should preserve the semantics of both inputs and follow the same
↪ structural style and syntax. Output only the composed skeleton. Do not include any explanation
↪ or additional text.

Simple skeleton 1:\{example1\}

Simple skeleton 2:\{example2\}

Composed skeleton:

837

838 B.2 Prompt for Pseudo Question Generation

You are an AI assistant that specializes in transforming structured query statements into fluent,
↪ contextually accurate natural language questions. Your task is to read a given formal query
↪ and its associated schema context, and then write a corresponding question in clear, natural
↪ language that would retrieve the correct answer if the formal query were executed on a
↪ database.

question:
{query}

schema:
{schema}

Follow these steps carefully:

1. Understand the Query Semantics:
Analyze the query logic, including the target variables, the filtering conditions, the joins, and
↪ the structure of the query. Infer what specific information the query is intended to retrieve.

2. Incorporate Schema Semantics:
Use the provided schema elements (such as table names, entity types, or relation labels) to guide
↪ the terminology and phrasing in the question. Map schema components to human-interpretable
↪ phrases as necessary.

3. Avoid Direct Translation:
Do not simply convert the query structure into a rigid, mechanical form. Instead, aim for a fluent,
↪ natural-sounding question that a human might ask when seeking the same information.

4. Preserve Answer Equivalence:
Ensure that the generated question is logically equivalent to the formal query in terms of the
↪ expected answer. The question should not broaden, narrow, or alter the scope of the query.

5. Maintain Clarity and Brevity:
The question should be unambiguous and concise, while preserving all the critical information
↪ needed to align with the query.

6. Output Format Constraint:
Only output the final question, without any explanatory text, metadata, or formatting symbols. The
↪ output should consist solely of a single fluent question in English.

839

840 B.3 Examples of Structure Synthesis

Structure 1:
(ARGMAX [T1] [C1])

Structure 2:
(AND [T1] (JOIN [C1] [E1]))

Synthetic Structure:
(ARGMAX (AND [T1] (JOIN [C1] [E1])) [C2])

841

```

Structure 1:
SELECT [C1] FROM [T1] WHERE [C2] = [V1];

Structure 2:
SELECT [C1] FROM [T] WHERE [C3] IN [V1];

Synthetic Structure:
SELECT * FROM [T1] WHERE [C1] IN (SELECT [C2] FROM [T2] WHERE [C3] = [V1]);

```

842

843 C Hyperparameter settings

844 Our hyperparameter settings are as follows:

Table 3: Hyperparameter Configuration for Experiments

Hyperparameter	Value
GPU	NVIDIA RTX 4090
PEFT Module	LoRA [19]
LoRA Target	gate_proj, down_proj, up_proj, q_proj, v_proj, k_proj, o_proj
LoRA Rank	8
LoRA α	16
Memory Size per Task	$ \mathcal{M}_a^k = 5, \mathcal{M}_b^k = 5$
Real to Pseudo Memory Ratio	4 : 1
Batch Size	12
Global Batch Size	32
Learning Rate	5×10^{-5}
Optimizer	Adam
Training Epochs	5
Maximum Input Length	1024
Maximum Output Length	256
Parameter T in Algorithm 1	5