GENERALIZATION AND BACK-TRANSLATION IN RESPONSE GENERATION FROM SQL

Saptarashmi Bandyopadhyay (University of Maryland, College Park)
Tianyang Zhao (Pennsylvania State University, University Park)

EMNLP 2020
Intex-SemPar Workshop
November 19, 2020

OVERVIEW

- Natural Language Response Generation from SQL and Query Execution Results
- Related Works
- Baseline model and fine-tuning
- Pre-processing method:
  - Keyword Generalization
  - True-casing of Response
  - Byte pair encoding
  - Linguistic factorization
- Shuffled Back-translation
- Experiments and Results
- Conclusion

NATURAL LANGUAGE RESPONSE GENERATION FROM SQL AND QUERY EXECUTION RESULTS

Problem Introduction

NATURAL LANGUAGE RESPONSE GENERATION FROM SQL AND QUERY EXECUTION RESULTS

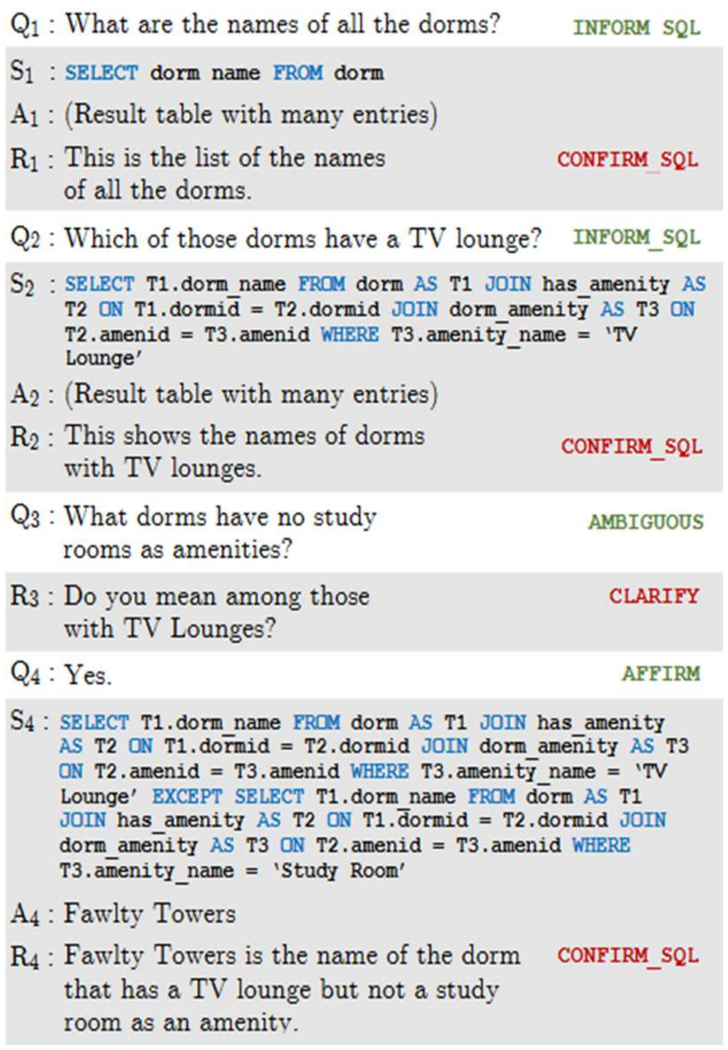
- This research is inspired by **CoSQL** (Conversational text-to-SQL corpus).
  - **Input**: a SQL query, the results, and the table schema
  - **Output**: a natural language response that describes the query and results
- **Example**:

  SELECT T1.model FROM MODEL_LIST AS T1 JOIN CAR_MAKERS AS T2 ON T1.Maker = T2.Id GROUP BY T2.id order by count (*) desc limit 1 | pontiac
- The model which is the most popular is pontiac

EVALUATION METRICS

- BLEU (bilingual evaluation understudy) score
  - Evaluates the similarity between generated NL response and the reference NL response
- Logic Correctness Rate (LCR): 0/1, indicating if a response is right logically
- Grammar: Between 1 and 5, indicating a response’s correctness on grammar
- LCR and Grammar are evaluated by Human Evaluators
- Results presented are BLEU scores on the development set.
  - We used Moses’ implementation to do so.
  - (https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl)

TASK MOTIVATION: NATURAL LANGUAGE INTERFACE TO DATABASE (NLIDB)



Q_1 : What are the names of all the dorms? INFORM_SQL

S_1 : `SELECT dorm_name FROM dorm`

A_1 : (Result table with many entries)

R_1 : This is the list of the names of all the dorms. CONFIRM_SQL

Q_2 : Which of those dorms have a TV lounge? INFORM_SQL

S_2 : `SELECT T1.dorm_name FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3 ON T2.amenid = T3.amenid WHERE T3.amenity_name = 'TV Lounge'`

A_2 : (Result table with many entries)

R_2 : This shows the names of dorms with TV lounges. CONFIRM_SQL

Q_3 : What dorms have no study rooms as amenities? AMBIGUOUS

R_3 : Do you mean among those with TV Lounges? CLARIFY

Q_4 : Yes. AFFIRM

S_4 : `SELECT T1.dorm_name FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3 ON T2.amenid = T3.amenid WHERE T3.amenity_name = 'TV Lounge' EXCEPT SELECT T1.dorm_name FROM dorm AS T1 JOIN has_amenity AS T2 ON T1.dormid = T2.dormid JOIN dorm_amenity AS T3 ON T2.amenid = T3.amenid WHERE T3.amenity_name = 'Study Room'`

A_4 : Fawlty Towers

R_4 : Fawlty Towers is the name of the dorm that has a TV lounge but not a study room as an amenity. CONFIRM_SQL

(Yu,2019)

CHALLENGES FOR RESPONSE GENERATION

- No grammatical error in the generated NL Response
- Model must do well on the SQL queries and database schema not seen before
- Under-researched

RELATED WORKS

- Focus on text to SQL generation - Shin (2019) and Zhong et al. (2017)
  - emphasize self-attention and reinforcement
- Generalize SQL keywords for better response generation - Gray et al. (1997)
- Back-translation - Sennrich et al. (2015) and Hoang et al. (2018)
  - to increase the BLEU score
- Linguistic generalization results – Bandyopadhyay (2019) and Bandyopadhyay (2020)
  - lemma and the Part-of-Speech tag are added to the natural language dataset for better generalization.

BASELINE MODEL AND FINE-TUNING

BASELINE MODEL

- Intuition: The SQL queries and the results are seen as the source language, and the natural language responses are regarded as the target language.
- Baseline model: Seq2seq, implemented by Tensorflow NMT. (https://github.com/tensorflow/nmt)
- Files required for training: parallel corpus and vocabularies in both “languages”.

FINE-TUNING OF BASELINE MODEL

Encoder type	Encode & Decoder layers	Source sequence length	Target sequence length	Optimizer	Learning rate	Learning decay scheme	Training steps	Dropout rate	BLEU
uni	2	50	50	sgd	1	No decay	12000	0.2	4.76
uni	2	50	50	adam	0.001	luong10	12000	0.2	6.46
uni	2	50	50	adam	0.0001	luong10	12000	0.2	4.63
uni	2	50	50	adam	0.001	luong234	12000	0.2	5.76
uni	4	50	50	adam	0.001	luong10	12000	0.2	6.82
uni	8	50	50	adam	0.001	luong10	12000	0.2	4.64
uni	4	60	60	adam	0.001	luong10	12000	0.2	6.93
uni	4	60	60	adam	0.001	luong10	12000	0.4	7.17
uni	4	70	70	adam	0.001	luong10	12000	0.4	6.84
uni	4	60	60	adam	0.001	luong10	24000	0.4	5.75
bi	4	60	60	adam	0.001	luong10	12000	0.4	**7.60**

PRE-PROCESSING FOR IMPROVEMENT:

A. KEYWORD GENERALIZATION

B. TRUE CASING

C. BYTE PAIR ENCODING

D. LINGUISTIC FACTORIZATION

KEYWORD GENERALIZATION

Idea: To avoid the cases where the model reacts poorly to certain keywords because they are under-represented in the training set.

SQL Keywords	**Generalization**
UNION, INTERSECTION, EXCEPT	SET
AND, OR	LOGIC
EXISTS, UNIQUE, IN	NEST
ANY, ALL	RANGE
AVG, COUNT, SUM, MAX, MIN	AGG

KEYWORD GENERALIZATION: KEYWORDS LEFT

The common SQL keywords that are not generalized don’t fit into any of the groups in the table.

- For example, GROUP BY is followed by a “grouping list” such as GROUP BY age.
- HAVING is followed by a “group qualification” such as HAVING age > 20.

KEYWORD GENERALIZATION: OPERATOR GENERALIZATION

The operators in SQL queries are also not generalized because ≤, ≥, <, and > are only used to compare numerical values, while = and ≠ are used to compare non-numerical values like strings.

- BLEU with baseline model and keyword generalization: 9.72
- BLEU with baseline and keyword & operator generalization: 8.84

TRUE-CASING

- Appropriate capitalization of words in the corpus.
  - “**t**he average salary of all the instructors is result0 .”
  - “**h**ere are the titles of the films that were directed by **B**ill **S**chreiner”
- There is also an open source script for true-casing by Moses (https://github.com/moses-smt/mosesdecoder/tree/master/scripts/recaser)
- The given ground truth for the development set is already true-cased.

TRUE-CASING RESULTS

Pre-processing applied	BLEU
Keyword generalization	9.72
Keyword generalization, true-cased	10.39

BYTE PAIR ENCODING

- Byte Pair Encoding (BPE) is a popular way in NMT to deal with the rare words.
- We used the implementation by subword-nmt (https://github.com/rsennrich/subword-nmt) on natural language responses.
- BPE separates the common subparts of a rare word.
- For example, Waterloo -> W@@ at@@ er@@ l@@ o@@ o,

  2016 -> 201@@ 6
- Idea: To make the model react better to rare words by generalization.

LINGUISTIC FACTORIZATION

- Linguistic factorization transform each word in the data into the format The Original Word | The Lemma | Part of Speech.
- For example, the word “typical” in the original data would be replaced by “typical | typic | JJ” after pre-processing.
- Idea: To provide more information for each word in the training data and remove the additional lemma and PoS tag after inference.

BPE AND LINGUISTIC FACTORIZATION RESULTS

Pre-processing applied	BLEU
keyword generalization, true-cased	10.39
keyword generalization, true-cased, byte pair encoding	8.60
keyword generalization, true-cased, linguistic factorization	7.48
keyword generalization, true-cased, BPE, linguistic factorization	6.67

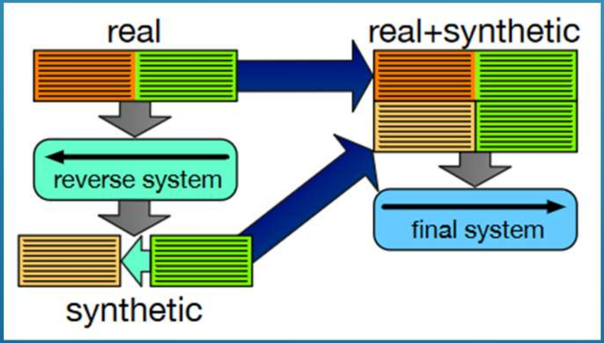
NL RESPONSE GENERATION
A. BACK - TRANSLATION
B. CYCLIC - TRANSLATION
C. SHUFFLED BACK - TRANSLATION

BACK-TRANSLATION

Motivation: To augment the training data in both the source and the target language.

Steps:

1. Train a response-to-query model with the training data.
2. Infer with gold development response file to obtain synthetic SQL data.
3. Append synthetic SQL data and gold development response file to the original SQL-to-Text corpora.
4. Train a query-to-response model with the new, augmented parallel dataset.



(Cong, 2018)

VARIATION 1: CYCLIC-TRANSLATION

Motivation: To deal with bias introduced by the vanilla back-translation.

Steps:

1. Train a query-to-response model with the original training data to create MODEL_QR.
2. Train a response-to-query model with the original training data to create MODEL_RQ.
3. Infer on MODEL_RQ with gold development response file to obtain synthetic SQL data.
4. Infer on MODEL_QR with synthetic SQL data to obtain synthetic response data.
5. Append synthetic SQL data and synthetic response data to the original SQL-to-Text corpora.
6. Train another query-to-response model with the augmented parallel data.

VARIATION 2: SHUFFLED BACK-TRANSLATION

Motivation: To deal with bias introduced by the vanilla back-translation.

Steps:

1. Train a response-to-query model with the training data.
2. Shuffle gold development response file.
3. Inference on shuffled gold development response file to obtain synthetic SQL data.
4. Append synthetic SQL data and shuffled gold development response file to the original SQL-to-Text corpora.
5. Train a query-to-response model with the new, augmented parallel data.

RESULTS WITH PROPOSED MODELS

NL Response Generation	Training steps	Dropout rate	BLEU
Back-translation	12000	0.4	11.05
Cyclic-translation	12000	0.4	10.85
Shuffled back-translation	12000	0.4	10.46
Back-translation	20000	0.5	11.75
Cyclic-translation	20000	0.5	9.50
Shuffled back-translation	20000	0.5	12.12

Note: Pre-processing involves SQL keyword generalization and truecasing

CONCLUSION

- In this research work, we tackled Natural Language Response Generation from SQL queries and Results with Seq2seq model implemented by TensorFlow NMT.
- We fine-tuned the baseline model and explored many ideas to improve its performance.
    - Several worked well: keyword generalization, true-casing, back-translation, cyclic-translation, and shuffled back-translation.
    - Some showed variation from our expectation: byte pair encoding and linguistic factorization.
- We purposed a novel idea cyclic-translation.

REFERENCES

Tao Yu, Rui Zhang, He Yang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, et al. 2019. Cosql: A conversational text-to-sql challenge towards cross-domain natural language interfaces to databases. arXiv preprint arXiv:1909.05378.

Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative backtranslation for neural machine translation. In Proceedings of the 2nd Workshop on Neural Machine Translation and Generation, pages 18–24.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. arXiv preprint arXiv:1511.06709.

Richard Shin. 2019. Encoding database schemas with relation-aware self-attention for text-to-sql parsers. arXiv preprint arXiv:1906.11790.

REFERENCES

Xiaoshi Zhong, Aixin Sun, and Erik Cambria. 2017. Time expression analysis and recognition using syntactic token types and general heuristic rules. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 420–429, Vancouver, Canada. Association for Computational Linguistics.

Saptarashmi Bandyopadhyay. 2019. Factored neural machine translation at loresmt 2019. In Proceedings of the 2nd Workshop on Technologies for MT of Low Resource Languages, pages 68–71.

Saptarashmi Bandyopadhyay. 2020. Factored neural machine translation on low resource languages in the covid-19 crisis.

Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. 1997. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. Data mining and knowledge discovery, 1(1):29–53.

THANK YOU!