DialSQL: Dialogue Based Structured Query Generation

Подготовил:

Пугачев Александр, 151

Natural language interfaces to databases

What are the countries that joined the NATO before 2004?

SELECT country WHERE date of join < 2004

WikiSQL

Question: How many CFL teams are from York College?

SQL query: SELECT COUNT "CFL Team"
FROM "CFLDraft"
WHERE "College = YORK"

Table:

Pick#	CFL Team	Player	Position	College
27	Hamilton Tiger-Cats	Connor Healy	DB	Wilfrid Laurier
28	Calgary Stampeders	Anthony Forgone	OL	York
...	...	...	...	...

Dialogue Simulation

What are the countries that joined the NATO before 2004?

Step 1: Initial Query Generation

SELECT COUNT country WHERE date of join = 2004

AGG (1, 2, COUNT)

SELECT country WHERE date of join = 2004

WHERE_CHANGE (6, 7, =)

SELECT country WHERE date of join < 2004

Step 2: Dialogue Generation

System: validate_agg (1, 2) – offer (no_agg)
User: negate() – select()

System: validate_where_change (6, 7) – offer (<)
User: negate() – select()

System: confirm()
User: confirm()

User Question	What are the countries that joined the NATO before 2004?
True SQL Query	SELECT country WHERE date of join < 2004
Candidate SQL Query	SELECT COUNT country WHERE date of join = 2004

Are you asking about **total number of results**?

validate_agg(1, 2)

No.

negate()

Select one option:
a) Average number of results.
b) All the results.

offer_agg(average, no_agg)

It is b.

select(2)

Confirm if **date of join is 2004?**

validate_where_change(6, 11)

That's wrong.

negate()

Select one option:
a) Date of join is before 2004.
b) Date of invitation is before 2004.

offer_where(date of j. < 2004, date of inv. < 2004)

It is a.

select(1)

Interactive Query Generation

Question $Q = \{q_1, q_2, \ldots, q_N\}$

Table column names $T = \{T_1, T_2, \ldots, T_k\}$

Initial SQL query U generated by black box

Turn t : (S_t, R_t) with H_t

S_t - system response

R_t - user response

H_t - dialogue history (list of previous turns)

Interactive Query Generation

System response

$$S_t = (c, s, C)$$

c - error category

s - error span

C - set of candidate choices

User response

Is represented by:

affirmation or negation

index c' to identify choice

Error categories

Error Category	Meaning in the dialogue
validate_sel	Validate the select clause
validate_agg	Validate the aggregation operator
validate_where_changed	Validate if a segment of a where clause is incorrect
validate_where_removed	Validate if a new where clause is needed
validate_where_added	Validate if an incorrect where clause exists

Interactive Query Generation Task

At each turn t :

- predict error category c
- extract error span s from query U
- decode set of candidate choices C

Synthetic Query

Given a ground truth query:

- Randomly select an error category
- Extract a related span from the current query
- Randomly generate a valid choice for chosen span
- Update the query by replacing span with choice

DialSQL

Given a (Q, T, U) triplet:

Encode Q , each column name $T_i \in T$, and query U with GloVe and RNN

At each turn t :

1) Encode dialogue history and predict error category
2) Decode position of error span based on encoded query and error category
3) Decode a list of choices to offer the user based on error category and error span

Encoding

Question, Column names and Query

- Encode (Q, T, U) with RNN (Enc)
- Produce hidden states o and last hidden state h for each object

System and User turns

- Encode (S_t, R_t) using embedding lookup through GloVe

Encoding Dialogue History

$$h_0^{D_1} = h^Q$$

$$o_t^{D_1},\ g_t^{D_1} = Enc([E_c, E_a])$$

$$h_t^{D_1} = \left[Attn(g_t^{D_1}, H^T), o_t^{D_1}\right]$$

$$Attn(h, O) = \sum softmax(tanh(hWO)) * O$$

$[.]$ - vector concatenation

E_c - error category encoding

E_a - user turn encoding

$h_0^{D_1}$ - initial hidden state

$h_t^{D_1}$ - current hidden state

Predicting Error Category

$$c_t = tanh\left(Lin\left(\left[Attn(h_t^{D_1}, O^U), h_t^D\right]\right)\right)$$

$$l_t = softmax(c_t \cdot E(C))$$

Lin - linear transformation

$E(C)$ - matrix with error category embeddings

l_t - probability distribution over categories

Decoding Error Span

$$p_i = softmax\left(tanh(h_t^{D_2} L_1 H^U)\right)$$

$$c_i = \sum p_i * H^U$$

$$\hat{p_j} = softmax\left(tanh([h_t^{D_1}, c_i] L_2 H^U)\right)$$

p_i - probability of start position over the i-th query token

$\hat{p_j}$ - probability of end position over the jth query token

Decoding Candidate Choices

Column choice

$$h = Attn(Lin(o_{i-1}^U, o_j^U, E_c]), H^T)$$

$$s_{col} = u^T * tanh(Lin([H^T, h]))$$

o_{i-1}^U - output vector of the query encoder preceding the start position

o_j^U - output of query encoder at the end position

Aggregation choice

$$s_{agg} = v^T * tanh(Lin(Attn(e, H^Q)))$$

e - encoding of aggregation function

Agg: (MIN, MAX, COUNT, NO AGGREGATION)

Decoding Candidate Choices

Where condition choice

$$s_{op} = w^T * tanh(Lin(Attn(e, H^Q)))$$

$$s_{st} = Attn(e, H^Q)$$

$$s_{end} = Attn([e, h_{st}, H^Q])$$

e - encoding of operator

h_{st} - context vector generated from Attention

Conditions: (<, >, =)

Evaluation Setup and Metrics

- Query-match accuracy
- Dialogue length
- Question complexity

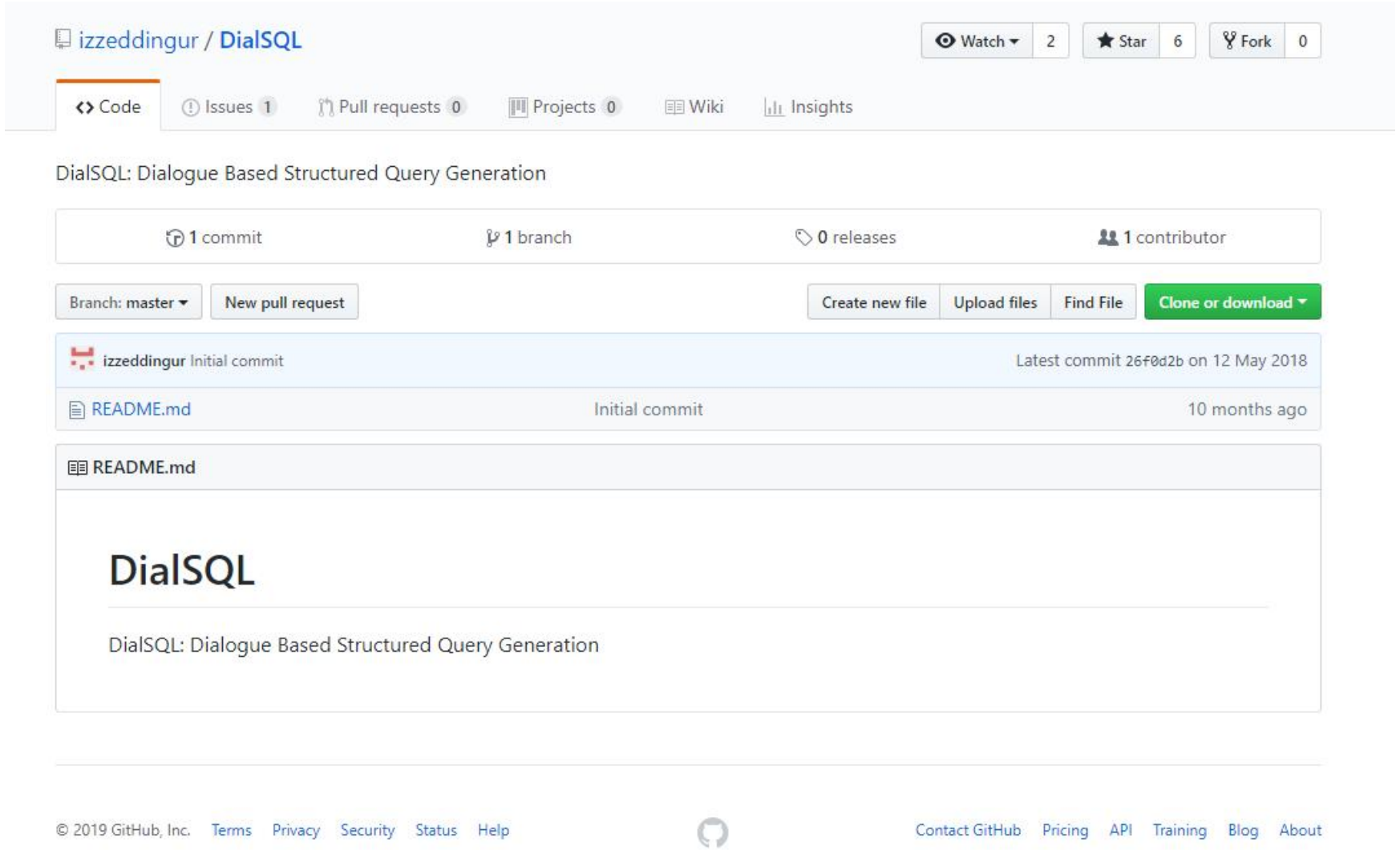
Evaluation

Model	QM-Dev	QM-Test
Seq2SQL	53.5	51.6
SQLNet	63.2	61.3
Seq2SQL + DialSQL	62.2 ↑	61.0 ↑
SQLNet + DialSQL	70.9 ↑	69.0 ↑

Query-match accuracy on WikiSQL dataset

Human evaluation

Model	Query-match Accuracy
SQLNet	58
SQLNet + DialSQL + User Simulation	75
SQLNet + DialSQL + Real Users	65


izzeddingur / DialSQL
Watch 2
Star 6
Fork 0
Code
Issues 1
Pull requests 0
Projects 0
Wiki
Insights
DialSQL: Dialogue Based Structured Query Generation
1 commit
1 branch
0 releases
1 contributor
Branch: master
New pull request
Create new file
Upload files
Find File
Clone or download
izzeddingur Initial commit
Latest commit 26f0d2b on 12 May 2018
README.md
Initial commit
10 months ago
README.md
DialSQL
DialSQL: Dialogue Based Structured Query Generation
© 2019 GitHub, Inc.
Terms
Privacy
Security
Status
Help
Contact GitHub
Pricing
API
Training
Blog
About

Bibliography

- Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. 2018. “DialSQL: Dialogue Based Structured Query Generation”
- “DialSQL” – GitHub [Online source], URL: https://github.com/izzeddingur/DialSQL