ABSTRACT

Title of dissertation: HUMAN-IN-THE-LOOP QUESTION ANSWERING WITH NATURAL LANGUAGE INTERACTION

Ahmed Elgohary Ghoneim
Doctor of Philosophy, 2021

Dissertation directed by: Associate Professor, Jordan Boyd-Graber
Department of Computer Science

Generalizing beyond the training examples is the primary goal of machine learning. In natural language processing (NLP), impressive models struggle to generalize when faced with test examples that differ from the training examples: e.g., in genre, domain, or language. I study interactive methods that overcome such limitations by seeking feedback from human users to successfully complete the task at hand and improve over time while on the job. Unlike previous work that adopts simple forms of feedback (e.g., labeling predictions as correct/wrong or answering yes/no clarification questions), I focus on using free-form natural language as the communication interface for providing feedback which can convey richer information and offer a more flexible interaction.

An essential skill that language-based interactive systems should have is to understand user utterances in conversational contexts. I study conversational question answering (CQA) in which humans interact with a question answering QA system by asking a sequence of related questions. CQA requires models to link questions together to resolve the conversational dependencies between them such as coreference and ellipsis. I introduce question-in-context rewriting to reduce context-dependent conversational questions to independent stand-alone questions that can be answered with existing QA models. I collect a large dataset of human rewrites and I use it to evaluate a set of models for the question rewriting task.

Next, I study semantic parsing in interactive settings in which users correct parsing errors using natural language feedback. Most existing work frames semantic parsing as a one-shot mapping task. I establish that the majority of parsing mistakes that recent neural text-to-SQL parsers make are minor. Hence, it is often feasible for humans to detect and suggest corrections for such mistakes if they have the opportunity to provide precise feedback. I describe an interactive text-to-SQL parsing system that enables users to inspect the inferred parses and correct any errors they find by providing feedback in free-form natural language. I construct SPLASH: a large dataset of SQL correction instances paired with a diverse set of human-authored natural language feedback utterances. Using SPLASH, I posed a new task: given a question paired with an initial erroneous SQL parse, to what extent can we correct the parse based on a provided natural language feedback?

Then, I present NL-EDIT: a neural model for the correction task. NL-EDIT combines two key ideas: 1) interpreting the feedback in the context of the other elements of the interaction and, 2) explicitly generating edit operations to correct the initial query instead of re-generating the full query from scratch. I create a simple SQL editing language whose basic units are add/delete operations applied to different SQL clauses. I discuss evaluation methods that help understand the usefulness and limitations of semantic parse correction models.

I conclude this thesis by identifying three broad research directions for further advancing collaborative human–computer NLP: (1) developing user-centered explanations , (2) designing and evaluating interaction mechanisms, and (3) learning from interactions.

HUMAN-IN-THE-LOOP QUESTION ANSWERING WITH NATURAL LANGUAGE INTERACTION

by

Ahmed Elgohary Ghoneim

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Dedication

To my parents, Eman and Ali.

Acknowledgments

During the past seven years, I have been very fortunate to know and work with great people to whom I owe a lot of gratitude for their support.

Foremost, I would like to thank my advisor, Jordan Boyd-Graber for his tireless guidance, patience, and support. Jordan took me as his advisee when I was starting my fourth year. Not all advisors would do that. He was always available to meet with me, discuss my initial research ideas and provide a lot of feedback and key pointers to resources that shaped my interest in human-in-the-loop NLP. I have been always amazed by how keen he is on training his student on all aspects of research and academia, from formulating ambitious and creative research problems to the very details of writing papers and preparing posters/talks. I am also very thankful to Jordan for his advice throughout my job search process, starting from preparing my research statement to deciding between the offers. It was totally worth it to wait until Jordan joined UMD at the beginning of my fourth year.

I also would like to thank my other committee members: Rachel Rudinger, Rob Patro, Dragomir Radev, and Doug Oard for their valuable feedback on my work and the great discussions during the defense. Doug Oard, in particular, has been very supportive to me on several occasions during my time at UMD.

During my internships at Microsoft Research, AI2, and IBM Research, I collaborated with very inspiring people, not just at the technical level but also at the personal level. Ahmed Hassan Awadallah introduced me to the interactive semantic parsing line of work that ended up being the central focus of my thesis. That work would not have been completed without his continuous mentorship and feedback. I am really grateful to Ahmed for beliving in me and offering me the opportunity to do two internships at Microsoft Research. I am also very grateful to my mentors, collaborators, and colleagues: Matthias Boehm, Fred Reiss, Peter Haas, Waleed Ammar, Matt Peters, Saghar Hosseini, Chris Meek, Matt Richardson, Adam Fourney, Gonzalo Ramos, Xiang Deng, Alex Polozov, Tao Yu, and Guoqing Zheng.

At UMD, I learned a lot from (and enjoyed interacting with) my colleagues and friends at the CLIP lab over the years. I was lucky to know and work closely with Allyson Ettinger early in my Ph.D. I am grateful to Yogarshi Vyas, Xing Niu, Hadi Amiri, Mossaab Bagdouri, Mahmoud Sayed, Amr Sharaf, Weiwei Yang, Chen Zhao, Denis Peskov, Khanh Nguyen, Mohit Iyyer, Sudha Rao, Joe Barrow, Jo Shoemaker, Michelle Yuan, Alison Smith, Shi Feng, Pedro Rodriguez, and Fenfei Guo.

Also from UMD, Tom Hurst and Pedram Sadeghian were among the most supportive and accommodating people to me. Tom wrote me several letters to get internship and program extension approvals. I worked as a TA with Pedram for three semesters. I am very grateful for his flexibility in accomodating my paper deadlines and job interviews.

I deeply thank IBM for funding me through a Ph.D. fellowship during 2017/2018. The fellowship greatly helped me focus on catching up with question answering and human-in-the-loop NLP research, which ended up being the topic for my thesis.

Indeed, there have been a lot of challenging times during the past seven years. Certainly, I would not have reached this point without the support of my family and friends in Egypt and the US. Thank you, All.

Table of Contents

List of Tables

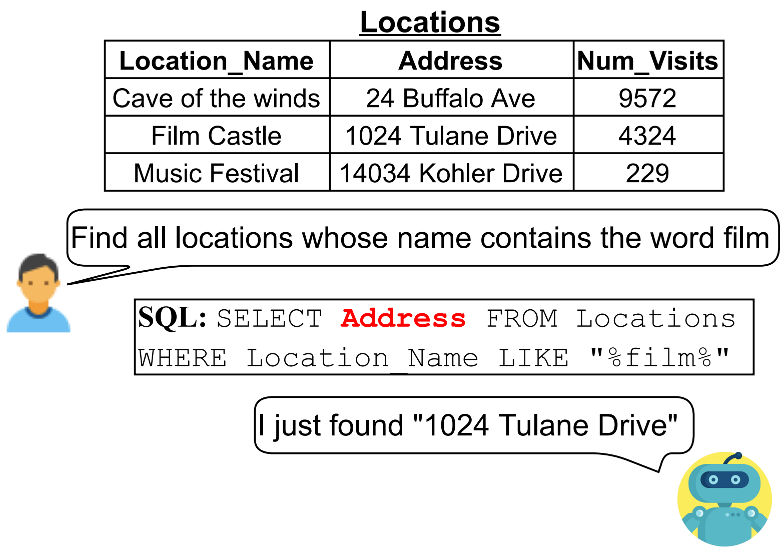
List of Figures

Chapter 1: Introduction

1.1 Motivation

The recent successes and impressive results in natural language processing (NLP) come from the following evaluation scenario: a system learns a pre-defined task (e.g., categorizing product reviews based on their sentiment) using a given set of training examples. The system is then evaluated on similar examples from the same task, domain, and language. When such systems are evaluated on slightly different examples or tasks (e.g., categorizing movie reviews based on their sentiment or categorizing reviews as spam or ham), their accuracy drops significantly and restoring a system's accuracy on a different evaluation setup requires collecting a large enough amount of training examples from that setup and adapting the system on them.

With more evaluation setups, that adaptation process (including training data collection) becomes slow, expensive, and hard to manage. Take question answering (QA) as an example: most of the research progress in the past five years has been centered around constructing datasets, each focuses on a reasoning process, a domain, a genre or a linguistics phenomenon. Examples include datasets that focus on multi-hop reasoning (Yang et al., 2018), coreference resolution (Dasigi et al., 2019),


Locations

Location_Name	Address	Num_Visits
Cave of the winds	24 Buffalo Ave	9572
Film Castle	1024 Tulane Drive	4324
Music Festival	14034 Kohler Drive	229

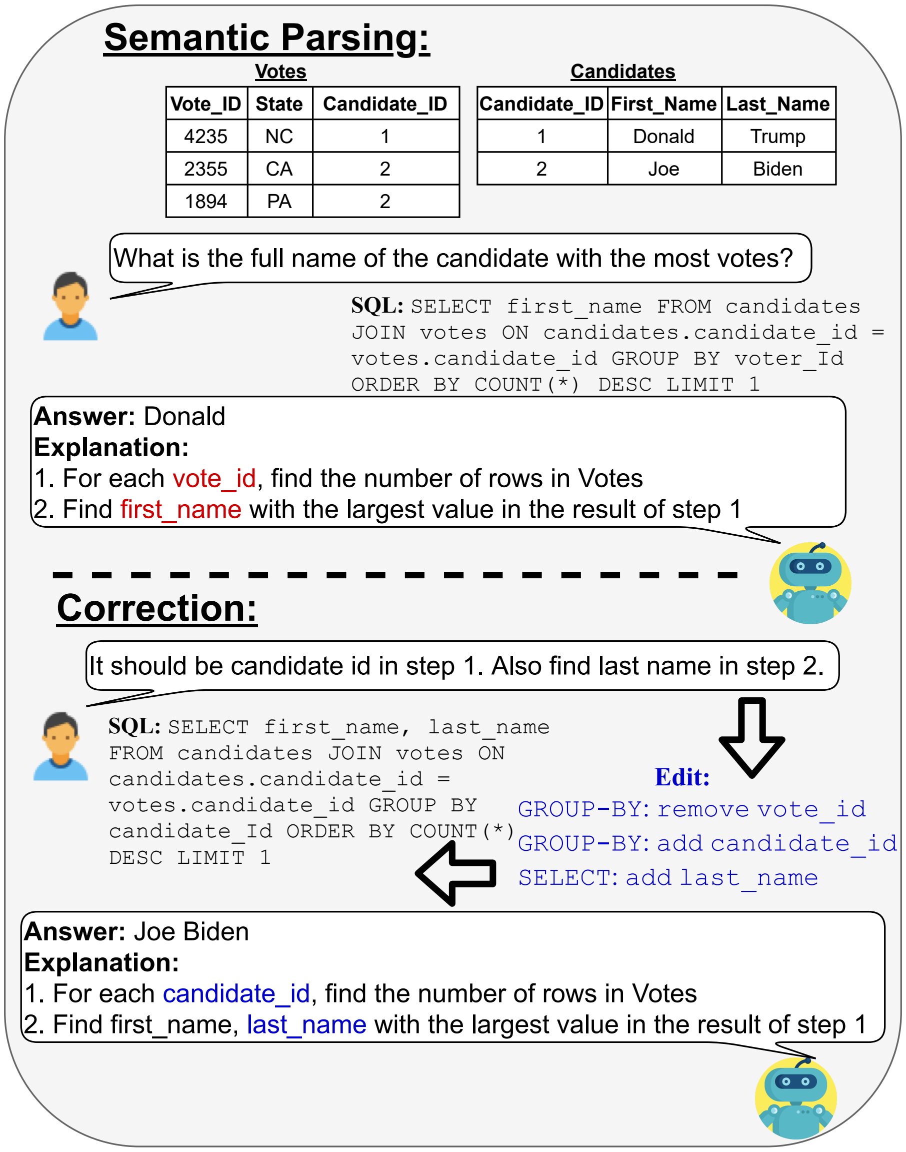
```
SQL: SELECT Address FROM Locations
WHERE Location_Name LIKE "%film%"
```

Find all locations whose name contains the word film
I just found "1024 Tulane Drive"

Figure 1.1: Example minor test-time error of a question answering system.

conversational question answering (Choi et al., 2018; Reddy et al., 2019), QA over research papers (Dasigi et al., 2021), etc.

At test time, even the strongest existing neural network models are not perfect. For a given input, a minor mistake with the operations performed by the model can result in a wrong output or an output that does not fully satisfy the user needs. Furthermore, it is impractical to assume that users will always be able to provide precise and unambiguous inputs to the model. Figure 1.1 demonstrates an example on that behaviour. The input "Find all locations whose name contains the word film" does not really specify whether to output the names of those locations "Location_Name" or their "Addresses". The deployed QA system makes an assumption that the user is interested in the "Address" and generate a SQL query whose execution result is "1024 Tulane Drive". What if the user was interested in the names of those locations rather than their addresses? Although it can be frustrating to


Semantic Parsing:
Votes
Candidates
What is the full name of the candidate with the most votes?
Answer: Donald
Explanation:
1. For each vote_id, find the number of rows in Votes
2. Find first_name with the largest value in the result of step 1
Correction:
It should be candidate id in step 1. Also find last name in step 2.
Edit:
GROUP-BY: remove vote_id
GROUP-BY: add candidate_id
SELECT: add last_name
Answer: Joe Biden
Explanation:
1. For each candidate_id, find the number of rows in Votes
2. Find first_name, last_name with the largest value in the result of step 1

Vote_ID	State	Candidate_ID
4235	NC	1
2355	CA	2
1894	PA	2

Candidate_ID	First_Name	Last_Name
1	Donald	Trump
2	Joe	Biden

SQL:
```
SELECT first_name FROM candidates
JOIN votes ON candidates.candidate_id =
votes.candidate_id GROUP BY voter_Id
ORDER BY COUNT(*) DESC LIMIT 1
```

SQL:
```
SELECT first_name, last_name
FROM candidates JOIN votes ON
candidates.candidate_id =
votes.candidate_id GROUP BY
candidate_Id ORDER BY COUNT(*)
DESC LIMIT 1
```

Figure 1.2: End-to-end interaction with the NL-EDIT system.

the user, by looking at the output the user can potentially realize the assumption the system made and repeat the whole question in a more specific way as “Find the names of the locations that contain the word film in their name”. In other scenarios, it might even be impossible for users to notice if the system made a mistake or operated under a certain assumption. Take for example the question “How many hotels are in Cairo?”: How can the user verify an answer like “7501”?

This thesis studies how we can build *interactive* NLP *systems* that enable users to inspect and provide feedback on the operations performed by the underlying models. By adopting and counting on interaction components within NLP systems, we will be able to quickly deploy models that are trained on the available data at-hand and learn over time from user interactions about any missing skills (e.g., linguistic phenomena, reasoning operations) while the models are on the job. Also, providing users with a means to interact with the system allows them to guide (collaborate with) the system to correct for test-time errors. Figure 1.2 show an example user interaction with a question answering system that I present in Chapter 6. The system answers user questions about a votes database by automatically translating them into SQL queries that produce the answer when executed against the database (Section 2.3.4). In the first attempt, the system generates a SQL query that contains two mistakes hence, gives the wrong answer “Donald”. The user then further interacts with the system and provides *feedback* that the system uses to revise the initial query and produce the correct answer “Joe Biden”.

Existing research considers constrained interaction mechanisms such as asking users to label system outputs as correct or wrong, rating the overall quality of the output with a numeric score, or answering yes/no clarification questions posed by the system. Humans have the ability to learn new concepts or correct others based on natural language description or feedback. A few previous studies have explored how machines can learn from language in tasks such as playing games (Branavan et al., 2012), robot navigation (Karamcheti et al., 2017), and concept learning (e.g., shape, size, etc.) (Srivastava et al., 2018). Also, with the increasing popularity of of personal assistants that use natural language as their primary user interface (e.g., Alexa, Cortana, Siri), the potential of natural language based interaction is becoming more and more evident. I focus this work on using free-form natural language as the means for providing feedback which can convey richer information and offer a more flexible user interaction.

Question answering is a central problem in NLP. In order for a system to answer questions, it should be able to implicitly perform a wide range of language understanding and reasoning tasks. In this thesis, I use question answering to study interactive human-in-the-loop NLP systems. An essential skill that language-based interactive systems should have is to understand user utterances in conversational contexts. I develop tasks and models for conversational QA and question-in-context rewriting. Then, I switch gears to cross-domain QA over structured databases. I create a framework, datasets, and models through which users can collaborate with the QA system to answer questions by providing free-form natural language feedback.

1.2 Thesis Contributions

I outline the contributions I make in this thesis. First, I use natural language as a source of supervision to learn vector representations of sentences (Section 1.2.1). Then in the context of question answering, I study tasks and methods that enable machines to understand conversational natural language (Sections 1.2.2 and 1.2.3). Finally, I build a system and models for question answering with natural language feedback from users (Section 1.2.4).

1.2.1 Learning Representations from Naturally-Occurring Translation

The recent progress and impressive results in NLP are primarily attributed to pre-trained representations (Mikolov et al., 2013b; Pennington et al., 2014; Wieting et al., 2016) and architectures (Peters et al., 2018; Devlin et al., 2019). Several approaches have been developed and evaluated for pre-training text representations (Section 2.1). In the first contribution of this thesis, I demonstrate the richness and effectiveness of using *natural language supervision* to learn general-purpose vector representations of sentences (Chapter 3). In particular, I use naturally-occurring translations (bilingual text) as meaning annotations for the text in the language of interest. Using the semantic textual similarity task, I show that such representations outperform representations learned using raw text and monolingual paraphrases.

1.2.2 Task and Baselines for Sequential Question Answering

Previous research has made ample progress on answering stand-alone questions using a given textual evidence (machine reading comprehension) (Seo et al., 2017; Devlin et al., 2019; Lan et al., 2019). New neural models are continuously pushing the state-of-the-art results of popular benchmark datasets. In information-seeking dialogs, e.g., personal assistants, users interact with a QA system by asking a sequence of related questions, where questions share the same predicate, entities, or at least a topic. Answering each question in isolation is sub-optimal as information from previously asked questions and previously obtained answers can help better answer the current question. I study answering sequences of interrelated

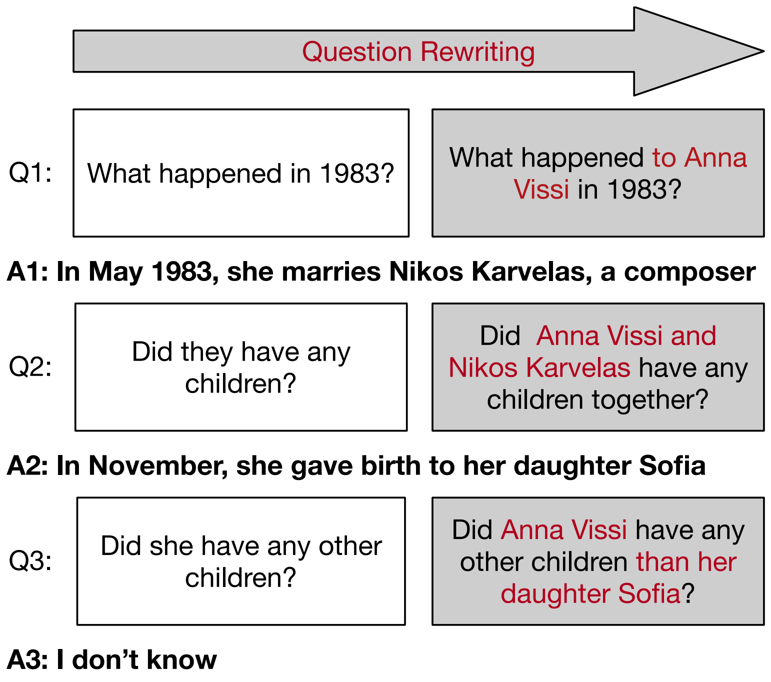

Question Rewriting
Q1:
What happened in 1983?
What happened to Anna Vissi in 1983?
A1: In May 1983, she marries Nikos Karvelas, a composer
Q2:
Did they have any children?
Did Anna Vissi and Nikos Karvelas have any children together?
A2: In November, she gave birth to her daughter Sofia
Q3:
Did she have any other children?
Did Anna Vissi have any other children than her daughter Sofia?
A3: I don't know

Figure 1.3: Example questions in conversation context with corresponding rewrites.

questions where for each question, the QA system has to retrieve the evidence document from Wikipedia before starting the reading comprehension step. In that work, I present and evaluate models that incorporate connections between questions in the same sequence to improve both the retrieval and the reading steps of the QA process (Chapter 4).

1.2.3 Task, Dataset and Models for Question-in-Context Rewriting

In the more naturally conversational settings, individual questions cannot be understood without resolving conversational dependencies. For example, the question "What was he like in that episode?" cannot be understood without knowing what "he" and "that episode" refer to. Instead of implicitly reasoning about the conversational dependencies between questions (as in Section 1.2.2), I introduce the task of question-in-context rewriting as a means to reducing context-dependent conversational questions to independent stand-alone questions that can be answered with existing QA models (Chapter 4). I create CANARD: a dataset of 40,000 questions asked in conversational contexts paired with corresponding human-authored (crowd-sourced) stand-alone paraphrases. Example question-rewriting pairs are given in Figure 1.3. I show that the task and dataset do not only challenge the ability of models to resolve coreferences, but also resolve ellipses. A question such as "Why?" is expected to be written as "Why did the publicity of Jerry Lee Lewis' personal life cause an uproar?". Using CANARD, I develop and analyze models for the rewriting tasks. Follow-up work has shown that answering conversational questions through rewriting with models trained on CANARD improves the end-to-end QA accuracy and sets new state-of-the-art results on the passage retrieval setup of QA (Vakulenko et al., 2021) and on conversational semantic parsing (Chen et al., 2021).

1.2.4 Framework for Interactive Semantic Parsing with Natural Language Feedback

Major progress in natural language processing has been made towards fully automating challenging tasks such as question answering, translation, and summarization. On the other hand, several studies have argued that machine learning systems that can explain their own predictions (Doshi-Velez and Kim, 2017) and learn interactively from their end-users (Amershi et al., 2014) can result in better user experiences and more effective learning systems. I develop an interactive semantic parsing framework that employs both explanations and interaction in order to boost the end-to-end accuracy of text-to-SQL systems.

A large fraction of the data of interest in various domains are maintained in structured formats (e.g., relational databases) that support performing complex analytics and data manipulation operations through specialized query languages (e.g., SQL). Natural language interfaces (NLIs) make structured data accessible to users with no/limited knowledge of such query languages by employing a semantic parser that maps natural language utterances to the target query language. For example, in Figure 1.2 instead of expecting users to write complex SQL queries such as

```
SELECT  first_name,  last_name FROM candidates JOIN votes ON
candidates.candidate_id = votes.candidate_id GROUP BY candidate_Id
ORDER BY COUNT(*) DESC LIMIT 1
```

the user expresses the information they need in natural language as "What is the full name of the candidate with the most votes?", and the system generates the corresponding SQL query automatically.

NLIs have been the "holy grail" of natural language understating and human–computer interaction for decades (Woods et al., 1972; Codd, 1974; Hendrix et al., 1978; Zettlemoyer and Collins, 2005). However, early attempts in building NLIs to databases did not achieve the expected success due to limitations in language understanding capability (Androutsopoulos et al., 1995; Jones and Galliers, 1995). NLIs have been receiving increasing attention recently motivated by interest in developing virtual assistants, dialogue systems, and semantic parsing systems. NLIs to databases were at the forefront of this wave with several studies focusing on parsing natural language utterances into executable

SQL queries (text-to-SQL parsing).

Most of the existing work on semantic parsing aims at directly translating the full natural language input into a query at the first attempt (one-shot mapping). I show that the majority of the mistakes made by recent neural text-to-SQL parsers are minor (e.g., "`SELECT first_name`" instead of "`SELECT last_name`"). As such, it is often feasible for users to detect and suggest *corrections* for such mistakes if they are afforded a means of providing precise feedback.

I introduce a framework for interactive semantic parsing in which users interact with the parser through natural language feedback (Chapter 5). Additionally, to enable users with no SQL knowledge to inspect the inferred queries, I develop a template-based approach for explaining SQL queries in the form of intuitive natural language steps. With those components put together, I create a crowdsourcing pipeline to construct SPLASH: a large dataset of SQL correction instances paired with a diverse set of human-authored natural language feedback utterances.

Using SPLASH, I pose a new NLP task: given a question paired with an initial erroneous SQL parse, to what extent can we correct the parse based on a provided natural language feedback? I develop and evaluate a series of handcrafted and deep neural models for that task. NL-EDIT (Chapter 6)—the state-of-the-art model—-combines two key ideas: (1) interpreting the feedback in the context of the other elements of the interaction (database schema, question, and explanation of the initial query). (2) Explicitly generating edit operations to correct the initial query instead of re-generating the full query from scratch. To do that, I design a simple SQL editing language (see **Edit** in Figure 1.2) whose basic units are add/delete operations applied to different SQL clauses e.g.,

```
<SELECT> add last_name </SELECT>
```

which add the column `last_name` to the `SELECT` clause. I show that with only one turn of feedback, NL-EDIT is able to fully correct up to 40% of the erroneous queries, boosting the end-to-end parsing accuracy by more than 20%. Figure 1.2 demonstrates the end-to-end operation of the framework.

Starting from Chapter 3, I present the details of the contributions I outline in this chapter. In the next chapter, I provide background and review previous work relevant to the contributions I present. I discuss relevant deep learning modules that I use to develop my methods, and I discuss different paradigms/formats for question answering including semantic parsing and text-to-SQL. As this thesis is centred around collaborative human–computer question answering, I review previous work on human-in-the-loop NLP.

Chapter 2: Background and Related Work

In this Chapter, I provide essential background that I build on in the following chapters. I start by introducing neural network architectures and models for NLP. QA methods can be categorized according to the knowledge source used to find the answer as: QA over raw text and QA over structured knowledge base. I review relevant details about each category. As I aim at improving QA with human feedback, I also review related work on human-in-the-loop NLP.

2.1 Deep Learning for NLP

Among other application domains, NLP has benefited from the recent advancements in deep neural networks. It is hard to think of an NLP task whose state-of-the-art model is something other than a neural network model. Throughout my thesis, I use neural models for all the tasks I study. In this section, I introduce relevant deep learning components and models that I build on.

2.1.1 Representations for NLP

Deep learning is based on computing vector representations (embeddings) of the input data and using such representations for making predictions. In NLP, deep learning methods aim to learn representations for words, sentences, documents, etc. For individual words, vector representations are typically learned from raw text with various objective functions including language modeling (Bengio et al., 2003), word prediction within a short context window of surrounding words e.g., WORD2VEC (Mikolov et al., 2013a), and factorization of word co-occurrence matrices e.g., GLoVE (Pennington et al., 2014).

For tasks that require operating on sequences of multiple words (e.g., sentences), the embeddings of individual words are *composed* together to form a representation for each sequence. Such representation can either be one vector that summarizes information about the entire sequence or *contextualized* vector representations of the individual tokens that are "aware of" the entire sequence.

A significant body of research has focused on developing (Mitchell and Lapata, 2008; Socher et al., 2013; Kim, 2014; Iyyer et al., 2015, inter alia) and evaluating (Ettinger et al., 2016, 2018, inter alia) composition functions. Besides simply averaging the embeddings of individual words, the two other commonly used composition functions in NLP are recurrent neural networks and transformers. I cover both of them in the following two sections.

2.1.2 Recurrent Neural Networks

Given a sequence of tokens,[1] each is encoded individually as a vector (e.g., a GLoVE word embedding) a recurrent neural network (RNN) computes contextualized representations of the input tokens as follows: Suppose the corresponding word embedidings of the tokens in the sequence are x_1, x_2, …, x_m. An RNN computes a hidden representation z_i of each token i as a function of the hidden representation of its previous token z_{i-1} and the word

[1]Recurrent networks and Transformers are general models for various data types. For simplicity, I limit the discussion in this and the following section to natural language inputs (each is a sequence of tokens).

embedding of x_i

$$z_i = \sigma_z(W_z z_{i-1} + W_x x_i + b_z) \tag{2.1}$$

The hidden representations are then used to compute the contextualized output representations as

$$y_i = \sigma_y(W_y z_i + b_y) \tag{2.2}$$

where W_z, W_x, W_y, b_z, b_y, z_0 are learned parameters and σ_z and σ_y are non-linear activation function. The output vectors $y_1, y_2, \ldots, y_m$ can be used directly as a representation of the input sequence of combined (pooled) in a single vector by e.g., averaging them or computing their element-wise maximum.

That formulation makes each y_i aware of all tokens in the sequence up to token i, but unaware of the following tokens in the sequence. In a bidirectional RNN the input sequence is processed once from left to right (as in the formulation above) and once from right to left, i.e.,

$$\overrightarrow{z_i} = \sigma_z(\overrightarrow{W_z}\overrightarrow{z_{i-1}} + \overrightarrow{W_x}x_i + \overrightarrow{b_z})$$

$$\overrightarrow{y_i} = \sigma_y(\overrightarrow{W_y}\overrightarrow{z_i} + \overrightarrow{b_y})$$

$$\overleftarrow{z_i} = \sigma_z(\overleftarrow{W_z}\overleftarrow{z_{i-1}} + \overleftarrow{W_x}x_i + \overleftarrow{b_z})$$

$$\overleftarrow{y_i} = \sigma_y(\overleftarrow{W_y}\overleftarrow{z_i} + \overleftarrow{b_y})$$

to produce two output embeddings for each token that are concatenated to form the final representation $y_i = [\overrightarrow{y_i}; \overleftarrow{y_i}]$ that is aware of the full sequence. Also, multiple layers of RNN can be stacked on top of each other where the outputs of each layer $y_1, y_2, \ldots, y_m$ are fed as the inputs to the layer on top of it.

In Chapter 4, I use variants of RNN and bidirectional-RNN namely, long short-term memory networks (LSTM) and BILSTM (Hochreiter and Schmidhuber, 1997). LSTMs follow the same formulation as in Equations 2.1 and 2.2, but they apply additional transformations to the inputs and hidden representations to ease the training.

2.1.3 Transformers

RNNs propagate information about token i sequentially to token j by updating the hidden representations starting from z_i to z_j. In the Transformer model (Vaswani et al., 2017), each token i *directly* accesses information about every other token in the input through *self-attention*. Suppose the input tokens are initially represented with vectors x_1, x_2, …, x_m. A contextualized representation z_i of the i-th token is computed as a weighted sum of the *value-encoding* v_j of all tokens in the sequence

$$z_i = \sum_{j=1}^{m} \alpha_{ij} v_j \tag{2.3}$$

where the weights $\alpha_{ij} = \text{softmax}(e_{i1}, e_{i2}, \ldots, e_{im})$ and the unnormalized scores e_{ij} are computed as

$$e_{ij} = \frac{q_i^T k_j}{\sqrt{d_k}} \tag{2.4}$$

In the two equations above, q_i, k_i and v_i are the *query-encoding*, *key-encoding* and *value-encoding* of the i-th token, respectively, and they are computed with linear transformations of x_i as $q_i = x_i W_Q$, $k_i = x_i W_K$, and $v_i = x_i W_V$, where W_Q, W_K, and W_V are learnable parameters. d_k is the dimensionality is the key (and query)-encodings.

Instead of computing one version of z_i, Vaswani et al. (2017) proposed the idea of multi-head attention in which H copies of z_i are computed, each with a different set of the learned parameters W_Q, W_K and W_V. The H copies $z_i^1 \ldots z_i^H$ are then concatenated to form the final vector z_i which is then passed through a non-linear function (ReLU). To ease the training process, the outputs are further passed through a residual connection (He et al., 2016a) and a layer normalization (Ba et al., 2016). The final output y_i is used as the contextualized representation of each token i. Typically, multiple such layers are stacked on top of each other where the output of one layer is fed as the input to the layer on top of it, and the final y_i is the output of the last layer.

2.1.4 Relation-Aware Transformer

The self-attention scores α_{ij} in Equation 2.3 can be viewed as scores of the relatedness (syntactic, semantic, etc) between the token i and the token j. The types of those relations and the interpretation of the assigned score are implicit in the model. That is still useful as a well-trained model is expected to learn the relevant relations to the task at hand. However, in some scenarios (Chapter 6 and Section 2.3.4), we might be interested in incorporating preexisting relations in the transformer model besides the automatically/implicitly discovered relations via self-attention. For example, Shaw et al. (2018) show that the relative position between token i and token j can be a useful information for some tasks.

The relation-aware transformer (RAT) (Shaw et al., 2018; Wang et al., 2020) is an extension of the transformer model that allows for hard encoding preexisting relations between different tokens in the self-attention computation. Suppose for a particular task, we have R types of relations that can hold between input tokens. RAT learns a vector representation (embedding) r_t for each relation type t. For example, to add relative position information, we can define 12 relations for the relative positions {-5, -4, . . . , 0, 1, . . . 5, Other}, and learn an embedding for each of them. The relation embeddings are incorporated in Equations 2.3 and 2.4 as follows

$$z_i = \sum_{j=1}^{m} \alpha_{ij}(v_j + r_{\text{type}(i,j)}) \tag{2.5}$$

$$e_{ij} = \frac{q_i^T(k_j + r_{\text{type}(i,j)})}{\sqrt{d_k}} \tag{2.6}$$

where type(i, j) returns the index of the relation type that holds between i and j. The general framework presented in (Wang et al., 2020) learns a set of relation embeddings that are used for Equation 2.5 and a different set for Equation 2.6. Throughout this thesis, I use the same set of embeddings in both equations (following Wang et al. (2020)'s application of RAT to text-to-SQL parsing).

2.1.5 Sequence-to-Sequence Models

Many tasks in NLP involve generating an output sequence using an input sequence such as machine translation, abstractive summarization, and dialog response generation. The input and output sequences are often sequences of tokens, but they can also be multiple dialog utterances (Serban et al., 2016), grammar production rules (Yin and Neubig, 2017), waveforms (van den Oord et al., 2016), etc. Sequence-to-sequence (SEQ2SEQ) models (Sutskever et al., 2014) are a general class of neural models for such tasks.

A basic SEQ2SEQ model operates by encoding the input sequence and then generate the output sequence (decoding) one item (token) at a time by conditioning on the input encoding and the previously generated output tokens. To mitigate the potential degeneration of the output sequence as a result of the greedy decoding approach (the most likely token is selected at each step), a beam search decoder is used in which multiple partial outputs (a beam) are maintained and scored throughout the generation process.

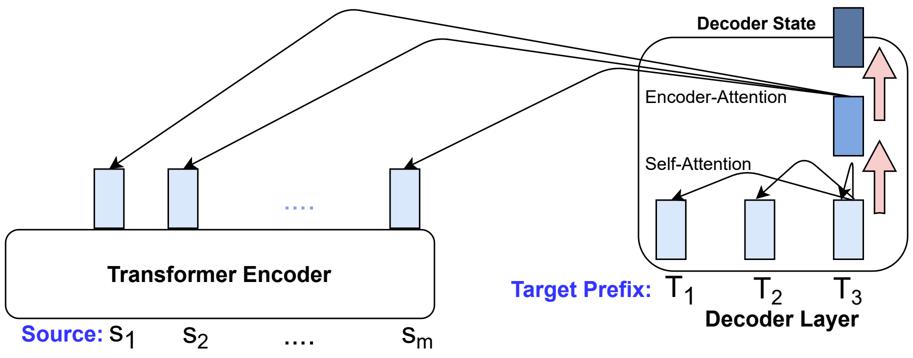


Figure 2.1: Transformer Sequence-to-Sequence Architecture

LSTMs (Section 2.1.2) and transformer networks (Section 2.1.3) are the typical choices for encoding the input and the partial output sequences. Also, attending on the encoding of the relevant individual input tokens (Bahdanau et al., 2015) while generating each output token is often employed. To enable copying tokens from the input in certain decoding steps, a copy mechanism (Vinyals et al., 2015; See et al., 2017) that alternates between copying and generating tokens is used.

Figure 2.1 explains how the transformer model is extended to the SEQ2SEQ setup. The input (source) is encoded with a multi-layer transformer exactly the same way as explained in Section 2.1.3. The target output is generated one token at a time. With a decoded prefix (a special start token in the first step), each decoder layer applies the self-attention computations in Equation 2.5 and 2.6. The obtained representation of the last token is used to do an encoder-decoder attention: Using Equation 2.5 and 2.6, key and value-encodings are computed with linear transformations of the source representations and a query-encoding is computed for the decoder representation. The resulting representation of that process is the decoder state that is used to decide the next word in the target.

SEQ2SEQ models are typically trained with teacher-forcing: At training time, the gold (rather than the inferred) prefix of the target is what is used to compute the decoder state. In some SEQ2SEQ tasks (e.g. summarization, grammatical error correction, sentence splitting, etc.), there could be a large overlap between the input and the output. Recent models (Malmi et al., 2019; Panthaplackel et al., 2020; Stahlberg and Kumar, 2020) cast text generation as a text editing task where target texts are reconstructed from inputs using several edit operations.

2.1.6 Pre-Trained Transformers

Neural architectures that are pre-trained with language modeling objectives are steadily pushing the state-of-the-art results on several NLP tasks (Devlin et al., 2019; Radford et al., 2019; Rogers et al., 2021). The basic idea of pre-trained architectures is that the inputs to many NLP tasks can be cast as a sequence of tokens that is encoded with one network whose parameters are *pre-trained* on a large amount of raw text with a language modeling objective. A pre-trained architecture is then incorporated for a particular task (e.g., sentiment analysis) by using the task training data to train a lightweight task-specific prediction layer that sits on top of the pre-trained architecture. The pre-trained parameters can also be fine-tuned on the task-specific training set. The most commonly used pre-trained architecture is the transformer model (Section 2.1.3).

One example of such architectures is BERT (Devlin et al., 2019). BERT is a transformer model that is trained on raw text jointly with a masked language modeling objective (predict a masked word within a span) and a next-sentence prediction objective. To compute BERT representations for an input text (e.g., a movie review in a sentiment analysis task), it should be tokenized with WordPiece tokenization (Wu et al., 2016) and a special token [CLS] is prepended to it. In addition to the contextualized vector representations that BERT produces for each token, the corresponding representation to the [CLS] token can be used as a summary of the whole input text. BERT is also used for encoding multiple sequences (e.g., two sentences in a paraphrase detection task) by simply concatenating them together with a special separator token ([SEP]) in between to distinguish the boundaries of each sequence. BERT comes in two sizes: BERT-BASE which has 12 layers, 12 attention heads, and hidden representations of size 768, and BERT-LARGE which has 24 layers, 16 attention heads, and hidden representations of size 1024.

Improving the pre-training approach (e.g., by training on a different corpus or using different objectives) is an active research topic. RoBERTa (Liu et al., 2019b) is an improved variant of BERT that is trained for more iterations. Other variants included domain specific models, e.g., SciBERT (Beltagy et al., 2019), models for languges other than English (Antoun et al., 2020; Martin et al., 2020), and multilingual models (Lample and Conneau, 2019). Task-specific pre-training has also been explored, e.g., for dialogue response generation (Zhang et al., 2020), semantic parsing (Yu et al., 2021a,b; Deng et al., 2021), and SEQ2SEQ tasks (Lewis et al., 2020).

2.2 Question Answering over Raw Text

Given a question q and a a set of potential evidence documents D, the general structure of an open-domain QA system consists of three components:

1. A *retrieval* component that decides a narrow subset D' of D given q.

2. A *reading comprehension* component that jointly analyzes q and each document in D' and explicitly produces relevant spans (e.g., answer candidates) or implicitly encodes spans relevance to q in an intermediate representation.

3. An *answer generation* component that aggregates the outputs of the reading com-

Document: Anna Vissi

In May 1983, she married Nikos Karvelas, a composer, with whom she collaborated in 1975 and in November she gave birth to her daughter Sofia. After their marriage, she started a close collaboration with Karvelas. Since 1975, all her releases ...

Question What happened in 1983?

Figure 2.2: A reading comprehension examples. The answer is extracted as a span from the document (underlined).

ponent to synthesize the final answer.

Existing work has explored several variations for each component and different strategies for training them jointly and separately (Chen et al., 2017; Wang et al., 2018b; Das et al., 2019; Clark and Gardner, 2018; Swayamdipta et al., 2018, inter alia).

For example, the DrQA system (Chen et al., 2017) adopts a pipeline approach in which the retrieval component ranks Wikipedia articles according to their TF-IDF (Salton and Buckley, 1987) matching with the question. The paragraphs of the top-5 ranked documents construct D'. The reading components is a neural network model (detailed below) that assigns a score to each span in the given paragraph. The span with highest score over all paragraphs in D' is outputted as the answer.

2.2.1 Machine Reading Comprehension

Ample research has focused on extractive machine reading comprehension (MRC). Several large datasets (Rajpurkar et al., 2016; Joshi et al., 2017; Yang et al., 2018; Dua et al., 2019; Rodriguez et al., 2019) and neural models (Seo et al., 2017; Yu et al., 2018a; Devlin et al., 2019) have been proposed.

In the MRC setup, a question q of m tokens q_1, q_2, $\ldots$, q_m together with a document d of n tokens t_1, t_2, $\ldots$, t_n are given as input to a model that output a span of the paragraph tokens as the answer (Figure 2.2). QA accuracy is often measured using two metrics that compare the predicted answer span and the gold span—(1) exact match (EM) and (2) unigram overlap measured by the macro-averaged F1 score (Rajpurkar et al., 2016).

One intuitive model for MRC is DocumentReader of the DrQA system (Chen et al., 2017) in which an answer start/end score is estimated for each token in d. The span with largest start and end scores is produced as the answer. The model starts by encoding each token i in d as a vector $\tilde{t}_i$ that is constructed by concatenating its pre-trained GLoVE (Pennington et al., 2014) word embedding and indicator features of its part-of-speech tag, named-entity tag and whether the token has a matching token in the question. The token encodings $\tilde{t}_1$, $\tilde{t}_2$, $\ldots \tilde{t}_n$ are fed to a multi-layer bidirectional long short-term memory network (BILSTM) (Section 2.1.2) whose output representations t_1, t_2, $\ldots t_n$ are used as contextualized embeddings of the document tokens. Question tokens are encoded using their corresponding GLoVE embeddings and fed through another BILSTM whose output representations are averaged to compute a vector encoding of the question q.

To estimate a score for each span in d, each token i is assigned a score for being a start/end of the answer span as

$$\begin{aligned} Start(i) &= \exp({t_i}^T W_{start} q); \\ End(i) &= \exp({t_i}^T W_{end} q), \end{aligned} \tag{2.7}$$

where W_{start} and W_{end} are trained parameters. The span $t_{[i,j]}$ with the largest $Start(i) \times End(j)$ is predicted as the answer. The model is trained by maximizing the log likelihood of the gold spans.

Pre-trained transformers (Section 2.1.6) have introduced significant accuracy boosts on several MRC benchmarks (Devlin et al., 2019; Alberti et al., 2019; Beltagy et al., 2020). For example, BERT is used for MRC by simply appending the document tokens to the question tokens and feeding them through the pre-trained network. The network produces a vector t_i encoding of each token t_i in the document conditioned on the other tokens and the question. A span $t_{[i,j]}$ in the document is assigned a score $Start(i) \times End(j)$, where

$$\begin{aligned} Start(i) &= \exp({t_i}^T w_{start}); \\ End(i) &= \exp({t_i}^T w_{end}), \end{aligned} \tag{2.8}$$

and w_{start} and w_{end} are the only parameters trained from scratch on MRC-specific datasets.

2.2.2 Conversational Reading Comprehension

In information-seeking dialogs, e.g., personal assistants, users interact with a QA system by asking a sequence of related questions. The QA system needs to link questions together to resolve the conversational dependencies between them: each question needs to be understood in the conversation context. For example, the system cannot answer the question 'Did she have any other children?' without context that tells what 'she' refers to and 'which child' is already mentioned.

Document: Anna Vissi

In May 1983, she married Nikos Karvelas, a composer, with whom she collaborated in 1975 and in November she gave birth to her daughter Sofia. After their marriage, she started a close collaboration with Karvelas. Since 1975, all her releases ...

Question 1: What happened in 1983?

Answer 1: she married Nikos Karvelas

Question 2: Did they have any children?

Answer 2: In November she gave birth to her daughter Sofia

Question 3: Did she have any other children?

Answer 3: Cannot answer.

Figure 2.3: Conversational Reading Comprehension Example from QuAC

Existing research introduces datasets and models for conversational QA under various setups including retrieval-only (Dalton et al., 2019), reading-only (Choi et al., 2018; Reddy et al., 2019; Qu et al., 2019b; Huang et al., 2019), open-domain (Elgohary et al., 2018) and QA over structured knowledge base (Saha et al., 2018; Guo et al., 2018; Yu et al., 2019b,a). In conversational reading comprehension, a single pre-specified document is used to answer all questions with potentially unanswerable questions that models need to identify as part of the task.

Models for conversational reading comprehension are typically trained on a large dataset of questions, each is paired with its gold answer, an evidence document, and a conversation context (previous questions and their answers). For instance, Reddy et al. (2019) tweaked the DocumentReader model (described in Section 2.2.1) by simply prepending the conversation context to the question. Alternatively, Huang et al. (2019) and Qu et al. (2019a) explicitly model the interaction between the conversation history, the question and reference documents.

In this thesis, I use two datasets for conversational QA, QuAC (Choi et al., 2018) and QBLINK (Elgohary et al., 2018)[2]. The construction of QuAC involves using a pair of workers—a "student" and a "teacher"—to ask and respond to questions. The "student" asks questions about a topic based on only the title of the Wikpedia article and the title of the target section. The "teacher" has access to the full Wikipedia section and provides answers by selecting text that answers the question. With this methodology, QuAC gathers 98k questions across 13,594 conversations. Figure 2.3 shows an example on such setup from QuAC.

QBLink is a conversational QA dataset for the open-domain setup in which questions in the same conversation are linked by shared entities and relations that connect those entities. For example a question whose answer is 'Bitcoin' is followed by a question that asks about the 'inventor of Bitcoin'. A full example in shown in Figure 2.4. In that setup, QA systems need to model entities and relations in both their retrieval and the reading components. QBLink consists of 18,000 question sequences, each sequence consists of three naturally occurring human-authored questions (totaling around 56,000 unique questions) collected from previously asked questions in Quiz Bowl tournaments.

In Chapter 4, I approach CQA by rewriting questions in stand-alone form. Both Ras-

[2]Elgohary et al. (2018) present two contributions: (1) The dataset (work done by Chen Zhao) and (2) Models (my contributions) which I present in Chapter 4.

Lead-in: Only twenty-one million units in this system will ever be created. For 10 points each:

Question 1: Name this digital payment system whose transactions are recorded on a "block chain".

Answer: Bitcoin

Question 2: Bitcoin was invented by this person, who, according to a dubious Newsweek cover story, is a 64-year-old Japanese-American man who lives in California.

Answer: Satoshi Nakamoto

Question 3: This online drugs marketplace, Chris Borglum's one-time favorite, used bitcoins to conduct all of its transactions. It was started in 2011 by Ross Ulbricht using the pseudonym Dread Pirate Roberts.

Answer: Silk Road

Figure 2.4: An example sequence of questions from QBLink. The lead-in and question 1 are asking about the same object/answer. The subject of question 2 is the same as the object of question 1. All questions are about a narrow topic, Bitcoin.

togi et al. (2019) and Su et al. (2019) introduce utterance rewriting datasets for dialog state tracking. Rastogi et al. cover a narrow set of domains and the rewrites of Su et al. are based on Chinese dialog with two-turn fixed histories. In contrast, the conversational question rewriting dataset I introduce in Chapter 4 has histories of variable turn lengths and covers more topics. Training question rewriting using reinforcement learning with the task accuracy as a reward signal is explored in retrieval-based QA (Liu et al., 2019a) and in MRC (Buck et al., 2018). A natural question is whether reinforcement learning could learn to retain the necessary context to rewrite questions in CQA. My dataset could be used to pre-train a question rewriter that can further be refined using reinforcement learning.

2.3 Question Answering over Structured Databases

Data can already exist in a structured format, e.g., as relational tables or in the form of entities ('Anna Vissi', 'Nikos Karvelas', '1983') linked with relations ('spouse', 'marriage_date'). Such structures often have their own formal query languages (a query is also referred to as an executable logical form) that are able to perform complex processing over the data and compute/retrieve an output result (e.g., SQL, SPARQL) .

QA systems over structured data either generate a logical form (semantic parsing) whose execution result answers the question (Berant et al., 2013; Zhong et al., 2017) or use information extraction methods that link each input question to a set of knowledge base entities and score their related entities based on their relevance to the relations mentioned in the question (Yao and Van Durme, 2014; Dong et al., 2015; Xu et al., 2016). In this work, I focus on the former and review recent neural models for logical form generation then, I provide specific details for parsing questions into SQL queries.

2.3.1 Semantic Parsing Models

SEQ2SEQ models (Section 2.1.5) are used for semantic parsing where the question is used as the input sequence and the (linearized) logical form the the output to be generated (Dong and Lapata, 2016; Iyer et al., 2017; Zhong et al., 2017). A copy mechanism is often employed as values in the question such as 'Anna Vissi' and '1983' in 'who did Anna Vissi marry in 1983?' also appear in the logical form, e.g.,

$$\lambda x : marrige\_date(x, \text{“Anna Vissi”}, 1983)$$

Token-level SEQ2SEQ models struggle as the logical forms to be generated become more complex. Since, by definition, logical forms adhere to specific syntax, existing research considers augmenting the generation process with syntactic knowledge (Rabinovich et al., 2017; Yin and Neubig, 2017; Su et al., 2017; Yu et al., 2018b). For example, Yin and Neubig (2017) develop a general syntax-guided generation scheme is which logical forms are replaced with their corresponding abstract syntax trees which are linearized to a sequence of production rules each either comes from the underlying grammar of the target logical forms or a terminal generation. Their model is then trained to generate such production rules instead of individual tokens which guarantees that the outputs are syntactically-valid and simplifies the generation process as much less decoding steps are needed to reach the final output. Other work (Su et al., 2017; Yu et al., 2018b) decompose the generation process into multiple components that are pre-defined based on the syntax of the target logical forms.

2.3.2 Semantic Parsing with Synthetic Data

Semantic parsing systems have frequently used synthesized data to alleviate the challenge of labeled data scarcity. In their semantic parser overnight work, Wang et al. (2015) proposed a method for training semantic parsers quickly in a new domain using synthetic data. They generate logical forms and canonical utterances and then paraphrase the canonical utterances via crowd-sourcing. Several other works have demonstrated the benefit of adopting this approach to train semantic parsers in low-resource settings (Su et al., 2017; Zhong et al., 2017; Cheng et al., 2018; Shah et al., 2018; Xu et al., 2020). Another line of work has proposed using synthesized data to adapt semantic parsing (Jia and Liang, 2016; Yoo et al., 2019; Campagna et al., 2019) and task-oriented dialogues (Campagna et al., 2020) models to new domains. Most recently, synthetic data was used to continue pre-training language models (Section 2.1.6) for semantic parsing tasks (Herzig et al., 2020; Yu et al., 2021a,b). In Chapter 6, I build on this line work and generate synthetic data automatically without human involvement to simulate user feedback on erroneous semantic parses. I use that synthetic data to improve the accuracy on my semantic parse correction models.

2.3.3 Semantic Parsing in Conversational Context

Semantic parsing often backs conversational agents in which related questions are asked sequentially. Existing work study the problem of inducing semantic parses for utterances in conversational contexts (Zettlemoyer and Collins, 2009; Iyyer et al., 2017; Suhr et al., 2018; Yu et al., 2019b,a; Zhang et al., 2019; Andreas et al., 2020). Two modeling aspects are studied in that task: (1) how to interpret an utterance in a given context and (2) how to reuse generated parses for previous utterances as utterances in the same conversations are often highly related. For example, Zhang et al. (2019) add a copy from the previous parse mechanism to a SEQ2SEQ parsing model.

In Chapter 5, I introduce the task of semantic parse correction with natural language feedback. While conversational semantic parsing focuses on modeling conversational dependencies between questions, the task I introduce evaluates the extent to which models can interpret and apply feedback on the generated parses. In Section 5.4, I empirically confirm the distinction between the two tasks.

2.3.4 Text-to-SQL Parsing

In Chapters 5 and 6, I focus on one form of structured data, namely relational databases where the logical form to be generated is SQL (also referred to as text-to-SQL and natural language interfaces to databases (Androutsopoulos et al., 1995)).

Several datasets of natural language utterances (questions) with corresponding SQL annotations are presented. Unlike other datasets that are small scale (e.g., GeoQuery (Popescu et al., 2003), domain specific (e.g., ATIS (Iyer et al., 2017)) and limited to only simple SQL (e.g., WikiSQL (Zhong et al., 2017)), Yu et al. (2018c) construct SPIDER—a large dataset of more than 10,000 utterance-SQL pairs covering 200 databases in 138 domain. SPIDER contains SQL queries of varying complexities including nested queries, group by and join operations. A single sample from SPIDER is a triplet of utterance, database schema structure (table/column names and primary/foreign keys) and the corresponding SQL annotation. An example is shown in figure 2.5. Large-scale conversational text-to-SQL datasets, namely SPaRC and CoSQL, based on SPIDER are introduced in (Yu et al., 2019b) and (Yu et al., 2019a).

SPIDER adopts a splitting by database scheme (Finegan-Dollak et al., 2018) where there are no shared databases between its training, development and testing splits. Hence, models not only need to generalize to unseen utterances, but also need to generalize to unseen databases (cross-domain generalization). The primary evaluation metric used in SPIDER is the exact match accuracy between the predicted and gold parses. Instead of string matching the parses, Yu et al. provide a script that ignores the ordering infor-

Table 1: instructor – **Columns:** id, name, department_id, salary, ...

Table 2: department – **Columns:** id, name, building, budget, ...

Foreign keys: (instructor.department_id, department.id)

Question: What are the name and budget of the departments with average instructor salary greater than the overall average?

SQL: `SELECT T2.name, T2.budget FROM instructor as T1 JOIN department as T2 ON T1.department_id = T2.id GROUP BY T1.department_id HAVING avg(T1.salary) > (SELECT avg(salary) FROM instructor)`

Figure 2.5: A text-to-SQL example from SPIDER. Primary keys are underlined

mation (e.g., `‘SELECT first_name, last_name’` is the same as `‘SELECT last_name and first_name’`) that does not affect the correctness of the generated parses.

It is worth noting that the exact match measure can introduce false negatives as it does not really compare the semantics of the queries. For instance, it fails to match `SELECT name FROM people WHERE age > 30` and `SELECT name FROM people WHERE age >= 31`. Alternatively (or additionally), existing work (Suhr et al., 2020; Deng et al., 2021) also compares the denotations (execution results) of two queries to decide if they are semantically equivalent or not. However, the execution accuracy can also be misleading as it can introduce false positives—two semantically different queries can produce the same result on a particular database. Most recently, Zhong et al. (2020) proposed to address the false positives problem by automatically generating multiple different databases (a test suite) that would distinguish semantically different queries when executed against at least one of them.

Suhr et al. (2020) pointed out limitations with SPIDER and proposed more realistic evaluation setups. Most of the time, SPIDER utterances explicitly mention the column/table names that should appear in the inferred query. Also, the queries and databases used for training and testing are of similar structures and complexities. Suhr et al. (2020) proposed a cross-dataset evaluation setup in which models are trained on SPIDER and evaluated on other datasets (e.g., ATIS and GeoQuery). More recently, Deng et al. (2021) created a modified test set in which they avoid mentioning the exact form of the column/table names in the input utterance to be parsed.

Both token-level SEQ2SEQ (Dong and Lapata, 2016; Zhong et al., 2017; Zhang et al., 2019) and syntax-guided models have been applied to text-to-SQL (Yu et al., 2018b; Dong and Lapata, 2018; Shin et al., 2019). Another line of modeling has focused on better encoding the input schema (Bogin et al., 2019) and linking schema items to the given utterance (Guo et al., 2019; Wang et al., 2020). Also, Yin et al. (2020); Yu et al. (2021a); Deng et al. (2021) introduced customized approaches for better pre-training transformer models as a means to improve the text-to-SQL parsing accuracy.

In Chapters 5 and 6, I use four text-to-SQL parsers for my experiments: **SEQ2STRUCT** (Shin, 2019) encodes the schema and the input utterance as a graph and employs a variant of self-attention that encodes preexisting relations between columns, tables and the tokens on the input utterance. For the decoder, they use the syntax-guided decoder of Yin and Neubig (2017). **EDITSQL** (Zhang et al., 2019) computes an initial encoding of the utterance and the schema using BERT , then it computes updated representations by passing the output of BERT through BILSTMs (Section 2.1.2) combined with self-attention. They use an LSTM-based decoder that generates a SQL query one token at a time (autoregressive).

TaBERT (Yin et al., 2020) is a pre-trained transformer that computes a joint encoding of an utterance and a set of tables. For text-to-SQL, it is paired with the syntax-guided decoder of Yin and Neubig (2017). **RAT-SQL** (Wang et al., 2020) encodes the utterance and the schema first with BERT then, it passes the output of BERT through a relation-aware-transformer (Section 2.1.3) to hard-code relations between tables, columns and tokens of the input utterance. It also uses the syntax-guided decoder of Yin and Neubig (2017).

2.4 Human-in-the-Loop NLP

The standard machine learning pipeline consists of three steps: (1) annotators construct a training dataset, (2) a model is trained, and (3) the model is deployed to make predictions. Previous work in NLP study how to best leverage human abilities/collective intelligence in each of those steps.

2.4.1 Humans as Annotators

Active learning (Settles, 2010) is applied for iteratively selecting useful examples to provide to human annotators in various NLP tasks (Tang et al., 2002; Olsson, 2009; Ambati et al., 2010). To enable crowdsourcing complex annotations for semantic role labeling, FitzGerald et al. (2018) study simplifying the annotation process by posing simple question to the workers (e.g., "who proposed something?" whose answer is the agent of the predict "proposed"). Another line of related work is avoiding artifacts in datasets (that often encourage models to learn undesirable shortcuts rather than the intended task) by having human annotators play the role of an adversary against a baseline model that detects/exploits artifacts they produce while generating training examples (Zellers et al., 2018; Wallace et al., 2019; Dasigi et al., 2019; Dua et al., 2019).

2.4.2 Humans as Teachers

Previous work study having humans as teachers who provide implicit or explicit feedback to improve models over time. Agichtein et al. (2006) use clickthrough and browsing information to improve search engine ranking. Werling et al. (2015) study deploying an untrained system and keep querying realtime crowdworkers about uncertain parts of the output (e.g., the name-entity tag of a word) until the system is confident about the output. Crowdworkers' responses are used to improve the model over time, and hence reduce the reliance on the crowdworkers. Iyer et al. (2017) consider the the task of text-to-SQL parsing. In their setup, for an input question the model's generated SQL is executed and a human judges the results as correct/wrong. SQLs corresponding to correct judgments are added as new examples to the training data and correct SQLs are crowdsourced for the mispredicted examples and also added to the training data. The model is retrained peroidically.

Reinforcement learning is a natural framework for incorporating human feedback to improve an initial model over time. Nguyen et al. (2017) study improving machine translation systems with human rating feedback (e.g., star rating). Liu et al. (2018) improve task-oriented dialog systems by having humans provide the correct dialog actions to the model or provide rating feedback (e.g., thumb up on successful task completion). Ling and Fidler (2017) consider the task of caption generation for images. They take natural language feedback on the generated caption and use it to improve the model with reinforcement learning. Their model learns to turn the natural language feedback into numeric rewards assigned to the phrases of generated captions. Li et al. (2017a) consider learning a QA model over textual assertions from numeric reward and/or textual feedback in a *synthetic* setup where the questions, assertions and the possible textual feedbacks are automatically generated. Further, the feedback provider (human teacher) is assumed to know the correct answer and the provided feedback is limited to indicating whether the predicted answer matches the gold answer or not. In a similar synthetic environment in which a QA model would need assistance with predefined scenarios (e.g., the input question contains misspelled words), Li et al. (2017b) use reinforcement learning to learn a policy that decides when to ask for assistance (e.g., ask for a rephrasing of a misspelled question). The policy is optimized using positive rewards received when providing a correct answer and a negative reward (cost) received when asking for assistance. Lawrence and Riezler (2018) improve a text to Overpass (query language for OpenStreetMap) semantic parser using historic logs of human feedback on generated parses. The feedback is collected using a graphical user interface that maps an Overpass query to predefined blocks. Humans provide feedback by marking each block as correct/incorrect which is translated into token-level positive/negative reward and used afterwards to refine an initial parser.

2.4.3 Humans as Collaborators

At prediction time, humans are involved as collaborators who work with a model to improve the output result. One form of human involvement at prediction time is answering followup clarification questions about an underspecified/ambiguous question in QA settings (Rao and Daumé III, 2018; Saeidi et al., 2018; Xu et al., 2019; Yao et al., 2019a). The challenge addressed in that line of work is deciding what information is missing, how to ask about it and how to use the provided answers to make the final prediction.

2.4.3.1 Interactive Semantic Parsing

More related to this thesis, humans are also involved at prediction time to refine the initial outputs of semantic parsing models. A growing body of recent work demonstrates that semantic parsing systems can be improved by including users in the parsing loop—giving them the affordance to examine the parses, judge their correctness, and provide feedback accordingly. He et al. (2016b) ask simplified questions about uncertain dependencies in CCG parsing and use the answers as soft constraints to regenerate the parse. Both Li and Jagadish (2014) and Narechania et al. (2021) generate SQL queries and present them in graphical user interfaces that humans can control to edit the initial logical forms. Similarly, Su et al. (2018) enable users to edit inferred RESTful API calls through a graphical user interfaces. Gur et al. (2018) ask specific predefined multiple choice questions about a narrow set of predefined parsing errors. This interaction model together with the synthetically generated erroneous parses that are used for training can be appropriate for simple text-to-SQL parsing instance as in WikiSQL, which was the only dataset used for evaluation. Yao et al. (2019b) ask yes/no and multiple-choice grammar-based questions (e.g., "Does the system need to return average of values in temperature?") about SQL components (e.g., `‘‘SELECT avg temperature’’`) while generating a SQL parse one component at a time for an input utterance. Labutov et al. (2018) study correcting semantic parses with natural language feedback, but they consider only the domains of email and biographical research.

The work I present in Chapters 5 and 6 falls under the category of collaborative human–machine NLP. A distinct characteristic of my work is that it aims at a more *realistic* and *natural* involvement of humans. I do not assume that humans know the execution results of the predicted parses, but rather humans engage in a conversation with the machine until the correct result is obtained. Another unique characteristic to my work is the reliance on natural language as the interface between the human and the machine. While processing the natural language interactions of humans is by itself a challenging language understanding problem, natural language gives humans more flexibility, richness and naturalness in their interactions with machines.

In the following fours chapters, I present the details of the contributions of this thesis. I start in next chapter with demonstrating the effectiveness of using natural language as a source of learning. I use naturally occurring Spanish translation of English text to train general-purpose sentence representations.

Chapter 3: Learning Paraphrastic Representations with Bilingual Supervision

The effectiveness of new representation learning (Section 2.1.1) methods for distributional word representations (Baroni et al., 2014) has brought renewed interest to the question of how to compose semantic representations of words to capture the semantics of phrases and sentences. These representations offer the promise of capturing phrasal or sentential semantics in a general fashion, and could in principle benefit any NLP applications that analyze text beyond the word level, and improve their ability to generalize beyond contexts seen in training. As this thesis studies human–computer communication (interaction) through natural language, in this chapter I demonstrate the richness and cost-effectiveness of using naturally-occurring natural language supervision to learn vector representations of sentences and phrases.

Wieting et al. (2016) recently showed that a simple composition architecture (vector averaging) can yield sentence models that consistently perform well in semantic textual similarity tasks in a wide range of domains, and outperform more complex sequence models

The work in this chapter is published as "Ahmed Elgohary and Marine Carpuat. 2016. Learning monolingual compositional representations via bilingual supervision. In Proceedings of the Association for Computational Linguistics" (Elgohary and Carpuat, 2016).

(Tai et al., 2015; Kiros et al., 2015). Interestingly, these models are trained using PPDB, the paraphrase database (Ganitkevitch et al., 2013), which was learned from bilingual parallel corpora. In bilingual settings, models that capture the semantics of sentences are typically only evaluated on cross-lingual transfer tasks such as cross-lingual document categorization or machine translation (Hermann and Blunsom, 2014), which do not directly evaluate the quality of the sentence-level semantic representations learned.

In this chapter, I directly evaluate the usefulness of modeling semantic equivalence using compositional models of translated texts for detecting semantic textual similarity *in a single language*. For instance, in addition to using translated texts to model cross-lingual transfer from English to a foreign language, we can view text in the foreign language as a semantic annotation of the English text, and evaluate the usefulness of the resulting English representations. Compared to (Wieting et al., 2016), I avoid the intermediate step of learning monolingual paraphrases, and train my models directly on the naturally-occurring parallel text. I show that learning from bilingual supervision yields sentence representations that outperform those learning with monolingual supervision on several semantic textual similarity tasks.

3.1 Models

Following the bilingual model of Hermann and Blunsom (2014), and paraphrase model of Wieting et al. (2016), representations for multi-word segments are built with a simple bag-of-word additive combination of word representations, which are trained to minimize the distance between semantically equivalent segments.

3.1.1 Learning Objective

I use the same learning objective of the bilingual composition model (BICVM) (Hermann and Blunsom, 2014). The goal is to learn a word embedding matrix W from a training set of aligned (i.e., assumed to be semantically equivalent) pairs of multi-word segments $\langle x_1, x_2 \rangle$. Each of x_1 and x_2 is represented as a bag-of-words, i.e., a set of column indices in W. Each aligned pair $\langle x_1, x_2 \rangle$ is augmented with k randomly selected segments that are not aligned to x_1, and another k that are not aligned to x_2. Given that augmented example $\langle x_1, x_2, \bar{x}_1^1, ..., \bar{x}_1^k, \bar{x}_2^1, ..., \bar{x}_2^k \rangle$, the training objective is defined as follows:

$$
\begin{aligned}
J_{bicvm}(W) = \frac{\lambda}{2}||W||_F^2 + \sum_{\langle x_1, x_2, \bar{x}_1, \bar{x}_2 \rangle} \sum_{i=1}^{k} \\
[\delta + ||g(x_1) - g(x_2)||^2 - ||g(x_1) - g(\bar{x}_2^i)||^2]_h + \\
[\delta + ||g(x_1) - g(x_2)||^2 - ||g(x_2) - g(\bar{x}_1^i)||^2]_h
\end{aligned}
\tag{3.1}
$$

where $g(x) = \sum_{i \in x} W_{:i}$, $[.]_h$ is the hinge function (i.e., $[v]_h = \max(0, v)$) whose margin is δ, and λ is a regularization parameter.

3.1.2 Three Views of Semantic Equivalence

I compare three views of semantic equivalence (examples in Table 3.1):

Monolingual paraphrases are invaluable resources, but rarely occur naturally, and creating paraphrase resources requires considerable effort (Ganitkevitch et al., 2013).

Parallel (Bilingual) sentences occur naturally and provide training examples that are more consistent with downstream applications. However, they can be noisy due to

Bilingual Sentences	+	thus, in fact, we might say that he hurried ahead of the decision by our fellow member	as que podemos decir, de hecho, que se adelanta la decisión de nuestro colega
	-	thus, in fact, we might say that he hurried ahead of the decision by our fellow member	señor presidente, la votación sobre sellafield ha sido una novedad en el parlamento europeo
English Paraphrases	+	by our fellow member	by our colleague
	-	by our fellow member	of the committee's work
	+	slowly than anticipated	slowly than expected
Bilingual Paraphrases	+	by our fellow member	de nuestro colega diputado
	-	by our fellow member	miles de personas de todo
	+	book and buy airline tickets	reserva y adquisición de billetes
	+	the air fare advertised should show	el precio del billete anunciado deberá indicar
	+	a book by the american writer noam	un libro del escritor norteamericano noam

Table 3.1: Positive and negative examples for each of the three types of supervision

Condition	**# Examples**	**Avg. Length**	**Provenance**
Bilingual Sentences	1.9M	28	Europarl-v7
Bilingual phrases	3M	5	+ Moses phrase extraction
Monolingual phrases	3M	3	PPDB XL

Table 3.2: Sources of the three types of semantic equivalence.

automatic sentence alignment, and bag-of-word representations of sentence meaning are likely to be increasingly noisier as segments get longer.

Bilingual phrases (phrasal translations) might provide a tighter definition of semantic equivalence than longer sentence pairs, but phrase pairs have to be extracted automatically based on word alignments, an automatic and noisy process.

For bilingual sentences, I use the English-Spanish corpus from Europarl-v7 (Koehn, 2005) and for the monolingual paraphrases I use the XL distribution of PPDB which contains three million pairs. To extract bilingual phrases, I run the phrase extraction implementation of Moses (Koehn et al., 2007) on the same English-Spanish corpus I use as the bilingual sentences. Moses produces phrase pairs ranked by confidence. I limit the phrase pairs I train with to the first three million. Table 3.2 summarizes the three sources of training data.

3.2 Experiments

3.2.1 Evaluating Sentence Representations

Following Wieting et al. (2016), the models above are evaluated on four semantic textual similarity (STS) benchmarks (Agirre et al., 2012, 2013, 2014, 2015), which provide pairs of English sentences from different domains (e.g., tweets, news, web forums, image captions) annotated with human judgments of similarity on a one to five scale. Additionally, the Sentences Involving Compositional Knowledge (SICK) test set (Marelli et al., 2014) provides a complementary evaluation. It consists of sentence pairs annotated with semantic relatedness scores. While STS examples were simply drawn from existing NLP datasets, SICK examples were constructed to avoid non-compositional phenomena such as multiword expressions, named entities and world knowledge.

For each test set, systems have to output a similarity score for each sentence pair. I construct sentence-level embeddings by averaging the representations of words in each sentence, and compute cosine similarity to capture the similarity between the two sentences of each pair. Systems are then evaluated using the Pearson correlation between gold and predicted scores.

3.2.2 Baselines and Reference Models

The PARAGRAM objective of Wieting et al. (2016) shares the same structure as the BICVM objective I use (Equation 3.1), but differs in the nature of segments used to define semantic equivalence (monolingual paraphrases), the distance function used (cosine distance rather than the Euclidean distance), the negative sampling strategies, and word embeddings initialization and regularization. As reference points, I report evaluation results for the PARAGRAM embeddings as well as the GLOVE embeddings (Pennington et al., 2014) in which word embeddings were trained with a non-composition objective.

The PARAGRAM embeddings were initialized with high-quality but resource intensive embeddings, and were regularized to penalize deviations from GLOVE embeddings. For a fair comparison with the BICVM objective (in which the embeddings are initialized randomly), I use a simplified variant of PARAGRAPH objective, $J_{paragram}$ in which the learned embeddings are initialized randomly and are regularized with the frobenius norm ($||W||_F^2$ in Equation 3.1).

3.2.3 Training and Hyper-parameters Tuning

At training time, I learn word embeddings for each combination of objective (J_{bicvm} and $J_{paragram}$) and type of training examples (Table 3.2), using modified versions of the open-source implementations provided by the authors of BICVM[1] and PARAGRAM[2]. This results in six model configurations. Each was trained for 10 epochs using tuned hyper-parameters.

I use the SMT-europarl subset of STS-2012 as the development set. I consider mini-batches of size $\{25, 50, 100\}$, $\delta \in \{1, 10, 100\}$ for Euclidean distance, $\delta \in \{0.4, 0.6, 0.8\}$ for cosine distance, $\lambda \in \{1, 10^{-3}, 10^{-5}, 10^{-7}, 10^{-9}\}$, and $k \in \{1, 5, 10, 15\}$. To speed up the tuning of $J_{paragram}$, I follow (Wieting et al., 2016) and limit the training to five epochs on $100k$ examples.

[1]https://github.com/karlmoritz/bicvm

[2]https://github.com/jwieting/iclr2016

	Monolingual Phrases		**Bilingual Phrases**		**Bilingual Sentences**		**Reference Results**	
	J_{bicvm}	$J_{paragram}$	J_{bicvm}	$J_{paragram}$	J_{bicvm}	$J_{paragram}$	Paragram	GloVe
MSRpar	0.28	0.42	**0.54**	0.38	**0.54**	0.36	0.44	0.47
MSRvid	0.33	0.55	**0.71**	0.38	**0.71**	0.19	0.77	0.64
SMT-eur	0.39	0.41	**0.49**	0.46	0.47	0.47	0.48	0.46
SMT-news	0.40	0.50	**0.59**	0.40	0.58	0.38	0.63	0.50
OnWN	0.52	0.57	**0.64**	0.62	0.46	0.62	0.71	0.55
2012 Avg	0.39	0.49	**0.59**	0.45	0.54	0.41	0.61	0.53
headline	0.56	0.66	**0.70**	0.58	0.66	0.61	0.74	0.64
OnWN	0.55	0.53	**0.75**	0.34	0.48	0.25	0.72	0.63
FNWN	0.35	0.29	**0.41**	0.32	0.25	0.16	0.47	0.34
2013 Avg	0.49	0.49	**0.62**	0.41	0.46	0.34	0.58	0.42
deft forum	0.35	0.47	**0.51**	0.36	0.36	0.33	0.53	0.27
deft news	0.59	0.68	**0.77**	0.59	0.76	0.58	0.75	0.68
headline	0.56	0.63	**0.73**	0.58	0.67	0.58	0.72	0.60
images	0.58	**0.73**	**0.73**	0.59	0.66	0.49	0.80	0.61
OnWN	0.65	0.62	**0.80**	0.55	0.55	0.47	0.81	0.58
tweet news	0.59	0.66	**0.73**	0.64	0.56	0.69	0.77	0.51
2014 Avg	0.55	0.63	**0.71**	0.55	0.59	0.52	0.73	0.54
forums	0.35	0.42	**0.55**	0.48	0.50	0.45	0.66	0.31
students	0.66	0.66	**0.73**	**0.73**	0.65	0.69	0.77	0.63
headline	0.64	0.60	**0.79**	0.64	0.73	0.66	0.76	0.62
belief	0.46	**0.71**	0.68	0.67	0.48	0.61	0.77	0.41
images	0.52	0.71	**0.75**	0.62	0.67	0.56	0.82	0.68
2015 Avg	0.53	0.63	**0.70**	0.63	0.59	0.60	0.76	0.53
SICK	0.53	0.62	**0.66**	0.57	0.63	0.54	0.72	0.66

Table 3.3: Pearson correlation scores obtained on the English STS sets (with per year averages) and on semantic-relatedness task SICK. The left columns report results based on new representations learned in this work, while the two rightmost columns report reference results from prior work (Wieting et al., 2016).

3.3 Results

Table 3.3 reports the Pearson correlation for each approach and dataset.

3.3.1 Bilingual phrases yield the best models in controlled settings

Overall, the best representations are obtained using bilingual phrase pairs and the J_{bicvm} objective. They produce the highest correlation score among all compositional models for all tasks, except for one subset of STS-2015.

The best objective for a given type of training example varies: $J_{paragram}$ generally yields higher correlation scores with monolingual phrases, while J_{bicvm} performs better with bilingual examples. Bilingual phrases seem to benefit from larger number of randomly selected negative samples and from using the Euclidean distance rather than cosine distance. The best bilingual compositional representations are better than non-compositional GLOVE , but worse than compositional PARAGRAM. However, PARAGRAM initialization requires large amounts of text and human word similarity judgments for tuning (Wieting et al., 2015), while our models were initialized randomly.

3.3.2 Bilingual sentences vs. bilingual phrases

Why do bilingual phrases outperform the bilingual sentences they are extracted from? In this section, I verify that this is not explained by systematic biases in the distribution of training examples.

First, Table 3.4 shows that bilingual sentences have the smallest ratios of under-trained words, and are therefore not penalized by rare words more than bilingual phrases. Further, more than 80% of the words that appear in both bilingual sentences and bilingual

Dataset	**Monolingual Phrases**	**Bilingual Phrases**	**Bilingual Sentences**
2012 Avg	0.15	0.17	0.09
2013 Avg	0.16	0.17	0.11
2014 Avg	0.19	0.22	0.11
2015 Avg	0.15	0.19	0.11
SICK	0.2	0.25	0.15

Table 3.4: Undertrained word ratios (ratio of tokens seen fewer than 100 times during training) are uncorrelated with performance in Table 3.3.

phrases occur in 460 (on average) more bilingual sentences than in bilingual phrases. The remaining 20% were found to be the rare words (e.g. zazvorkova, woldesmayat, yellow-bellies) that hardly occur in test sets.

Second, Table 3.5 confirms that the ranking of the three training sets is not biased due to memorization of the phrases seen during training. The ranking does not change when testing on unseen word sequences, as shown by SICK results with models trained using J_{bicvm} on a filtered training set that contains none of the bigrams observed at test time.

Third, the advantage of bilingual phrases over bilingual sentences is not due to the larger number of training examples. $1.9M$ (and even $1M$) bilingual phrase pairs still outperform the $1.9M$ bilingual sentence pairs on all subsets (Table 3.6).

Taken together, these additional results support my initial intuition that the main advantage of bilingual phrases over bilingual sentences is that phrase pairs have stronger

	Not Filtered		**Filtered**	
	# Pairs	**Score**	**# Pairs**	**Score**
Monolingual Phrases	3M	0.52	2.3M	0.54
Bilingual Phrases	3M	**0.67**	2.1M	**0.65**
Bilingual Sentences	1.9M	0.66	0.47M	0.58

Table 3.5: Impact of memorization: Pearson correlation scores on SICK with training sets with and without filtering out segment pairs that contain any bigrams that appear in SICK. Number of training pairs (# Pairs) is shown in millions.

	Bilingual Phrases				**Bilingual Sentences**	**GLoVE**
	0.5*M*	1*M*	1.9*M*	3*M*	*1.9M*	
2012	0.55	0.58	0.59	0.59	0.54	0.53
2013	0.59	0.61	0.61	0.62	0.46	0.42
2014	0.69	0.71	0.71	0.71	0.59	0.54
2016	0.68	0.69	0.70	0.70	0.61	0.53
SICK	0.62	0.64	0.65	0.66	0.63	0.66

Table 3.6: Impact of training set size: Average Pearson correlation per test set with different numbers (in millions) of bilingual phrase pairs, compared to the full set of bilingual sentences and monolingually pretrained GLoVE.

semantic equivalence than sentence pairs, since phrase pairs are shorter and are constructed by identifying strongly aligned subsets of sentence pairs.

3.3.3 Monolingual versus bilingual phrases

Based on the analysis thus far, I hypothesize that paraphrase pairs with overlapping tokens make the compositional training objective less useful. Around 40% of the monolingual paraphrase training pairs differ only by one token. With Euclidean distance in the training objective, overlapping tokens cancel each other out of the composition term. For example, the pair ⟨*healthy and stable, healthy and steady*⟩ yields the compositional term

$$\begin{aligned}&||(\mathit{healthy} + \mathit{and} + \mathit{stable}) - (\mathit{healthy} + \mathit{and} + \mathit{steady})||_2\\ &= ||\mathit{stable} - \mathit{steady}||_2\end{aligned}$$

In contrast, overlap cannot occur in the bilingual setting, and all words within bilingual phrases contribute to the compositional objective.

3.4 Conclusion

I conducted the first evaluation of compositional representations learned using bilingual supervision on monolingual textual similarity tasks. Phrase and sentence representations are constructed by composing word representations using a simple additive composition function. The resulting English sentence representations consistently outperform compositional models trained to detect monolingual paraphrases on five different English semantic textual similarity benchmarks.

Bilingual phrase pairs are consistently the best evidence of semantic equivalence in my experiments. They yield better results than the sentence pairs they are extracted from, despite the noise introduced by the automatic extraction process. Furthermore the composed representations outperform non-compositional word representations derived from monolingual co-occurrence statistics (GLoVE). While sizes of monolingual and bilingual corpora are not directly comparable, it is remarkable that representations learned with only 500k bilingual phrase pairs outperform GLoVE embeddings trained on 840B tokens.

Having demonstrated the effectiveness of learning from natural language (bilingual) supervision for learning sentence representations, in the next chapter I switch to studying how humans can engage in a conversation (again, entirely natural language-based) with machines. I conduct that research in the context of conversational question answering: I present models for conversational open-domain QA that focus on reasoning about the connections between different questions in the same conversation. Then, I present a more explicit task that focuses on understanding questions in conversational contexts.

Chapter 4: Sequential Question Answering and Question-in-Context Rewriting

The high level goal of this thesis is to create tools, systems, and models for collaborative human–computer question answering and NLP in general. In particular, I focus on human–computer interaction via natural language. A natural language-based interactive system should be able to understand human utterances in conversational contexts. In this chapter, I study conversational question answering and create models and a dataset that focus on implicitly (Section 4.1) and explicitly (Section 4.2) interpreting questions in the context of previously asked questions and their answers.

First, I study the task of sequential open-domain QA (Section 4.1). I ask how a standard open-domain QA system can incorporate connections between question-answer

The work in this chapter is published as "Ahmed Elgohary, Chen Zhao, and Jordan Boyd-Graber. 2018. Dataset and baslines for sequential open-domain question answering. In Proceedings of Empirical Methods in Natural Language Processing" (Elgohary et al., 2018) and "Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019. Can you unpack that? Learning to rewrite questions-in-context. In Proceedings of Empirical Methods in Natural Language Processing" (Elgohary et al., 2019). Elgohary et al. (2018) presents two contributions: (1) The construction of the QBLINK dataset (Section 2.2.2) which is work done by Chen Zhao, and (2) Baseline models which are my contributions that I present in Section 4.1 of this chapter.

pairs in the same sequence. I introduce context-aware models models for the retrieval and reading components, and demonstrate that context information can improve the QA accuracy.

Next, I switch to more naturally conversational settings, in which individual questions cannot be understood without resolving conversational dependencies. I study rewriting such questions in independent stand-alone forms (Section 4.2) which enables solving conversational QA with stand-alone MRC models (Section 2.2.1). I create CANARD—**C**ontext **A**bstraction: **N**ecessary **A**dditional **R**ewritten **D**iscourse—a new dataset that rewrites QuAC (Choi et al., 2018) (Section 2.2.2) questions given conversational context (previous questions and answers). I present and evaluate models for the rewriting task.

4.1 Sequential Open-Domain QA

4.1.1 Dataset and setup

Given a document collection D and questions grouped into sequences $\{S_i \,|\, i = 1 \ldots n\}$ where each S_i is an ordered sequence of question-answer pairs, and a subset of documents $S_i = ((q_i^j,\ a_i^j,\ D_i^j) \,|\, j = 1 \ldots m)$, the task is to answer questions $q_i^{\hat{j}}$ with document evidences $D_i^{\hat{j}}$ given access to previously asked questions in the same sequence and their corresponding answers $\{(q_i^j,\ a_i^j) \,|\, j < \hat{j}\}$. To study that task, I use the QBLink dataset (Section 2.2.2) in which each sequence consists of three questions that start with a lead-in which sets the stage for the rest of the question.

I build my models on the DrQA framework (Section 2.2.1) that uses Wikipedia as the document collection. Following that framework, I split the task into two separate steps—a retrieval step and a reading step. In the retrieval step the current question $q_i^{\hat{j}}$

and previous questions and answers $\{(q_i^j,\ a_i^j)\,|\,j < \hat{j}\}$ are used to retrieve a ranked list of paragraphs $D_i^{\hat{j}}$ from D that are likely to contain the correct answer to the current question $q_i^{\hat{j}}$. The retrieved paragraphs $D_i^{\hat{j}}$ are the input to the reading step that selects a span from $D_i^{\hat{j}}$ as the answer to $q_i^{\hat{j}}$. The reading step has access to previous questions and answers $\{(q_i^j,\ a_i^j)\,|\,j < \hat{j}\}$ as well.

4.1.2 Models

Answering Questions in Isolation:

I experiment with three baselines that ignore the sequential connections between questions and answer each question in isolation. My first model is a simple information retrieval (IR) baseline that only uses the retrieval component: the title of the top-1 Wikipedia article is predicted as the answer. My second baseline is the full DrQA model whose reader is trained/tuned on the QBLINK training/development questions. To assign paragraphs to each of the training questions, I follow a similar distant-supervision approach to (Chen et al., 2017). I retrieve the top twenty Wikipedia articles for each question, exclude the paragraphs that do not contain the gold answer, and then rank the remaining paragraphs using TF-IDF. Each of the top ten paragraphs is paired with the question to form a data instance for training the reader. Finally, I tweak the DrQA reader to limit the candidate answer spans to entity mentions that are linked to Wikipedia. I set the start and end scores (Equations 2.7) of spans that are not detected mentions to zero.

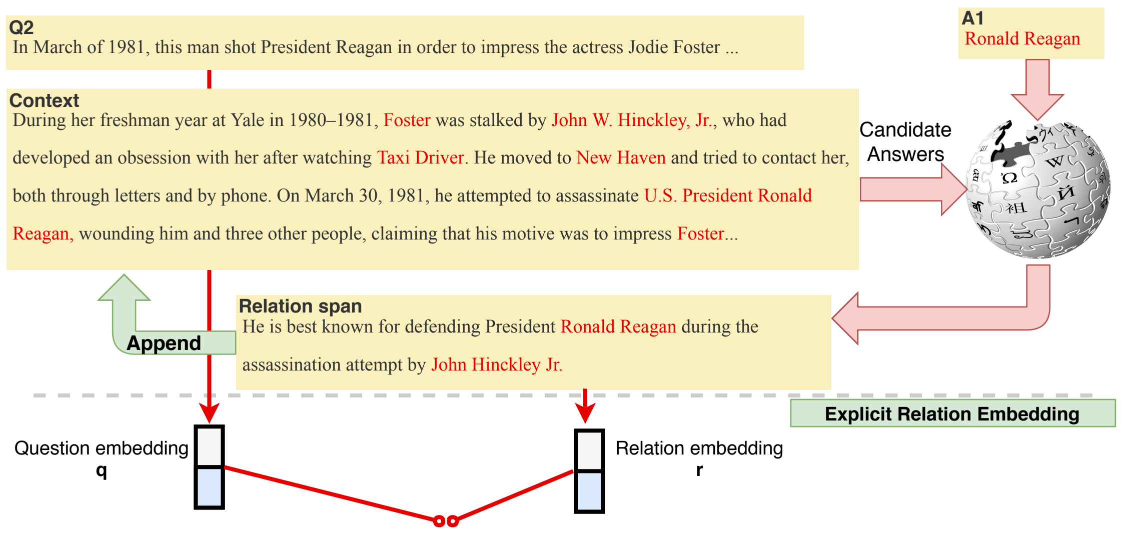

Q2
In March of 1981, this man shot President Reagan in order to impress the actress Jodie Foster ...
A1
Ronald Reagan
Context
During her freshman year at Yale in 1980–1981, Foster was stalked by John W. Hinckley, Jr., who had developed an obsession with her after watching Taxi Driver. He moved to New Haven and tried to contact her, both through letters and by phone. On March 30, 1981, he attempted to assassinate U.S. President Ronald Reagan, wounding him and three other people, claiming that his motive was to impress Foster...
Candidate Answers
Append
Relation span
He is best known for defending President Ronald Reagan during the assassination attempt by John Hinckley Jr.
Explicit Relation Embedding
Question embedding
q
Relation embedding
r

Figure 4.1: An overview of the relation-augmented reading model for sequential QA.

Incorporating Context in Retrieval:

To incorporate the sequential connections between questions in the retrieval phase, I append the previously asked questions to the current question. I also compare appending the predicted answers (top-1 span) to each of the previous questions as well as the gold answers to the current question.

Incorporating Context in Reader:

In addition to encoding which entities have appeared in previous questions, it might be useful to provide the models with *relationship* information. However, pre-defined relationships from knowledge bases tend to be brittle. Instead, I use a continuous representation of relationships (Iyyer et al., 2016). Suppose we want to encode the relationships for an entity (answer candidate) that starts at token i and ends at token j. I summarize the relationships of that entity from k possible relation-spans. A relation-span is a sequence of tokens from Wikipedia that contains both the answer candidate and an answer to a previous question. For example, the correct answer 'John W. Hinckley Jr.' in Figure 4.1 has a relation-span *"He is best known for defending President Ronald Reagan during the assassination attempt by John Hinckley Jr."* with the previous answer *"Ronald Reagan"*.

I compare two methods for incorporating such relation spans. In the first method, I append relation spans to the document and rely on the reading network to implicitly infer the relation encoded by the span and use it to score the answers. Alternatively, I merge all k relation-spans in a single span that is then fed through a BiLSTM (Section 2.1.2) whose hidden states are combined as a weighted sum with self-attention (Lin et al., 2017) to form an explicit vector encoding of the relation r_{ij}.

The stronger the similarity between the relation that the question is asking about and the relation-spans, the higher the score of the candidate answer should be. I estimate the similarity r by concatenating the elementwise absolute difference and Hadamard product between r_{ij} and the question embedding q. I then use a trainable weight vector w_{rel} to combine the components of the concatenation and produce a single similarity score

$$r = {w_{rel}}^T [|q - r_{ij}|; q \circ r_{ij}].$$

This influences the final selection of the answer span by adding the relation similarity score r to the start and end scores of the candidate answer (Equation 2.7),

$$\begin{aligned} Start(i) &= \exp({t_i}^T W_{start} q + r) \\ End(j) &= \exp({t_j}^T W_{end} q + r). \end{aligned} \tag{4.1}$$

The relation embedding module is trained jointly with the reader.

Method	**EM**
Baselines: Questions in Isolation	
IR	15.6
DrQA	39.3
DrQA + limiting to entities	39.7
DrQA + Retrieval with context	
Previous questions	36.4
Previous predicted answers	39.8
Previous gold answers	40.1
DrQA + Reading with context	
Append relation descriptions w/ predicted answers	40.2
Append relation descriptions w/ gold answers	40.7
Explicit relation embedding w/ predicted answers	38.3
Explicit relation embedding w/ gold answers	39.5
IR w/ Previous gold answers + **Reading** w/ Append relation descriptions w/ gold answers	40.7

Table 4.1: Incorporating sequence information in the retrieval and the reading step slightly improves overall accuracy compared to answering questions in isolation.

4.1.3 Experiments

I use the Wikipedia dump of 2017–09–20. I set the maximum number of retrieved documents to ten, and split each document into paragraphs of 400 tokens each. At test

Question: This man attempted to impress Jodie Foster by shooting Ronald Reagan, but he failed to kill the President. At trial, he was found not guilty by reason of insanity.

Gold answer to previous question: Ronald Reagan

Predict without relation span: George H. W. Bush

Correct answer: John Hinckley Jr.

Relation span: He is best known for defending President Ronald Reagan during the assassination attempt by John Hinckley Jr.

Figure 4.2: Modeling the relation between *President Ronald Reagan* and *John Hinckley Jr.* expressed by relation span helps the reader select the correct answer entity.

time, I merge the top ten ranked such paragraphs and feed them to the reader (following Clark and Gardner (2018)). I use the reader network of DrQA, and I limit the number of relation description spans for each entity pair to five. I use an LSTM of one hidden layer and 128 hidden units for the paragraph, question, and relation description encoders. Each reader is trained for twenty epochs.

4.1.4 Results

Table 4.1 summarizes the results. Question-answering accuracy is exact-match accuracy since I limit the answer spans to entity mentions whose boundaries are fixed for all models.

Incorporating the previous answer in the retrieval and the reading components slightly improves the overall question answering accuracy The accuracy drops by more than 3% when using the entire text of previous questions in the retrieval phase. Modeling relations reduces the accuracy slightly compared to augmenting paragraphs with relation spans. One possible explanation is that my relation embedding model is under-trained because many questions lack relevant relation-spans. Replacing Wikipedia with a larger corpus (e.g., ClueWeb) or improving reference detection might improve relation embedding model. Unsurprisingly, gold answers to previous questions are more useful than the predicted answers, which highlights a need for models that take into account the uncertainty about previous answers when gold previous answers are not available.

Figure 4.2 gives an example of how explicit relation embedding helps reader get a correct prediction. Without the relation span, the model predicts George H. W. Bush (vice president at that time) as correct answer. Including the direct relation span between Reagan and John Hinckley Jr., the model gets the correct answer.

4.2 Question-in-Context Rewriting

In this section, I study reducing challenging, interconnected conversational question answering in reading comprehension setup (CQA) examples to independent, stand-alone questions. I start by formally defining the rewriting task. Then, I describe the construction of CANARD in which I crowdsource context-independent paraphrases of QuAC questions and use the paraphrases to train and evaluate question-in-context rewriting.

4.2.1 Defining Question-In-Context Rewrites

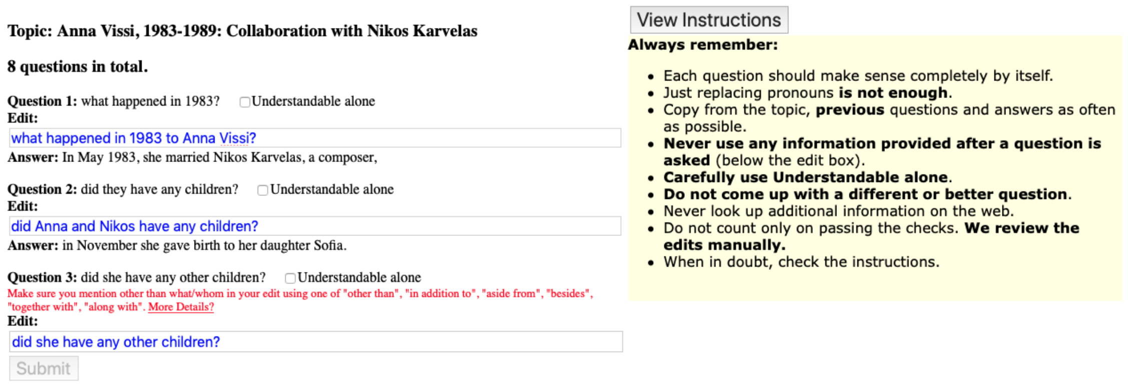
I formally define the task of question-in-context rewriting (de-contextualization). Given a conversation topic t and a history H of $m-1$ turns, each turn k is a question q_i and an answer a_i; the task is to generate a rewrite q'_m for the next question q_m based on H. Since q_m is part of the conversation, its meaning often involves references to parts of its preceding history. A valid rewrite q'_m should be self-contained: a correct answer to q'_m by itself is a correct answer to q_m combined with the question's preceding history H.

Figure 1.3 shows the assumptions of CQA and how they are made explicit in rewrites. The first question omits the title of the page (Anna Vissi), the second question omits the answer to the first question (replacing both Anna Vissi and her husband with the pronoun "they"), and the last question adds a scalar implicature that must be resolved.

4.2.2 Dataset Construction

I elicit paraphrases from human crowdworkers to make previously context-dependent questions *unambiguously* answerable. Through this process, I resolve difficult coreference linkages and create a pair-wise mapping between ambiguous and context-enriched questions. I take the entire dev set of QuAC and a sample of the train set and create a custom JavaScript task in Mechanical Turk that allows workers to rewrite these questions. JavaScript hints help train the users and provides automated, real-time feedback.

I provide workers with a comprehensive set of instructions (Figure 4.4) and task examples. I ask them to rewrite the questions in natural sounding English while preserving the sentence structure of the original question. I discourage workers from introducing new words that are unmentioned in the previous utterances and ask them to copy phrases when


Topic: Anna Vissi, 1983-1989: Collaboration with Nikos Karvelas

8 questions in total.

Question 1: what happened in 1983? ☐ Understandable alone
Edit:
what happened in 1983 to Anna Vissi?
Answer: In May 1983, she married Nikos Karvelas, a composer,

Question 2: did they have any children? ☐ Understandable alone
Edit:
did Anna and Nikos have any children?
Answer: in November she gave birth to her daughter Sofia.

Question 3: did she have any other children? ☐ Understandable alone
Make sure you mention other than what/whom in your edit using one of "other than", "in addition to", "aside from", "besides", "together with", "along with". More Details?
Edit:
did she have any other children?

Always remember:

- Each question should make sense completely by itself.
- Just replacing pronouns **is not enough**.
- Copy from the topic, **previous** questions and answers as often as possible.
- **Never use any information provided after a question is asked** (below the edit box).
- **Carefully use Understandable alone**.
- **Do not come up with a different or better question**.
- Never look up additional information on the web.
- Do not count only on passing the checks. **We review the edits manually.**
- When in doubt, check the instructions.

Figure 4.3: CANARD collection interface. The conversation has eight questions in total. I show one question at a time to encourage crowdworkers only use the *previous* utterances for the rewriting. After the eight questions are rewritten, I enable submitting the HIT. We show question-specific instructions as in question three to remind crowdworkers about relevant instructions. A compact set of the full instructions are shown on the right. Detailed instructions can be displayed by clicking on the "view instructions" button.

appropriate from the original question. These instructions ensure that the rewrites only resolve conversation-dependent ambiguities. Thus, I encourage workers to create minimal edits; in Section 4.2.3, I take advantage of this to use BLEU (Papineni et al., 2002) for evaluating model-generated rewrites.

I display (Figure 4.3) the questions in the conversation one at a time, since the rewrites should include only the previous utterance. After a rewrite to the question is submitted, the answer to the question is displayed. The next question is then displayed. This repeats until the end of the conversation.

Figure 4.4: Instructions for crowdsourcing question-in-context rewrites.

Rewrite each question in a dialogue so it can be understood by itself without looking at the rest of the dialogue.

1. In the given conversation between a student who asks questions about a topic of a Wikipedia article and a teacher who is answering the questions, each question can be understood as part of the ongoing dialogue. Edit each question so that it makes sense by itself without looking at the rest of the dialogue. Think how you would ask the same question to someone who has no idea about what the dialogue is about and has never seen any of the previous utterances and still get the same answer. For example, the question 'What was he like in that episode?' cannot be understood without knowing what 'he' and 'that episode' refer to. Both can be figured out by looking at the dialogue. If we edited the question to 'What was Daffy Duck like in Porky's Duck Hunt?', it would then make sense by itself.

2. Reading and understanding all previous utterances is very important. Do not just look for things like 'she' and 'that' and replace them. For example, the question Who was the star? is still unclear. We need to specify 'star in what movie, series, episode, ..etc.?' as in Who was the star in Porky's Duck Hunt?. To make sure the edits are based on understanding the conversations rather than naive replacements, we will manually review a random group of the edits before accepting the work. We also run pretty mild checks on the edits before allowing the HIT to be submitted. The checks are meant to help you make valid edits, but they still can miss some invalid edits (rather than being too strict). So, do not count only on passing the checks and make sure that each edit makes sense by itself.

3. Questions should stay in a natural-sounding english after editing. Do not use new words other than those in the conversation unless it is necessary. Copy from previous questions, answers and conversation topic as often as possible. Never use any information provided after a question is asked. Only use what was provided up to the question you are editing. When editing a question, keep its original structure whenever possible. For example, do not change 'Did they arrest him?' to 'Was Alex arrested?', but rather keep the original structure and do 'Did they arrest Alex?'. Do not come up with a different or better question. Stick to the given wording. It is okay to have pronouns in an edit as long as the edit has enough information about what the pronouns refer to. For example 'Where was Carroll when she joined the american party?' is a perfectly fine edit.

4. Pay special attention to questions that ask if there is anything/anyone else or asking for telling more about something. It is important to mention in your edit "anything/anyone else besides what/ whom". For example, 'Are there any other interesting aspects about this article?' should be edited to something like 'Besides Death of a Ladies Man and End of the Century, are there any other interesting aspects about this article?' as in example 2 below. Notice that it is okay to keep 'this article' without replacing it with something else.

5. If a question can be understood without looking at the rest of the conversation, mark it as Understandable alone. Do not make unnecessary edits. Try to keep the edits short as long as they make sense each by itself. Although typos are quite rare, please fix any typos you encounter.

6. Never look up additional information on the web. Only use the given conversations.

7. We show one question at a time. After editing or marking a question as Understandable alone, we show its following question. After all questions are edited, we allow you to submit the HIT and optionally provide any feedback you have on the HIT.

I apply quality control throughout my collection process. During the task, JavaScript checks automatically monitor and warn about common errors: submissions that are abnormally short (e.g., 'why'), rewrites that still have pronouns (e.g., 'he wrote this album'), or ambiguous words (e.g., 'this article', 'that'). Many QuAC questions ask about 'what/who else' or ask for 'other' or 'another' entity. For that class of questions, I ask workers to use a phrase such as 'other than', 'in addition to', 'aside from', 'besides', 'together with' or 'along with' with the appropriate context in their rewrite.

I gather and review the data in batches to screen potentially compromised data or low quality workers. A post-processing script flags suspicious rewrites and workers who take and abnormally long or short time. I flag about 15% of the data. *Every* flagged question is manually reviewed and an entire HIT is discarded if one is deemed inadequate. I reject 19.9% of submissions and the rest comprise CANARD. Additionally, I filter out under-performing workers based on these rejections from subsequent batches. To minimize risk, I limit the initial pool of workers to those that have completed 500 HITs with over 90% accuracy and offer competitive payment of $0.50 per HIT.

I verify the efficacy of my quality control through manual review. A random sample of fifty questions sampled from the final dataset is reviewed for desirable characteristics by a native English speaker in Table 4.2. Each of the positive traits occurs in 90% or more of the questions. Based on the sample, the collected rewrites retain grammaticality, leave the question meaning unchanged, and use pronouns unambiguously. There are rare occasions where workers use a part of the answer to the question being rewritten or where some of the context is left ambiguous. These infrequent mistakes should not affect my models. I provide examples of failures in Table 4.3.

I use the rewrites of QUAC's development set as my test set (5,571 question-in-context

Characteristic	**Ratio**
Answer Not Referenced	0.98
Question Meaning Unchanged	0.95
Correct Coreferences	1.0
Grammatical English	1.0
Understandable without Context	0.90

Table 4.2: Manual inspection of 50 rewritten context-independent questions from CANARD suggests that the new questions have enough context to be independently understandable.

and corresponding rewrite pairs) and use a 10% sample of QUAC's training set rewrites as the development set (3,418); the rest are training data (31,538).

4.2.3 Models

I compare three baseline models for the question-in-context rewriting task. In the **Copy** baseline, the rewrite q'_m is set to be the same as the input question q_m without making any changes. I also try a **Pronoun Substitution** baseline in which the first pronoun in q_m is replaced with the topic entity of the conversation. I use the title of the corresponding Wikipedia article to the original QUAC conversation as the topic entity.

Unlike the previous baselines which do not use the rewrites as training data, the third baseline is a neural SEQ2SEQ model (Section 2.1.5) with a copy mechanism (Bahdanau et al., 2015; See et al., 2017). I construct the input sequence by concatenating all utterances in the history H, prepending them to q_m, and adding a special separator token between

ORIGINAL: Was this an honest mistake by the media?
REWRITE: Was the claim of media regarding Leblanc's room come to true?
ORIGINAL: What was a single from their album?
REWRITE: What was a single from horslips' *album*?
ORIGINAL: Did they marry?
REWRITE: Did Hannah Arendt and Heidegger marry?

Table 4.3: Not all rewrites correctly encode the context required to answer a question. We take two failures to provide examples of the two common issues: Changed Meaning (top) and Needs Context (middle). We provide an example with no issues (bottom) for comparison.

utterances.

I use a bidirectional LSTM (Section 2.1.2) encoder-decoder model with shared the word embeddings between the encoder and the decoder (the OpenNMT (Klein et al., 2018) implementation). I initialize the embeddings with GLoVE (Pennington et al., 2014) and train with a batch-size of 16 for 200000 steps.

Since questions are written by humans, human rewrites are the upper-bound for this task. However, annotators (especially crowdworkers) can be inconsistent or disagree. To estimate the human accuracy, I collect 100 pairs of rewritten questions; each pair has two rewrites of the same question (in its given context) by two different workers. I manually verify that all rewrites are valid and then use the pair of rewrites as a hypothesis and a reference.

	Dev	**Test**
Copy	33.84	36.25
Pronoun Sub	47.72	47.44
Seq2Seq	51.37	49.67
Human Rewrites*	59.92	

Table 4.4: BLEU scores of the baselines on development and test data. SEQ2SEQ improves up to four (statistically significant) points over naive baselines but still well (statistically significant) below human accuracy. Human accuracy (*) is computed with a small subset of the development set.

Table 4.4 shows the BLEU scores produced by the baselines and humans over both the validation and the test sets.[1] Although a well-trained standard neural SEQ2SEQ model improves 2–4 BLEU points over naive baselines, it is still 9 BLEU points below human-accuracy. Using paired bootstrap resampling (Efron and Tibshirani, 1994; Koehn, 2004) with 1000 samples from the test set, SEQ2SEQ outperforms the Copy and Pronoun Sub baselines with 95% statistical significance. Also, using bootstrap resampling I found that human rewrites outperforms the SEQ2SEQ baseline with 95% statistical significance. Since human accuracy is estimated with a different test set (100 examples) than the CANARD test set (5,571 examples), I repeated the following procedure 1000 times: 1) Sample with replacement 100 examples from the human rewrites, 2) Sample with replacement 5,571 examples from the SEQ2SEQ outputs for the CANARD test set, 3) Compare the BLEU scores of the two samples. In more than 95% of such comparisons, the BLEU score of the human

[1]I use multi-bleu-detok.perl (Sennrich et al., 2017).

Label	**Text**
QUESTION	How long did he stay there?
REWRITE	How long did Cito Gaston stay at the Jays?
HISTORY	*Cito Gaston* **Q:** What did Gaston do after the world series? ... **Q:** Where did he go in 2001? **A:** In 2002, he was hired by the Jays as special assistant to president and chief executive officer Paul Godfrey.

Table 4.5: An example that had over ten flagged proper nouns in the history. Rewriting requires resolving challenging coreferences.

rewrites was larger than that of the SEQ2SEQ output. I analyze sources of errors in the following section.

4.2.4 Dataset and Model Analysis

I analyze CANARD with automatic metrics after validating the reliability of the data collection pipeline. I compare CANARD to the original QUAC questions and to automatically generated questions by my models. Then, I manually inspect the sources of rewriting errors in the SEQ2SEQ baseline.

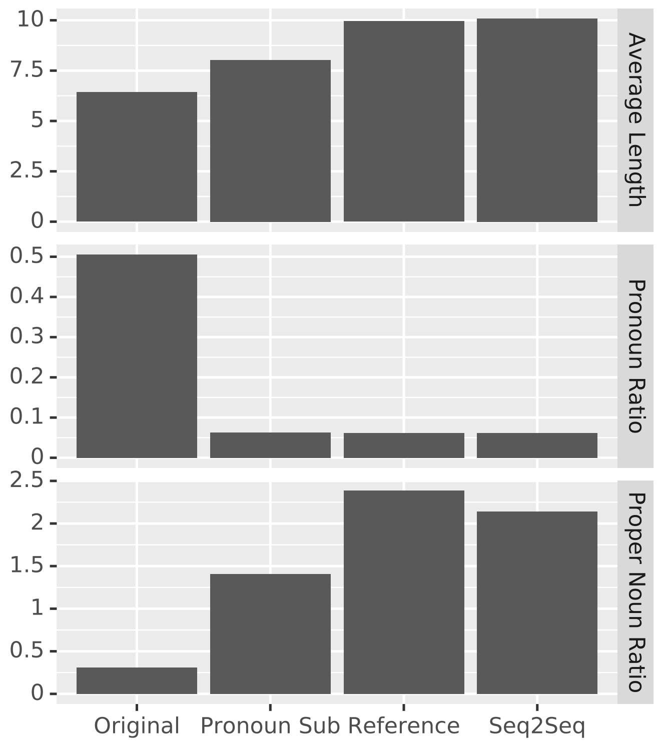


Figure 4.5: Human rewrites are longer, have fewer pronouns, and have more proper nouns than the original QUAC questions. Rewrites are longer and contain more proper nouns than the Pronoun Sub baseline and trained SEQ2SEQ model.

Data Analysis:

CANARD rewrites are longer, contain more nouns and less pronouns, and have more word types than the original data. Machine output lies in between the two human-generated corpora (original question and reference rewrite), but quality is difficult to assess. Figure 4.5 shows these statistics.

I motivate CANARD rewrites by exploring linguistic properties of the rewrites. Coreference resolution is a core NLP task applicable to this dataset in addition to the downstream tasks evaluated in Section 4.2.3. Pronouns occur in 53.9% of QUAC questions. Questions

Correctness Category	**%**
Correct: Slightly different from reference	42
Correct: Significantly different from reference	16
Incomplete rewrite	16
Incorrect rewrite	26

Table 4.6: Summary of manually comparing the outputs of the SEQ2SEQ model to the reference rewrites. In 58% of the cases, the model produces correct rewrites that are sometimes (16%) significantly different from the reference. Examples on each category are in Table 4.7.

with pronouns are more likely to be ambiguous than those without any. Only 0.9% of these have pronouns that span more than one category (e.g., 'she' and 'his'). Hence, pronouns within a single sentence are likely unambiguous. However, 75.0% of the aggregate history has pronouns and the percentage of mixed category pronouns increase to 27.8% of the data. Therefore, pronoun disambiguation potentially becomes a problem for a quarter of the original data. An example is provided in Table 4.5. Approximately one-third of the questions generated by my pronoun-replacement baseline are within 85% string similarity to the golden rewritten questions. That leaves two-thirds of the data that cannot be solved with pronoun resolution alone.

Model Analysis:

By manually examining the SEQ2SEQ rewrites of a sample of 100 test examples and comparing them to the reference rewrites, I find (Table 4.6) that 58% of the generated rewrites are correct (fully independent and have the same meaning as the in-context question). In 42% of the sample, the generated rewrite exactly matches the reference with possibly slight differences (e.g., using the last name of person rather than the full name). In 16% of the sample, the generated rewrite is still correct but significantly different from the reference in e.g., structure or the used lexical items. I also notice that 16% of the rewrites are not fully independent (e.g., with an unresolved coreference) and 26% of the rewrite are fully independent but still incorrect as they do not have the same meaning as the in-context questions (e.g., because of a wrong coreference resolution). Examples on each category are listed in Table 4.7.

Table 4.7: Examples on each of the rewrite correctness categories of the SEQ2SEQ model.

History	**SEQ2SEQ Rewrite**	**Reference Rewrite**
Correct: Slightly different from reference		
Wallace Shawn ||| Acting ||| What is he famous for acting in ?	What is Wallace Shawn famous for acting in ?	What is Shawn famous for acting in ?
Marc Chagall ||| France (1923-1941) ||| What was Marc doing in France in 1923 ? ||| In 1923 , Chagall left Moscow to return to France . ||| What was he looking to do once he returned ? ||| On his way he stopped in Berlin to recover the many pictures he had left there on exhibit ten years earlier, before the war began, ||| What kind of pictures he was into or did ?	What kind of pictures Marc Chagall was into or did ?	What kind of pictures was Marc into or did ?

Correct: Significantly different from reference		
Mario Winans ||| Story of My Heart (1995-97) ||| Were any other popular artists featured on the album ? ||| I don't know . ||| What were some singles off of the album ? ||| the song "Don't Know ||| Was this his first album? ||| his debut album Story of My Heart on ||| Did he win any awards for the album ?	Did Mario Winans win any awards for Story of My Heart?	Did Mario Winans win any awards for Story of My Heart?
Zenyatta ||| Retirement ||| When was retirement? ||| On November 17, 2010, Zenyatta was slated to be retired. ||| Was she injured ? ||| I do n't know. ||| Did she retire at that time ? ||| On February 23 , 2011 , she traveled by van to Darley Stud , where she was bred to Bernardini . ||| Are there any other interesting aspects about this article ?	Aside from retirement are there any other interesting aspects about this article ?	Aside from retirement are there any other interesting aspects about this article ?
Incomplete rewrite		

James Baldwin ||| Religion ||| Did Baldwin come from a religious family? ||| During his teenage years, Baldwin followed his stepfather's shadow into the religious life. ||| Was he happy with his feelings on religion? ||| However , he became dissatisfied with ministry, considering it hypocritical and racist, and ultimately left the church ||| What steps did he take after leaving the church? ||| At the age of 14 he attended meetings of the Pentecostal Church and , during a euphoric prayer meeting ||| Did he like that church ?	Did James Baldwin like that church?	Did James Baldwin like the Pentecostal church ?
Charles Lindbergh ||| Flight ||| When did Lindbergh start flying ? ||| In the early morning of Friday, May 20, 1927 , Lindbergh took off from Roosevelt Field across the Atlantic Ocean for Paris , France . ||| Did he make any stops along the flight ?	Did Lindbergh make any stops along the flight from Roosevelt Field ?	Did Charles Lindbergh make any stops along the flight ?
Incorrect rewrite		

Anna Politkovskaya ||| The murder remains unsolved , 2016 ||| Did they have any clues? ||| probably FSB) are known to have targeted the webmail account of the murdered Russian journalist Anna Politkovskaya. ||| How did they target her email? ||| On 5 December 2005, RFIS initiated an attack against the account annapolitovskaya@US Provider1, by deploying malicious software ||| Did they get into trouble for that ?	Did Anna Politkovskaya get into trouble for?	Did FSB get into trouble for attacking the email account of Anna Politkovskaya's?
David Vitter ||| Opposition to Franken amendment ||| What is the Franken amendment ? ||| bill that would forbid federal contractors from forcing victims of sexual assault , battery and discrimination to submit to binding arbitration ||| How did it proceed ?	How did forbid federal contractors proceed?	How did Franken amendment proceed?

Also, I notice that the main source of errors of the SEQ2SEQ model is that it tends to find a short path to completing the rewrites. That often results in *under-specified questions* as in Example 1 in Table 4.8, *question meaning change* as in Example 2 or *meaningless questions* as in Example 3. Another source of errors is having related entities mentioned in the context as Example 4 in Table 4.8, where the model confused "Copa America" with

	SEQ2SEQ output	**Reference**
1	What did Chamberlain's men do?	What did Chamberlain's men do during the Battle of Gettysburg?
2	How many games did Ozzie Smith win?	How many games did the Cardinals win while Ozzie Smith played?
3	Did 108th get to the finals?	Did the US Women's Soccer Team get to the finals in the 1999 World Cup?
4	Did Gabriel Batistuta reside in any other countries, besides touring in the Copa America?	Besides Argentina, did Gabriel Batistuta reside in any other countries?
5	Did La Comedia have any more works than La Comedia 3?	Did Giannina Braschi have any more works than United States of Banana, La Comedia and Asalto al tiempo?

Table 4.8: Example erroneous rewrites generated by the Seq2Seq models and their corresponding reference rewrites. The dominant source of error is the model tendency to produce short rewrites (Examples 1–3). Related entities (Copa America and Argentina in Example 4) distract the model. The model struggles with listing multiple entities mentioned in different parts of the context (Example 5).

"Argentina". The model also struggles with listing multiple entities mentioned in different parts of the context. Example 5 in Table 4.8 show the output and the reference rewrites of the question *"Did she have any more works than those 3?"*, where two of the three entities—“United States of Banana”, “La Comedia” and “Asalto al tiempo”—are lost in the rewrite.

4.3 Conclusion

I evaluated baselines for sequential open-domain QA using the QBLINK dataset. I showed that incorporating sequential information helps improve QA accuracy. Then, I introduced a new task and a dataset for question-in-context rewriting. I compared and analyzed baselines for the rewriting task, and show that there is still a gap between out top-performing model and human performance.

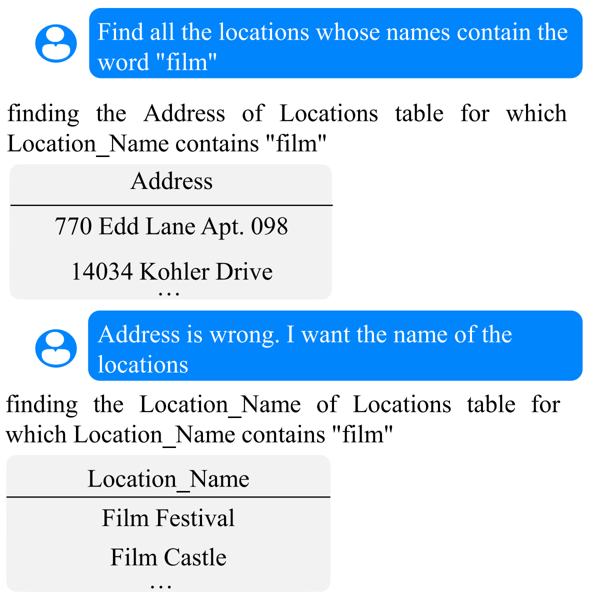
Overall, I demonstrated in this chapter that humans can conduct natural language conversations with machine and explored avenues for improving the ability of machines to understand context-dependent human utterances. In the next two chapters, I introduce an interactive system for text-to-SQL parsing. The system collaborates with human users to reach the correct SQL parse of a user provided utterance. All communication between the user and the system is conducted in natural language.

Chapter 5: Semantic Parsing with Natural Language Feedback

In the previous two chapters, I demonstrate the effectiveness of learning with natural language and I create and evaluate models for understanding natural language inputs in conversational contexts. In this and the following chapter, I present a system that puts all such components together to improve question answering accuracy through interacting with users who provide natural language feedback to the system in the context of their conversation with the system. In particular, the system answers natural language questions about structured data in a relational (SQL) database by parsing questions into SQL queries (Section 2.3.4).

Most of the work addressing the text-to-SQL problem and semantic parsing (Section 2.3) in general frames it as a one-shot mapping task. I establish (Section 5.3.1) that the majority of parsing mistakes that recent neural text-to-SQL parsers make are minor. Hence, it is often feasible for humans to detect and suggest fixes for such mistakes. Su et al. (2018) make a similar observation about parsing text to API calls and show that parsing mistakes could be easily corrected if humans have the opportunity to provide precise feedback. Likewise, an input utterance might be under- or mis-specified, thus extra

The work in this chapter is published as "Ahmed Elgohary, Saghar Hosseini, and Ahmed Hassan Awadallah. 2020. Speak to your parser: Interactive text-to-SQL with natural language feedback. In Proceedings of the Association for Computational Linguistics" (Elgohary et al., 2020).



Find all the locations whose names contain the word "film"

finding the Address of Locations table for which Location_Name contains "film"

Address
770 Edd Lane Apt. 098
14034 Kohler Drive
…

Address is wrong. I want the name of the locations

finding the Location_Name of Locations table for which Location_Name contains "film"

Location_Name
Film Festival
Film Castle
…

Figure 5.1: An example of human interaction with a text-to-SQL system to correct the interpretation of an input utterance. The system generates an initial SQL parse, explains it in natural language, and displays the execution result. Then, the system uses the human-provided natural language feedback to correct the initial parse.

interactions may be required to generate the desired output as in query refinements in information retrieval systems (Dang and Croft, 2010).

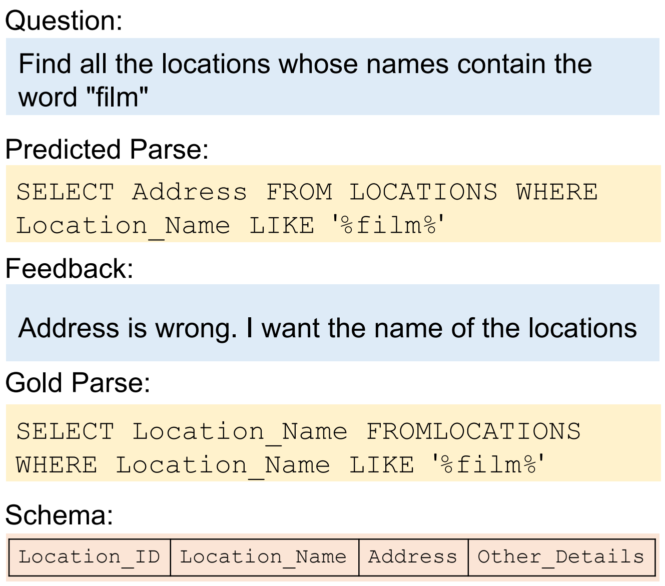
This chapter studies the task of SQL parse correction with natural language feedback to enable text-to-SQL systems to seek and leverage human feedback to further improve the overall performance and user experience. Figure 5.1 shows an example of a text-to-SQL system that collects feedback in natural language when the system misinterprets an input utterance. To enable this type of interactions, the system needs to: (1) provide an *explanation* of the underlying generated SQL, (2) support a means for humans to provide feedback, and (3) use the feedback along with the original question, to come up with a more accurate interpretation.

Towards that goal, I make the following contributions: (1) I define the task of SQL parse correction with natural language feedback (Section 5.1); (2) I create a framework for explaining SQL parses in natural language to allow text-to-SQL users (who may have a good idea of what kind of information resides on their databases but are not proficient in SQL (Hendrix et al., 1978)) to provide feedback to correct inaccurate SQL parses (Section 5.2.2); (3) I construct SPLASH—**S**emantic **P**arsing with **La**nguage **As**sistance from **H**umans—a new dataset of natural language questions that a recent neural text-to-SQL parser failed to generate correct interpretations for together with corresponding human-provided natural language feedback describing how the interpretation should be corrected (Sections 5.2 and 5.3) ; and (4) I establish several baseline models for the correction task and show that the task is challenging for state-of-the-art semantic parsing models (Section 5.4).

5.1 Task Definition

I formally define the task of SQL parse correction with natural language feedback. Given a question q, a database schema s, a mispredicted parse p', a natural language feedback f on p', the task is to generate a corrected parse p (Figure 5.2). Following Yu et al. (2018c), s is defined as the set of tables, columns in each table and the primary and foreign keys of each table.

Models are trained with tuples q, s, p', f and gold parse p. At evaluation time, a model takes as input tuples in the form q, s, p', f and hypothesizes a corrected parse $\hat{p}$. I compare $\hat{p}$ and the gold parse p in terms of their exact set match (Yu et al., 2018c) (Section 2.3.4) and report the average matching accuracy across the testing examples as the model's correction accuracy.



Question:

Find all the locations whose names contain the word "film"

Predicted Parse:

```
SELECT Address FROM LOCATIONS WHERE
Location_Name LIKE '%film%'
```

Feedback:

Address is wrong. I want the name of the locations

Gold Parse:

```
SELECT Location_Name FROMLOCATIONS
WHERE Location_Name LIKE '%film%'
```

Schema:

Location_ID	Location_Name	Address	Other_Details

Figure 5.2: An example from my SQL parse correction task (DB Name: `cre_Theme_park` and Table Name: `Locations`). Given a question, initial predicted parse and natural language feedback on the predicted parse, the task is to predict a corrected parse that matches the gold parse.

5.2 SPLASH Construction

In this section, I describe my approach for collecting training data for the SQL parse correction task. I first generate pairs of natural language questions and the corresponding erroneous SQL parses (Section 5.2.1). Then, I employ crowd workers (with no SQL knowledge) to provide feedback, in natural language, to correct the erroneous SQL (Section 5.2.3). To enable such workers to provide feedback, I show them an explanation of the generated SQL in natural language (Section 5.2.2). Finally, to improve the diversity of the natural language feedback, I ask a different set of annotators to paraphrase each feedback. I describe the process in detail in the remainder of this section.

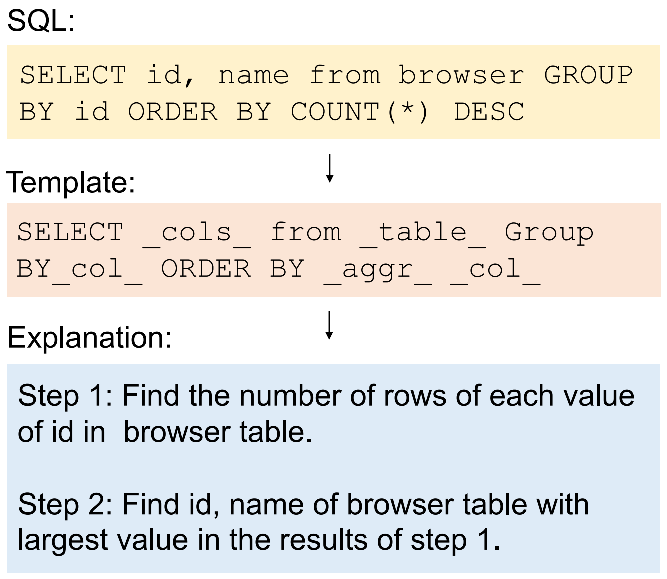
5.2.1 Generating Questions and Incorrect SQL Pairs

I use the SPIDER dataset (Section 2.3.4) as my source of questions. SPIDER has several advantages over other datasets. Compared to ATIS and GeoQuery, SPIDER is much larger in scale (200 databases vs. one database, and thousands vs. hundreds of SQL parses). Compared to WikiSQL, SPIDER questions require inducing parses of complex structures (requiring multiple tables, joining, nesting, etc.). SPIDER also adopts a cross-domain evaluation setup in which databases used at testing time are never seen at training time.

To generate erroneous SQL interpretations of questions in SPIDER, I opt for using the output of a text-to-SQL parser to ensure that the dataset reflects the actual distribution of errors that contemporary parsers make. This is a more realistic setup than artificially infusing errors in the gold SQL. I use the SEQ2STRUCT parser (Section 2.3.4) [1] to generate erroneous SQL interpretations. When I started the dataset construction at the beginning of June 2019, I was able to achieve a parsing accuracy of 41.30% with SEQ2STRUCT on SPIDER's development set which was the state-of-the-art accuracy at the time. Unlike current state-of-the-art models, SEQ2STRUCT does not use pre-trained language models. It was further developed into thee RAT-SQL model (Section 2.3.4) which achieved a new state-of-the-art accuracy as of April 2020. Also, I note that I make no explicit assumptions on the model used for this step and hence other models could be used as well.

I train SEQ2STRUCT on 80% of SPIDER's training set and apply it to the remaining 20%, keeping only cases where the generated parses do not match the gold parse (according to the exact set match). Following the by-database splitting scheme of SPIDER, I repeat

[1]`https://github.com/rshin/seq2struct`



```
SELECT id, name from browser GROUP
BY id ORDER BY COUNT(*) DESC
```

```
SELECT _cols_ from _table_ Group
BY_col_ ORDER BY _aggr_ _col_
```

Step 1: Find the number of rows of each value of id in browser table.

Step 2: Find id, name of browser table with largest value in the results of step 1.

Figure 5.3: An example of a SQL query, the corresponding template and the generated explanation.

the 80-20 training and evaluation process for three times with different examples in the evaluation set at each run. This results in 3,183 pairs of questions and an erroneous SQL interpretation. To further increase the size of the dataset, I also ignore the top prediction in the decoder beam (I use a beam of size 20) and use the following predictions. I only include cases where the difference in probability between the top and second to last SQLs is below a threshold of 0.2. The intuition here is that those are predictions that the model was about to make and hence represent errors that the model could have made. That adds 1,192 pairs to the dataset.

5.2.2 Explaining SQL

In one of the earliest work on natural language interfaces to databases, Hendrix et al. (1978) note that many business executives, government official and other decision makers have a good idea of what kind of information residing on their databases. Yet to obtain an answer to a particular question, they cannot use the database themselves and instead need to employ the help of someone who can. As such, in order to support an interactive text-to-SQL system, we need to be able to explain the incorrect generated SQL in a way that humans who are not proficient in SQL can understand.

I take a template-based approach to explain SQL queries in natural language. I define a template as follows: Given a SQL query, I replace literals, table and columns names and aggregation and comparison operations with generic placeholders. I also assume that all joins are inner joins (true for all SPIDER queries) whose join conditions are based on primary and foreign key equivalence (true for more than 96% of Spider queries). A query that consists of two subqueries combined with an intersection, union or except operations is split into two templates that are processed independently and I add an intersection/union/except part to the explanation at the end. I apply the same process to the limit operation—generate an explanation of the query without limit, then add a limit-related step at the end.

I select the most frequent 57 templates used in SPIDER training set which cover 85% of SPIDER queries. For each SQL template, I wrote down a corresponding explanation template in the form of steps (e.g., join step, aggregation step, selection step) that I populate for each query. Figure 5.4 shows several examples of how different SQL components can be explained in natural language. I also implemented a set of rules for compressing the steps based on SQL semantics. For instance, an ordering step following by a "`LIMIT 1`" is replaced with "find largest/smallest" where "largest" or "smallest" is decided according to the ordering direction. Figure 5.3 shows an example of a SQL queries, its corresponding template and generated explanations. I note that understanding the explanations still re-

SQL Component	**Explanation**
`INTERSECT`	show the rows that are in both the results of step 1 and step 2
`UNION`	show the rows that are in any of the results of step 1 and step 2
`EXCEPT`	show the rows that are in the results of step 1 but not in the results of step 2
`LIMIT n`	only keep the first n rows of the results in step 1
`JOIN`	for each row in Table 1, find corresponding rows in Table 2
`SELECT`	find Column of Table
`AGGREGATION`	find each value of {Column1} in Table along with the {OPERATION} of {Column2} of the corresponding rows to each value
`ORDERING`	order {Direction} by {Column}
`CONDITION`	whose {Column} {Operation} {Value}
`DISTINCT`	without repetition

Figure 5.4: Examples of how different SQL components can be explained in natural language

quires some awareness of the schema and a general idea about the operations of executing SQL queries which restricts the kinds of user who can interact with the system. In Chapter 7, I discuss future work on improving and evaluating explanations of text-to-SQL and question answering systems in general.

5.2.3 Crowdsourcing Feedback

I use a Microsoft-internal crowd-sourcing platform similar to Amazon Mechanical Turk to recruit annotators. Annotators are only selected based on their performance on other crowd-sourcing tasks and command of English. Before working on the task, annotators go through a brief set of guidelines explaining the task (Figure 5.5). I collect the dataset in batches of around 500 examples each. After each batch is completed, I manually review a sample of the examples submitted by each annotator and exclude those who do not provide accurate inputs from the annotators pool and redo all their annotations.

Annotators are shown (Figure 5.6) the original question, the explanation of the generated SQL and asked to: (1) decide whether the generated SQL satisfies the information need in the question and (2) if not, then provide feedback in natural language. The first step is necessary since it helps identifying false negative parses (e.g., another correct parse that does not match the gold parse provided in SPIDER). I also use the annotations of that step to assess the extent to which my interface enables target users to interact with the underlying system. As per my assumption that target users are familiar with the kind of information that is in the database (Hendrix et al., 1978), I show the annotators an overview of the tables in the database corresponding to the question that includes the table and column names together with examples (first 2 rows) of the content. I limit the maximum feedback length to 15 tokens to encourage annotators to provide a correcting

Correcting Steps for Answering Questions.

1. We have some information stored in tables; each row is a record that consists of one or more columns. Using the given tables, we can answer questions by doing simple systematic processing steps over the tables. Notice that the answer to the question is always the result of the last step. Also, notice that the steps might not be in perfect English as they were generated automatically. Each step, generates a table of some form.

2. For each question, we have generated steps to answer it, but it turned out that something is wrong with the steps. Your task is write down in English a short (one sentence most of the time) description of the mistakes and how it can be correct. Note that we are not looking for rewritings of steps, but rather we want to get short natural English commands (15 words at most) that describes the correction to be made to get the correct answer.

3. Use proper and fluent English. Don't use math symbols.

4. Don't rewrite the steps after correcting them. Just describe the change that needs to be made.

5. We show only two example values from each table to help you understand the contents of each table. Tables typically contain several rows. Never use the shown values to write your input.

6. There could be more than one wrong piece in the steps. make sure to mention all of them.

7. If the steps are correct, just check the "All steps are correct" box.

8. Some of the mistakes are due to additional steps or parts of steps. Your feedback can suggest removing those parts.

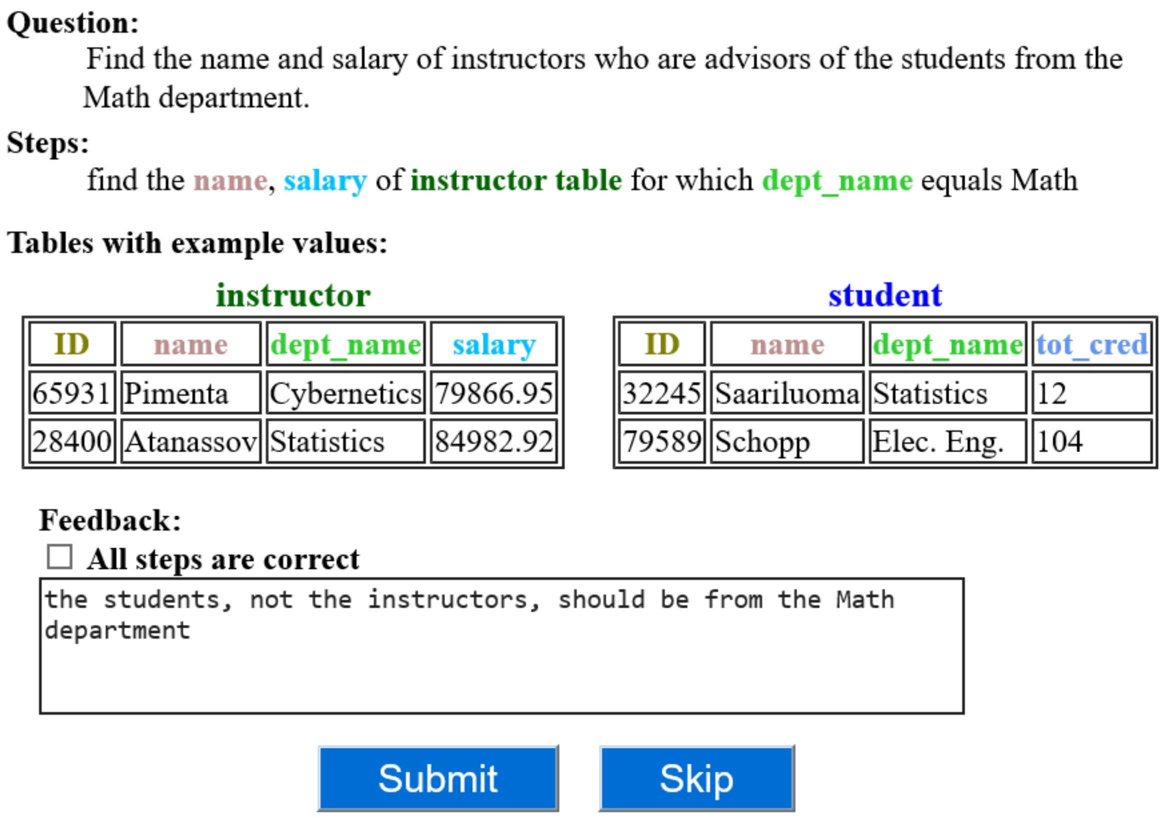
9. Do not just copy parts of the questions. Be precise in your input.

10. If in doubt about how to correct a mistake, just mention what is wrong.

11. You do not have to mention which steps contain mistakes. If in doubt, do not refer to a any steps.

12. The generated steps might not sound like the smartest way for answering the question. But it is the most systematic way that works for all kinds of questions and all kinds of tables. Please, do not try to propose smarter steps.

Figure 5.5: Crowd-sourcing instructions



Question:

Find the name and salary of instructors who are advisors of the students from the Math department.

Steps:

find the name, salary of **instructor table** for which **dept_name** equals Math

Tables with example values:

instructor

ID	name	dept_name	salary
65931	Pimenta	Cybernetics	79866.95
28400	Atanassov	Statistics	84982.92

student

ID	name	dept_name	tot_cred
32245	Saariluoma	Statistics	12
79589	Schopp	Elec. Eng.	104

Feedback:

☐ **All steps are correct**

```
the students, not the instructors, should be from the Math
department
```

Figure 5.6: An example of the data collection interface. The Predicted SQL is: `'SELECT name, salary FROM instructor WHERE dept_name LIKE "%math%"'`. Note that neither the gold nor the predicted SQL are shown to the annotator.

feedback based on the initial parse that focuses on the edit to be made rather than describing how the question should be answered. A total of 10 annotators participated in this task. They were compensated based on an hourly rate (as opposed to per annotation) to encourage them to optimize for qualiaty and not quantity. They took an average of 6 minutes per annotation.

To improve the diversity of the feedback I collect, I ask a separate set of annotators to generate a paraphrase of each feedback utterance. I follow the same annotators quality

Number of	**Train**	**Dev**	**Test**
Number of Examples	7,481	871	962
Number of Databases	111	9	20
Number of Unique Questions	2,775	290	506
Number of Unique Wrong Parses	2,840	383	325
Number of Unique Gold Parses	1,781	305	194
Number of Unique Feedback Utterances	7,350	860	948
Average Feedback Length (Number of tokens)	13.9	13.8	13.1

Table 5.1: SPLASH summary

control measures as in the feedback collection task. An example instance from the dataset is shown in Figure 5.2.

5.2.4 SPLASH Summary

Overall, I asked the annotators to annotate 5409 example (427 of which had the correct SQL parse and the remaining had an incorrect SQL parse). Examples with correct parse are included to test whether the annotators are able to identify correct SQL parses given their explanation and the question. Annotators are able to identify the correct parses as correct 96.4% of the time. For the examples whose predicted SQL did not match the gold SQL, annotators still marked 279 of them as correct. Upon manual examinations, I found that annotators were indeed correct in doing so 95.5% of the time. Even though the predicted and gold SQLs did not match exactly, they were equivalent (e.g., '`price BETWEEN 10 and 20`' vs. '`price` $\geq$ `10 and price` $\leq$ `20`)'.

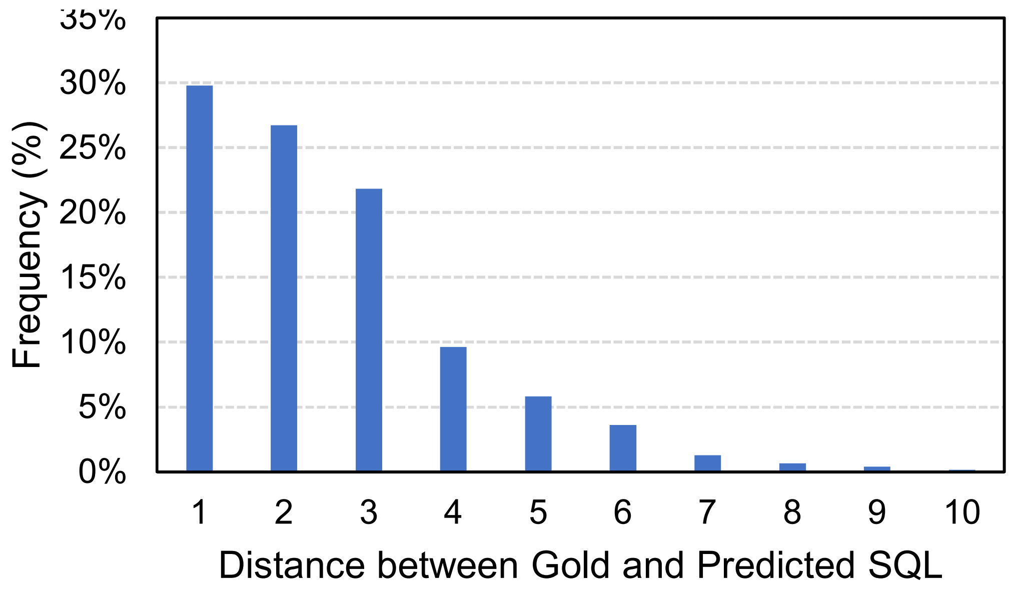


Figure 5.7: A histogram of the distance between the gold and the predicted SQL.

After paraphrasing, I ended up with 9,314 question-feedback pairs, 8352 of which correspond to questions in the SPIDER training split and 962 from the SPIDER development split. I use all the data from the SPIDER development split as my test data. I hold out 10% of the remaining data (split by database) to use as my development set and use the rest as the training set. Table 5.1 provides a summary of the final dataset.

5.3 SPLASH Analysis

I conduct a more thorough analysis of SPLASH in this section. I study the characteristics of the mistakes made by the parser as well as characteristics of the natural language feedback.

5.3.1 Error Characteristics

I start by characterizing the nature of errors usually made by the models in parsing the original utterance to SQL. To understand the relation between the gold and the predicted SQL, I measure the edit distance between them for all cases for which the model made a mistake in the SQL prediction. I measure the edit distance by the number of edit segments (delete, insert, replace) between both parses. I find the minimal sequence of token-level edits using the levenshtein distance algorithm. Then, I combine edits of the same type (delete, insert, replace) applied to consecutive positions in the predicted parse in one segment. Figure 5.7 shows a frequency histogram of different values of edit distance. I observe that most inaccurate predictions lie within a short distance from the correct SQL (78%+ within a distance of 3 or less).

In addition to the number of edits, I also characterize the types of edits needed to convert the predicted SQL to the gold one. My edit distance calculations support three operations replace, insert and delete. Those correspond to 58%, 31% and 11% of the edit operations respectively. Most of the edits are rather simple and require replacing, inserting or deleting a single token (68.1% of the edits). The vast majority of those correspond to editing a schema item (table or column name): 89.2%, a SQL keyword (e.g., order direction, aggregation, count, distinct, etc.): 7.4%, an operator (greater than, less than, etc.): 2.2% or a number (e.g. for a limit operator): 1.2%.

The edits between the predicted and generated SQL span multiple SQL keywords. The distribution of different SQL keywords appearing in edits and their distribution across edit types (replace, insert or delete) is shown in Figure 5.8. Note that a single edit could involve multiple keywords (e.g., multiple joins, a join and a where clause, etc.). Interestingly,

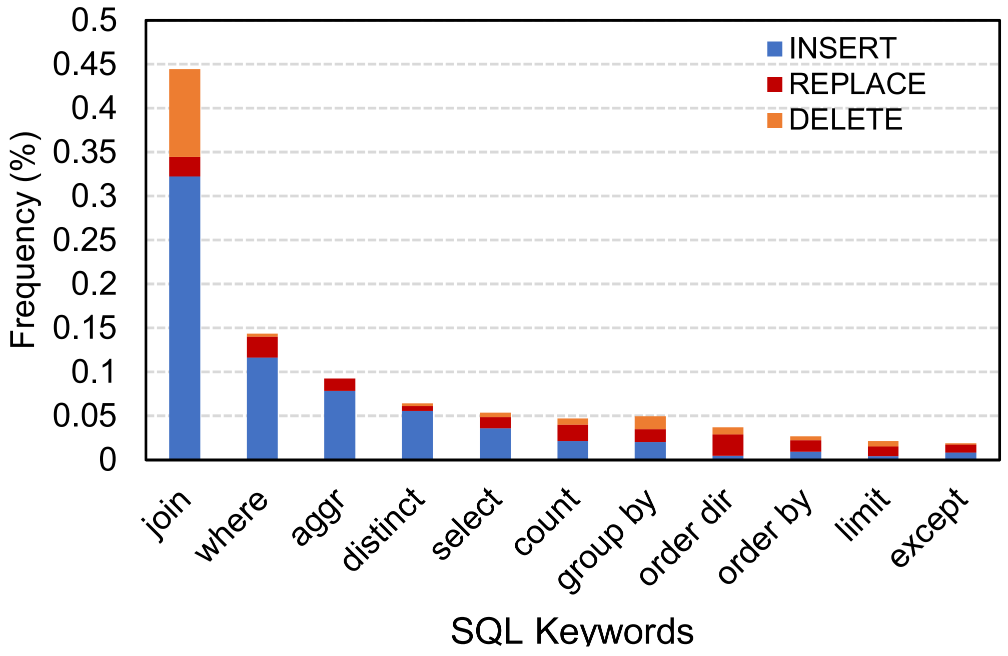


Figure 5.8: A histogram of different SQL keywords appearing in edits (between the gold and predicted SQL) and their distribution across edit types (replace, insert or delete).

many of the edits involve a `JOIN` highlighting that handling utterances that require a join is harder and more error prone. Following `JOIN`, most edits involve `WHERE` clauses (making the query more or less specific), aggregation operators, counting and selecting unique values. The error analysis demonstrates that many of the errors made by the model are in fact not significant and hence it is reasonable to seek human feedback to correct them.

5.3.2 Feedback Characteristics

To better understand the different types of feedback our annotators provided, I sample 200 examples from the dataset, and annotate them with the type of the feedback. I start by assigning the feedback to one of three categories: (1) **Complete:** the feedback fully describes how the predicted SQL can be corrected , (2) **Partial:** the feedback describes a

Complete Feedback: [81.5%]	
Question:	Show the types of schools that have two schools.
Pred. SQL:	`SELECT TYPE FROM school GROUP BY TYPE HAVING count(*) >= 2`
Feedback:	You should not use greater than.
Partial Feedback: [13.5%]	
Question:	What are the names of all races held between 2009 and 2011?
Pred. SQL:	`SELECT country FROM circuits WHERE lat BETWEEN 2009 AND 2011`
Feedback:	You should use races table.
Paraphrase Feedback: [5.0%]	
Question:	What zip codes have a station with a max temperature greater than or equal to 80 and when did it reach that temperature?
Pred. SQL:	`SELECT zip_code FROM weather WHERE min_temperature_f > 80 OR min_sea_level_pressure_inches > 80`
Feedback:	Find date , zip code whose max temperature f greater than or equals 80.

Table 5.2: Examples (question, predicted SQL and feedback) of complete, partial and paraphrase feedback

way to correct the predicted SQL but only partially, and (3) **Paraphrase:** the feedback is a paraphrase of the original question. The sample had 81.5% for Complete, 13.5% for Partial and 5.0% for Paraphrase feedback. Examples of each type of feedback are shown in Table 5.2. Upon further inspection of the partial and paraphrase feedback, I observe that they mostly happen when the distance between the predicted and gold SQL is high (major parsing errors). As such, annotators opt for providing partial feedback (that would

Feedback Type	%	Example
Information		
- Missing	13%	I also need the number of different services
- Wrong	36%	Return capacity in place of height
- Unnecessary	4%	No need to return email address
Conditions		
- Missing	10%	ensure they are FDA approved
- Wrong	19%	need to filter on open year not register year
- Unnecessary	7%	return results for all majors
Aggregation	6%	I wanted the smallest ones not the largest
Order/Unique	5%	only return unique values

Table 5.3: Examples of feedback annotators provided for different types

at least correct some of the mistakes) or decide to rewrite the question in a different way.

I also annotate and present the types of feedback, in terms of changes the feedback is suggesting, in Table 5.3. Note that the same feedback may suggest multiple changes at the same time. The table shows that the feedback covers a broad range of types, which matches my initial analysis of error types. I find that a majority of feedback is referencing the retrieved information. In many such cases, the correct information has not been retrieved because the corresponding table was not used in the query. This typically corresponds to a missing inner one-to-one join operation and agrees with my earlier analysis on edit distance between the gold and predicted SQL. The second most popular category is incorrect conditions or filters followed by aggregation and ordering errors. I split the first

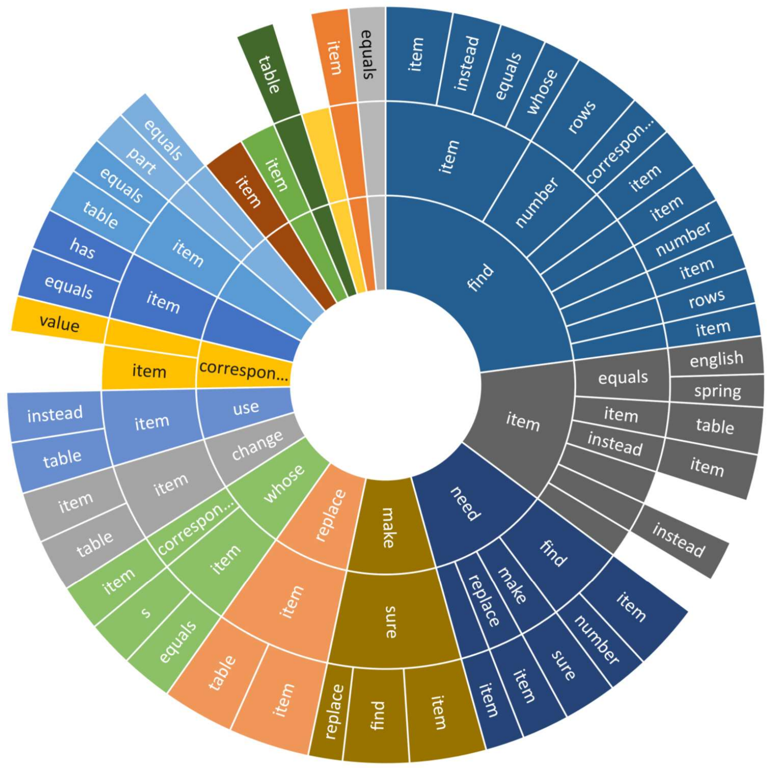


Figure 5.9: Patterns of feedback covered in our dataset. Patterns are extracted heuristically using predicates and arguments extracted from the feedback sentence. The figure shows the top 60 frequent patterns. Feedback utterances in SPLASH are expressed in *diverse* ways.

two categories by whether the information/conditions are missing, need to be replaced or need to be removed. I observe that most of the time the information or condition needs to be replaced. This is followed by missing information that needs to be inserted and then unnecessary ones that need to be removed.

I heuristically identify feedback patterns for each collected feedback. To identify the feedback pattern, I first locate the central predicate in the feedback sentence using a semantic role labeler (He et al., 2015). Since some feedback sentences can be broken into multiple sentence fragments, a single feedback may contain more than one central predicate. For each predicate, I identify its main arguments. I represent each argument with its first non stopword token. To reduce the vocabulary, I heuristically identify column mentions and replace them with the token 'item'.

I visualize the distribution of feedback patterns for the top 60 most frequent patterns in Figure 5.9, and label the ones shared among multiple patterns. As is shown, SPLASH covers a diverse variety of feedback patterns centered around key operations to edit the predicted SQL that reference operations, column names and values.

5.4 Experiments

I present and evaluate a set of baseline models for the correction task (Section 5.1) in which I use SPLASH for training and testing (unless otherwise stated). My main evaluation measure is the correction accuracy—the percentage of the testing set examples that are corrected—in which I follow Yu et al. (2018c) and compare the corrected parse to the gold parse using exact set match (Section 2.3.4). I also report on SPIDER development set (which I use to construct our testing set) the end-to-end accuracy of the two turn interaction scenario: first SEQ2STRUCT attempts to parse the input question. If it produced a wrong parse, the question together with that parse and the corresponding feedback are attempted using the correction model. An example is considered "correct" if any of the two attempts produces the correct parse. I note that SEQ2STRUCT produces correct parses for 427/1034 of SPIDER Dev. 511 of the remaining examples are supported by my SQL explanation patterns (Section 5.2.2). I estimate the end-to-end accuracy as $(427 + 511 * X/100)/1034$, where X is the correction accuracy.

5.4.1 Baselines

Methods that ignore the feedback: One approach for parse correction is re-ranking the decoder beam (top-n predictions (Section 2.1.5)) (Yin and Neubig, 2019). Here, I simply discard the top-1 candidate and sample uniformly and with probabilities proportional to the parser score of each candidate. I also report the accuracy of deterministically choosing the second candidate.

Handcrafted re-ranking with feedback: By definition, the feedback f describes how to edit the initial parse p' to reach the correct parse. I represent the "diff" between p' and each candidate parse in the beam p_i as the set of schema items that appear only in one of them. For example, the diff between `SELECT first_name, last_name FROM students` and `SELECT first_name FROM teachers` is {last_name, students, teachers}. I assign a score to p_i equals to the number of matched schema items in the diff with f. A schema item (e.g., "first_name") is considered to be mentioned in f is all the individual tokens ("first" and "name") are tokens in f.

SEQ2STRUCT+Feedback: The SEQ2STRUCT model I use to generate erroneous parses for data collection (Section 5.2.1) reaches an accuracy of 41.3% on SPIDER's development set when trained on the full SPIDER training set for 40,000 steps. After that initial training phase, I adapt the model to incorporate the feedback by appending the feedback to the question for each training example in SPLASH, and I continue training the model to predict the gold parse for another 40,000 steps. I note that SEQ2STRUCT+Feedback does not use the mispredicted parses.

EDITSQL+Feedback: EDITSQL (Section 2.3.4) is among the top-performing models for conversational text-to-SQL (Zhang et al., 2019). It generates a parse for an utterance at a conversation turn i by copying from (i.e., editing) the parse generated at turn $i-1$ while condition on all previous utterances as well as the schema. I adapt EditSQL for the correction task by providing the question and the feedback as the utterances at turn one and two respectively, and I force it to use the mispredicted parse as the prediction of turn one (rather than predicting it). I train the model on the combination of the training sets of SPLASH and SPIDER (which is viewed as single turn conversations). I exclude turn one predictions from the training loss when processing SPLASH examples otherwise, the model would be optimized to produce the mispredicted parses. I use the default hyper-parameters provided by the authors[2] together with the development set of SPLASH for early stopping.

To provide an estimate of **human performance**, I report the percentage of feedback instances labeled as *Complete* as described in Section 5.3.2. I also report the **re-ranking upper bound** (the percentage of test examples whose gold parses exist in SEQ2STRUCT beam).

5.4.2 Main Results

The results in Table 5.4 suggest that: (1) the feedback I collect is indeed useful for correcting erroneous parses; (2) incorporating the mispredicted parse helps the correction process (even a simple handcrafted baseline that uses the mispredicted parases outperforms a strong trained neural model); and (3) the conversational EditSQL model equipped with BERT achieves the best performance, yet it still struggles with the task, leaving a large gap for improvement.

[2]https://github.com/ryanzhumich/editsql

	Exact Match Accuracy (%)	
Baseline	**Correction**	**End-to-End**
Without Feedback		
⇒ Seq2Struct	N/A	41.30
⇒ Re-ranking: Uniform	2.39	42.48
⇒ Re-ranking: Parser score	11.26	46.86
⇒ Re-ranking: Second Best	11.85	47.15
With Feedback		
⇒ Re-ranking: Handcrafted	16.63	49.51
⇒ Seq2Struct+Feedback	13.72	48.08
⇒ EditSQL+Feedback	**25.16**	**53.73**
Re-ranking Upper Bound	36.38	59.27
Estimated Human Accuracy	81.50	81.57

Table 5.4: Correction and end-to-end accuracies of baseline models.

5.4.3 Analysis

Does EditSQL+Feedback use the feedback? To confirm that EditSQL+Feedback does not learn to ignore the feedback, I create a test set of random feedback by shuffling the feedback of splash test examples. The accuracy on the randomized test set drops to 5.6%.

Is splash just another conversational text-to-SQL dataset? I evaluate the trained EditSQL models on SParC and CoSQL, on splash test set, and I get accuracies of

3.4% and 3.2%, respectively. In comparision, EditSQL achieves 47.9% accuracy on SParC and 40.8% on CoSQL. That confirms that splash targets different modeling aspects as we discuss in Section 2.3.3.

Is splash only useful for correcting Seq2Struct errors? EditSQL is also shown to achieve strong results on spider (57.6% on the development set) when used as a single-turn mode. I collect feedback for a sample of 179 mispredicted parses produces by EditSQL. I started with 200, but 21 of them turned out to have alternative correct parses (false negatives). Using the EditSQL+Feedback model trained on splash I get a correction accuracy of 14.6% for EditSQL errors which confirms that splash is also useful for correcting mistakes of other parsers, but with some degradation in the correction accuracy. I study that aspect in more depths in the next chapter.

5.5 Conclusion

In this chapter, I introduced the task of SQL parse correction. The task aims to correct inaccurate SQL parses using natural language feedback provided by the user. I created a large-scale dataset, splash, of text-to-SQL with natural language feedback submitted by crowdworkers to correct inaccurate SQL interpretations of natural language utterances. splash features a diverse set of natural language feedback utterances aiming to correct a diverse set of text-to-SQL parsing mistakes. Further analysis showed that a large percentage of errors made by a recent neural text-to-SQL system are not far off from the correct interpretation. It also showed that collecting natural language feedback from humans is feasible and useful for addressing many of these errors. I experimented with multiple simple baselines that try to re-rank parses from the beam of the original model or re-generate a new SQL given the question, the feedback and the incorrect predicted

SQL. The evaluation showed that such methods have potential for improving the overall accuracy and hence, can create a better user experience. It also shows that the model accuracy still leaves significant room for improvement, suggesting the importance of better models for collecting and leveraging feedback from users to further improve performance.

In the next chapter, I present NL-EDIT, an improved model for the correction task. NL-EDIT combines two key ideas: (1) interpreting the feedback in the context of the other elements of the interaction (database schema, question, and explanation of the initial query), and (2) explicitly generate edit operations to correct the initial query instead of re-generating the full query from scratch. I design a simple SQL editing language whose basic units are add/delete operations applied to different SQL clauses. With that, I boost the correction accuracy to 41%.

Chapter 6: Edit-Based Model for Interactive Semantic Parsing

In the previous chapter, I introduced a framework for interactive text-to-SQL in which induced SQL queries are fully explained in natural language to users, who in turn, can correct such parses through natural language feedback. I described the construction of the SPLASH dataset and used it to evaluate baselines for the task of semantic parse correction with natural language feedback that I introduced.

In this chapter, I present a detailed analysis of the feedback and the differences between the initial (incorrect) and the correct parse. I argue that a correction model should be able to interpret the feedback in the context of other elements of the interaction (the original question, the schema, and the explanation of the initial parse). I observe from SPLASH that most feedback utterances tend to describe a few edits that the user desires to apply to the initial parse. As such, I pose the correction task as a semantic parsing problem that aims to convert natural language feedback to a sequence of edits that can be deterministically applied to the initial parse to correct it. Figure 1.2 demonstrates the operation of NL-EDIT. I use that edit-based modeling framework to show that we

The work in this chapter is published as "Ahmed Elgohary, Christopher Meek, Matthew Richardson, Adam Fourney, Gonzalo Ramos, and Ahmed Hassan Awadallah. 2021. NL-EDIT: Correcting semantic parse errors through natural language interaction. In Conference of the North American Chapter of the Association for Computational Linguistics" (Elgohary et al., 2021).

can effectively generate synthetic data to pre-train the correction model leading to clear performance gains.

Specifically, I present a scheme for representing SQL query Edits that benefits both the modeling and the analysis of the correction task (Section 6.1), (2) I present NL-EDIT (Section 6.2), an edit-based model for interactive text-to-SQL with natural language feedback. I show (Section 6.4) that NL-EDIT outperforms baselines presented in Chapter 5 by more than 16% in correction accuracy (Section 5.4), (3) I demonstrate that we can generate synthetic data through the edit-based framing and that the model can effectively use this data to improve its accuracy (Section 6.3) and (4) I present a detailed analysis of the model performance including studying the effect of different components, and generalization to errors of state-of-the-art parsers (Section 6.5).

6.1 SQL Edits

I define a scheme for representing the edits required to transform one SQL query to another. I use that scheme both in my model and analysis. My goal is to balance the granularity of the edits—too fine-grained edits result in complex structures that are challenging for models to learn, and too coarse-grained edits result in less compact structures that are harder for models to generate.

I view a SQL query as a set of clauses (e.g, `SELECT`, `FROM`, `WHERE`), each clause has a sequence of arguments (Figure 6.1). I mirror the SQL clauses `SELECT`, `FROM`, `WHERE`, `GROUP-BY`, `ORDER-BY`, `HAVING`, and `LIMIT`. For subqueries, I define a clause `SUBS` whose arguments are recursively defined as sets of clauses. Subqueries can be linked to the main query in two ways: either through an `IEU` clause (mirrors SQL `INTERSECT/EXCEPT/UNION`) whose first argument is one of the keywords `INTERSECT`, `EXCEPT`, `UNION` and its second

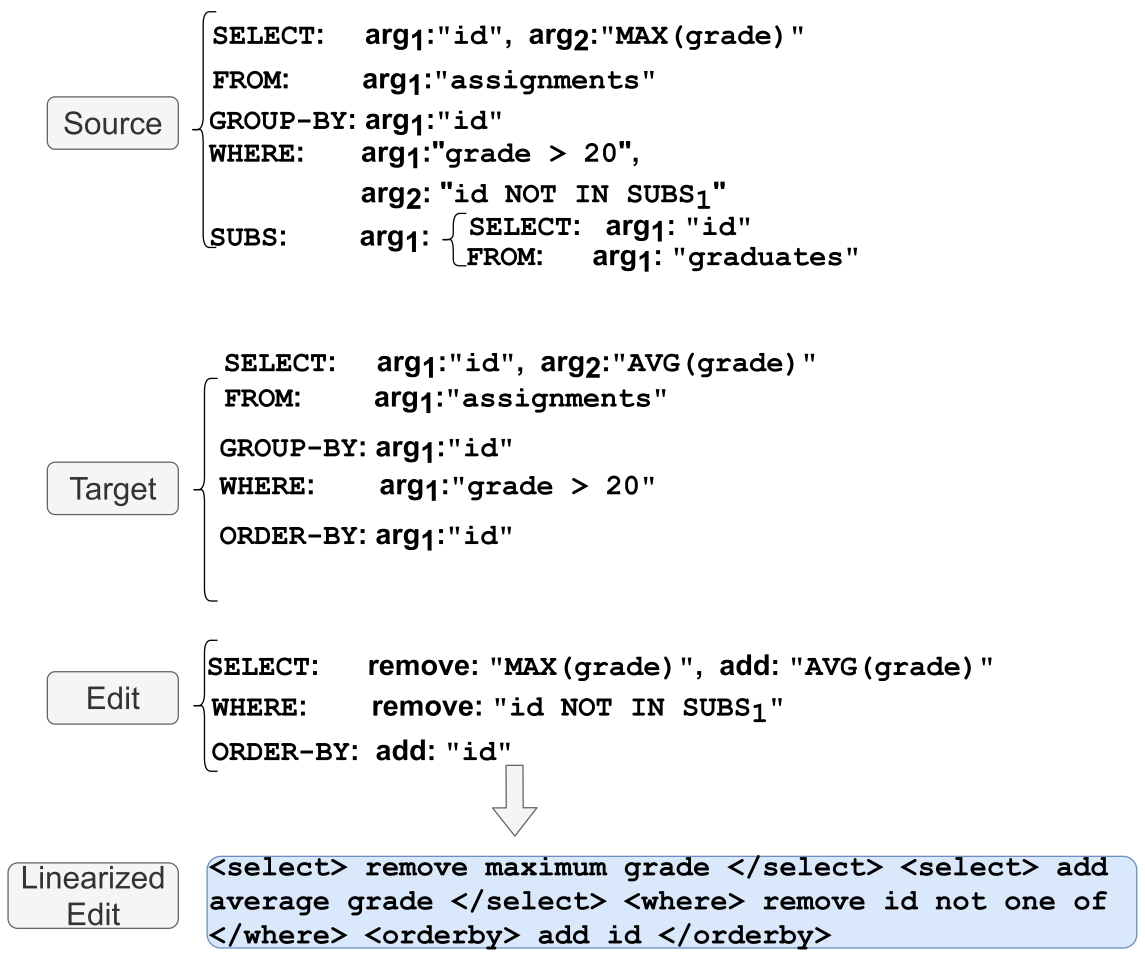

Source
SELECT: arg1:"id", arg2:"MAX(grade)"
FROM: arg1:"assignments"
GROUP-BY: arg1:"id"
WHERE: arg1:"grade > 20", arg2: "id NOT IN SUBS1"
SUBS: arg1: SELECT: arg1: "id" FROM: arg1: "graduates"
Target
SELECT: arg1:"id", arg2:"AVG(grade)"
FROM: arg1:"assignments"
GROUP-BY: arg1:"id"
WHERE: arg1:"grade > 20"
ORDER-BY: arg1:"id"
Edit
SELECT: remove: "MAX(grade)", add: "AVG(grade)"
WHERE: remove: "id NOT IN SUBS1"
ORDER-BY: add: "id"
Linearized Edit
<select> remove maximum grade </select> <select> add average grade </select> <where> remove id not one of </where> <orderby> add id </orderby>

Figure 6.1: Edit for transforming the **source** query "`SELECT id, MAX(grade) FROM assignments WHERE grade > 20 AND id NOT IN (SELECT id from graduates) GROUP BY id`" to the **target** "`SELECT id, AVG(grade) FROM assignment WHERE grade > 20 GROUP BY id ORDER BY id`". The source and target are represented as sets of clauses (left and middle). The set of **edits** and its **linearized** form (Section 6.2) are shown on the right. Removing the condition "`id NOT IN SUBS`$_1$" makes the subquery unreferenced, hence pruned from the edit.

argument is a pointer to a subquery in `SUBS`. The second is through nested queries where the arguments of some of the clauses (e.g., `WHERE`) can point at subqueries in `SUBS` (e.g., "`id NOT IN` $\texttt{SUBS}_1$").

With such view of two queries $\mathcal{P}_{source}$ and $\mathcal{P}_{target}$, I define their edit $\mathcal{D}_{source \to target}$ as the set of clause-level edits $\{\mathcal{D}^{c}_{source \to target}\}$ for all types of clauses c that appear in $\mathcal{P}_{source}$ or $\mathcal{P}_{target}$ (Figure 6.1). To compare two clauses of type c, I simply exact-match their arguments: unmatched arguments in the source (e.g., `MAX(grade)` in `SELECT`) are added as to-remove arguments to the corresponding edit clause, and unmatched arguments in the target (e.g., "`id`" in the `ORDER-BY`) are added as to-add arguments.

My current implementation follows SPIDER's assumption (Section 2.3.4) that the number of subqueries is at most one which implies that computing edits for different clauses can be done independently even for the clauses that reference a subquery (e.g., `WHERE` in Figure 6.1). The edit of the `SUBS` clause is recursively computed as the edit between two queries (any of them can be empty); the subquery of source and the subquery of target, i.e., $\mathcal{D}^{\texttt{SUBS}}_{source \to target} = \mathcal{D}_{source:\texttt{SUBS}_1 \to target:\texttt{SUBS}_1}$. I keep track of the edits to the arguments that reference the subquery. After all edit clauses are computed, I prune the edits of the `SUBS` clause if the subquery will no longer be referenced ($\texttt{SUBS}_1$ in Figure 6.1). I follow the SPIDER evaluation and discard the values in `WHERE/HAVING` clauses.

Figure 6.1 shows an example edit that transforms a source query into a target query. Starting from the `SELECT` clause, the argument `id` is matching in both source and target. The second argument in source is `MAX(grade)` while the second argument in target is `AVG(grade)`. To transformer the `SELECT` arguments of source to those in target, the edit *removes* `MAX(grade)` and *adds* `AVG(grade)`. The arguments of the `FROM` and `GROUP-BY` clauses are matching in both the source and target. So, the edit does not include any add/remove operations for both clauses. Source has an extra condition (`id NOT IN` $\texttt{SUBS}_1$) in its `WHERE` clause so, the edit of the `WHERE` clauses just *removes* that condition. Finally, source is missing an `ORDER-BY` clause with the argument `id` which is then *added* by the edit.

I refer to the number of add/remove operations in an edit as the **Edit Size**, and I denote it as $|\mathcal{D}_{source \rightarrow target}|$. For example, the edit in Figure 6.1 is of size four.

6.2 Model

I follow the task description in Section 5.1: the inputs to the model are the elements of the interaction—question, schema, an initial parse $\tilde{\mathcal{P}}$, and feedback. The model predicts a corrected $\bar{\mathcal{P}}$. The gold parse $\hat{\mathcal{P}}$ is available for training. The model (NL-EDIT) is based on integrating two key ideas in an encoder-decoder architecture (Section 2.1.5). I start with a discussion of the intuitions behind the two ideas followed by the model details.

6.2.1 Intuitions

Interpreting feedback in context: The feedback is expected to link to all the other elements of the interaction (Figure 1.2). The feedback is provided in the context of the *explanation* of the initial parse, as a proxy to the parse itself. As such, the feedback tends to use the same terminology as the explanation. For example, the SQL explanations (Section 5.2.2) express "`GROUP BY`" in simple language "for each vote_id, find ...". As a result, human-provided feedback never uses "`GROUP BY`". I also notice that in several SPLASH examples, the feedback refers to particular steps in the explanation as in the examples in Figure 1.2. Unlike the models I discussed in Chapter 5, I replace the initial parse with its natural language explanation. Additionally, the feedback usually refers to columns/tables in the schema, and could often be ambiguous when examined in isolation.

Such ambiguities can be usually resolved by relying on the context provided by the question. For example, "find last name" in Figure 1.2 is interpreted as "find last name besides first name" rather than "replace first name with last name" because the question asks for the "full name". My first key idea is based on grounding the elements of the interaction by combining self-learned relations by transformer models (Section 2.1.4) and hard-coded relations that we define according to the possible ways different elements can link to each other.

Feedback describes a set of edits: The difference between the erroneous parse and the correct one can mostly be described as a few edits that need to be applied to the initial parse to correct its errors (Section 6.5). Also, the feedback often only describes the edits to be made. As such, we can pose the task of correction with NL feedback as a semantic parsing task where we convert a natural language deception of the edits to a canonical form that can be applied deterministically to the initial parse to generate the corrected one. I train NL-EDIT to generate SQL Edits (Section 6.1) rather than SQL queries.

6.2.2 Encoder

My encoder (Figure 6.2) starts with passing the concatenation of the feedback, explanation, question, and schema through BERT (Section 2.1.6). Following Wang et al. (2020); Suhr et al. (2018); Scholak et al. (2020), I tokenize the column (table) names and concatenate them in one sequence (Schema) starting with the tokens of the tables followed by the tokens of the columns. Then, I average the BERT embeddings of the tokens corresponding to each column (table) to obtain one representation for the column (table). Each explanation is provided as a list of steps (Section 5.2.2). I concatenate the tokens of all steps in one sequence and separate the tokens of each step with a special token [STEP-SEP]. I also add special tokens [SUB-QUERY-START] and [SUB-QUERY-END] before and after the steps corresponding to each subquery (if exists).

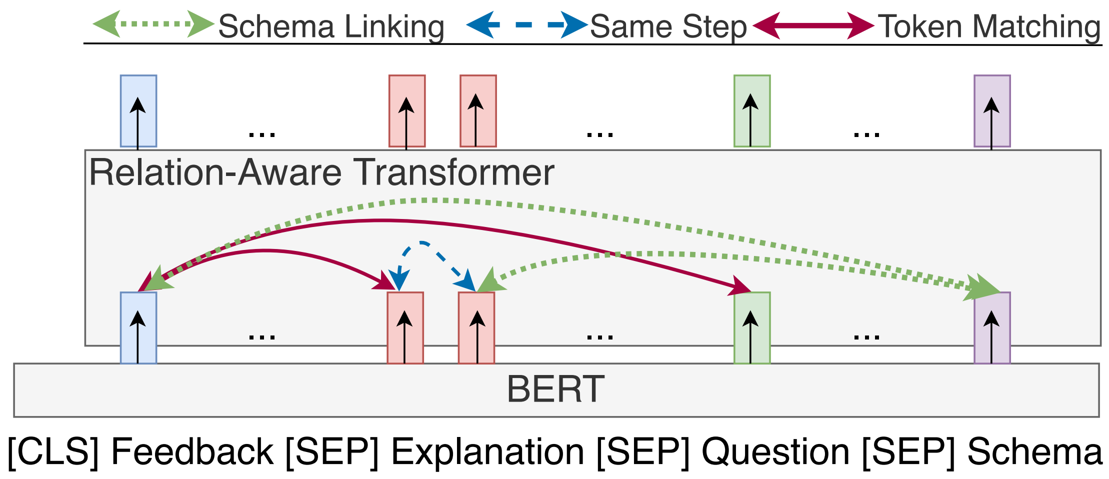

Schema Linking
Same Step
Token Matching
Relation-Aware Transformer
BERT
[CLS] Feedback [SEP] Explanation [SEP] Question [SEP] Schema

Figure 6.2: The Encoder of NL-EDIT grounds the feedback into the explanation, the question, and the schema by (1) passing the concatenation of their tokens through BERT, then (2) combining self-learned and hard-coded relations in a relation-aware transformer. Three types of relations (Interaction Relations) link the individual tokens of the inputs. Question-Schema and Schema-Schema relations are not shown.

The RAT-SQL model (Section 2.3.4) shows the benefit of injecting preexisting relations within the schema (column exists in a table, primary-foreign key), and between the question and schema items (column and table names) by: (1) name linking: link a question token to a column/table if the token and the item name match and (2) value linking: link a question token to a column if the token appears as a value under that column. To incorporate such relations in their model, they use the relation-aware transformer model (Section 2.1.3).

In addition to those relations, I define a new set of relations that aim at contextualizing the feedback with respect to the other elements of the interaction in our setup: (1) ***[Feedback-Schema]*** I link the feedback to the schema the same way the question is linked to the schema via both name and value linking, (2) ***[Explanation-Schema]*** Columns and tables are mentioned with their exact names in the explanation. I link the explanation to the schema only through exact name matching, (3) ***[Feedback-Question]*** I use partial (at the lemma level) and exact matching to link tokens in the feedback and the question, (4) ***[Feedback-Explanation]*** I link tokens in the feedback to tokens in the explanation through partial and exact token matching. Since the feedback often refers to particular steps, I link the feedback tokens to explanation tokens that occur in steps that are referred to in the feedback with a separate relation type that indicates step reference in the feedback, and (5) ***[Explanation-Explanation]*** I link explanation tokens that occur within the same step. I use the same relation-aware transformer formulation as in the RAT-SQL model and add the relation-aware layers on top of BERT to integrate all relations into the model (Figure 6.2).

6.2.3 Decoder

Using a standard teacher-forced cross-entropy loss (Section 2.1.5), I train NL-EDIT to generate *linearized* SQL Edits (Figure 6.1). At training time, I compute the reference SQL Edit $\mathcal{D}_{\tilde{\mathcal{P}}\to\hat{\mathcal{P}}}$ of the initial parse $\tilde{\mathcal{P}}$ and the gold parse $\hat{\mathcal{P}}$ (Section 6.1). Then I linearize $\mathcal{D}_{\tilde{\mathcal{P}}\to\hat{\mathcal{P}}}$ by listing the clause edits in a fixed order (`FROM`, `WHERE`, `GROUP-BY`, ... etc.). The argument of each clause—representing one add or remove operation—is formatted as

```
<CLAUSE> ADD/REMOVE ARG </CLAUSE>.
```

I express SQL operators in `ARG` with natural language explanation as in Section 5.2.2.

For example, the argument ``AVG(grade)'' is expressed as "average grade". At inference time, I generate a corrected parse $\bar{\mathcal{P}}$ by applying the produced edit to the initial parse $\tilde{\mathcal{P}}$.

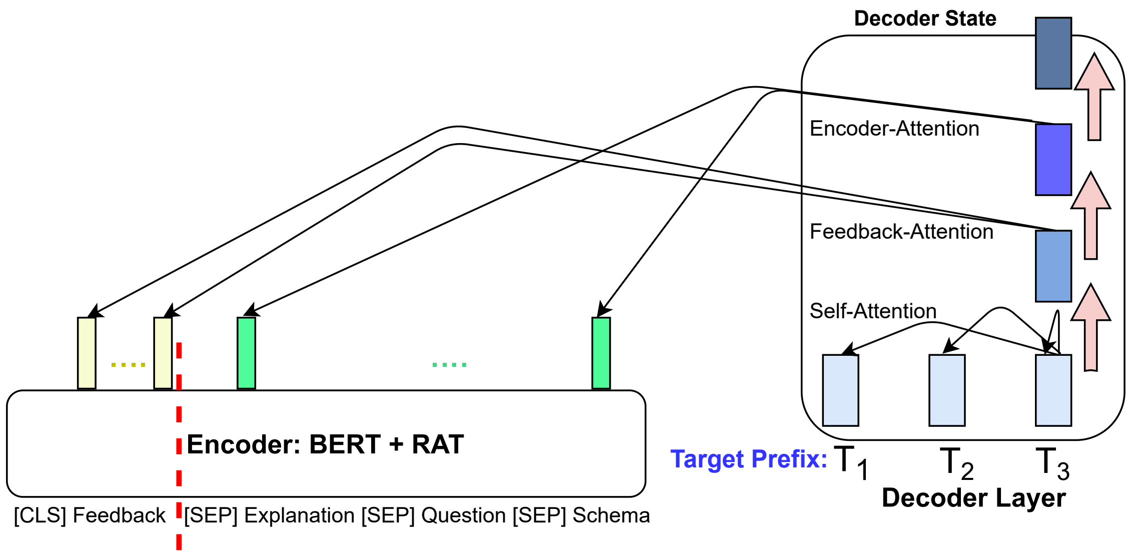


Figure 6.3: NL-EDIT with a transformer decoder that splits the attention to the encoder: First it attends to the feedback then uses the updated decoder state to attend to the other parts of the input. Figure 2.1 shows the standard encoder-decoder attention.

I use a standard transformer decoder (Figure 2.1) that either generates tokens from the output vocab or copies columns and tables from the encoder output. Since all editing operations should be directed by the feedback, I tried splitting the attention to the encoder into two phases (Figure 6.3): First, I attend to the feedback only and update the decoder state accordingly. Then, I use the updated decoder state to attend to the other inputs. With that, I only observed a marginal improvement of 0.5% in the accuracy. I conduct all my experiments in the rest of this chapter with standard decoder-encoder attention and leave the investigation of other attention patterns for future work.

6.3 Synthetic Feedback

Algorithm 1 Training Data Synthesis

```
1: for seed in SPIDER training set do
2:     for $p$ in CLONE(seed, $N$) do
3:         feedback = []
4:         for $i = 1$ : RAND-NUM-EDITS() do
5:             $e \leftarrow$ RAND-FEASIBLE-EDIT($p$)
6:             $p$.APPLY-EDIT($e$)
7:             feedback.ADD($e$.FEEDBACK())
8:         end for
9:         output: seed.DB, seed.Question, $p$,
10:                feedback, seed.Gold-SQL
11:    end for
12: end for
```

In this section, I describe the process for automatically synthesizing additional examples for training the correction model. Recall that each example consists of a question about a given schema paired with a gold parse, an initial erroneous parse, and feedback. Starting with a seed of questions and their corresponding gold parses from SPIDER's training set (8,099 pairs) (I ensure there is no overlap between examples in the seed and the dev set of SPLASH), my synthesis process applies a sequence of SQL editing operations to the gold parse to reach an altered parse that I use as the initial parse (Algorithm 1).

By manually inspecting the edits (Section 6.1) I induce for the initial and gold

Replace-Select-Column:

- replace {NEW-COL} with {OLD-COL}
- you should find {OLD-COL} instead

Add-Where-Condition:

- delete {COL} {OPERATOR} {VALUE}

Remove-Limit:

- only top {LIMIT-VALUE} rows are needed

Table 6.1: Example SQL Editors with corresponding feedback templates. The synthesized feedback is reversing the edit applied to a correct SQL as our synthesis process starts with the gold SQL and reaches an initial SQL after applying the edit.

parses in SPLASH training set, I define 26 SQL editors and pair each editor with their most frequent corresponding feedback template(s) (Examples in Table 6.1). I also associate each editor with a set of constraints that determines whether it can be applied to a given SQL query e.g., the "Remove-Limit" editor can only be applied to a query that has a limit clause.

Algorithm 1 summarizes the synthesis process. I start by creating N (controls the size of the dataset) clones of each seed example. The analysis of SPLASH (Section 5.3) shows that multiple mistakes might be present in the initial SQL, hence I allow my synthesis process to introduce up to four edits (randomly decided in line:4) to each clone p. For each editing step, I sample a feasible edit for the current parse (line:5) with manually set probabilities for each edit to balance the number of times each editor is applied in the final dataset. A feasible edit meets all constrains defined with the editor and a set sequence-dependent constraints to avoid applying editors that cancel each other out. Applying an edit (line:6) involves sampling columns/tables from the current parse and/or the schema, sampling operators and values for altering conditions, and populating the corresponding feedback template. I combine the feedback of all the applied editors into one string and use it as the feedback of the synthesized example.

6.4 Experiments

	Correction Acc. (%)	**Edit $\downarrow$ (%)**	**Edit $\uparrow$ (%)**	**Progress (%)**
Rule-based Re-ranking	16.63	38.35	32.81	-15.67
EditSQL+Feedback	25.16	47.44	23.51	7.71
NL-EDIT	**41.17**	**72.41**	**16.93**	**36.99**
Oracle Re-ranking	36.38	34.69	1.04	31.22

Table 6.2: Comparing NL-EDIT to **baselines** in (Section 5.4): Rule-based Re-ranking and EditSQL+Feedback and to the beam re-ranking upper-bound. **Edit** $\downarrow$ (**Edit** $\uparrow$) is the percentage of examples on which the number of edits/errors strictly decreased (increased). **Progress** is the average relative reduction in the number of edits (Section 6.4). NL-EDIT outperforms both baselines on all measures (with 95% statistical significance on the Correction Accuracy) and outperforms the oracle re-ranking on all measures except Edit $\uparrow$. The upper-bound on the correction accuracy is estimated as 81.5% (Section 5.4).

6.4.1 Setup

I conduct my experiments using SPLASH (Section 5.2.3) whose train, dev, and test sets are of sizes 7481, 871, and 962, respectively. Using the feedback synthesis process (Section 6.3), I generate 50,000 additional synthetic training examples. In preliminary experiments, I found that training the model on the synthetic dataset first then continuing on SPLASH outperforms mixing the synthetic and real examples and training on both of them simultaneously. I train the model on the synthetic examples for 20,000 steps and continue training on the real examples until reaching 100,000 steps in total. I choose the best checkpoint based on the development set accuracy. I varied the number of training steps on the synthetic examples and 20,000 steps achieved the highest accuracy on the dev set.

I use BERT-BASE-UNCASED (Section 2.1.6) in all my experiments. I set the number of layers in the relational-aware transformer to eight (Following Wang et al. (2020)) and the number of decoder layers to two. I train with batches of size 24. I use the Adam optimizer (Kingma and Ba, 2015) for training. I freeze BERT parameters during the first 5,000 warm-up steps and update the rest of the parameters with a linearly increasing learning rate from zero to 5×10^{-4}. Then, I linearly decrease the learning rates from 5×10^{-5} for BERT and 5×10^{-4} for the other parameters to zero. The learning rate schedule is only dependent on the step number regardless of whether we are training on the synthetic data or SPLASH. I tried resetting the learning rates back to their maximum values after switching to SPLASH, but did not observe any improvement in accuracy. I use beam search (Section 2.1.5) with a beam of size 20 and take the top-ranked beam that results in a valid SQL after applying the inferred edit.

6.4.2 Evaluation

As in Chapter 5, I use the *correction accuracy* as the main evaluation measure: each example in SPLASH test set contains an initial parse $\tilde{\mathcal{P}}$ and a gold parse $\hat{\mathcal{P}}$. With a predicted (corrected) parse by a correction model $\bar{\mathcal{P}}$, the correction accuracy is computed using the exact-set-match (Section 2.3.4) between $\bar{\mathcal{P}}$ and $\hat{\mathcal{P}}$ averaged over all test examples. While useful, correction accuracy also has limitations. It expects models to be able to fully correct an erroneous parse with only one utterance of feedback as such, it is defined in terms of the exact match between the corrected and the gold parse. I find (Table 6.2) that in several cases, models were still able to make *progress* by reducing the number of errors as measured by the edit size (Section 6.1) after correction. As such, I define another set of measures to estimate partial progress. I report (Edit $\downarrow$ and Edit $\uparrow$ in Table 6.2) the percentage of examples on which the size of the edit set strictly decreased/increased. To combine Edit $\downarrow$ and Edit $\uparrow$ in one measure and account for the relative reduction (increase) in the number of edits, I define

$$\text{Progress}(\mathcal{S}) = \frac{1}{|S|} \sum_{\tilde{\mathcal{P}},\bar{\mathcal{P}},\hat{\mathcal{P}} \in \mathcal{S}} \frac{|\mathcal{D}_{\tilde{\mathcal{P}}\to\hat{\mathcal{P}}}| - |\mathcal{D}_{\bar{\mathcal{P}}\to\hat{\mathcal{P}}}|}{|\mathcal{D}_{\tilde{\mathcal{P}}\to\hat{\mathcal{P}}}|}.$$

Given a test set $\mathcal{S}$, the Progress of a correction model is computed as the average relative edit reduction between the initial parse $\tilde{\mathcal{P}}$ and the gold parse $\hat{\mathcal{P}}$ by predicting a correction $\bar{\mathcal{P}}$ of $\tilde{\mathcal{P}}$. A perfect model that can fully correct all errors in the initial parse would achieve a 100% progress. A model can have a negative progress (e.g., Rule-based re-ranking in Table 6.2) when it frequently predicts corrections with more errors than those in the initial parse. Unlike correction accuracy, Progress is more aligned with user experience in an interactive environment (Su et al., 2018) as it assigns partial credit for fixing a subset of the errors and also, it penalizes models that predict an even more erroneous parse after receiving feedback.

6.4.3 Results

I compare (Table 6.2) NL-EDIT to the two top-performing baselines and also to the beam re-ranking upper-bound (Chapter 5). NL-EDIT significantly (based on McNemar tests (McNemar, 1947; Dror et al., 2018) with significance level of 0.05) increases the correction accuracy over the top baseline (EditSQL+Feedback) by more than 16% and it also outperforms oracle re-ranking by around 5%. I also note that in 72.4% of the test examples, NL-EDIT was able to strictly reduce the number of errors in the initial parse (Edit $\downarrow$) which potentially indicates a more positive user experience than the other models. NL-EDIT achieves 37% Progress which indicates faster convergence to the fully corrected parse than all the other models.

6.5 Analysis

6.5.1 Ablations

Following the same experimental setup in Section 6.4, I compare NL-EDIT to other variants with one ablated component at a time (Table 6.3). I ablate the **feedback**, the **explanation**, and the **question** from the encoder input. I also ablate the **interaction relations** (Section 6.2.2) that I incorporate in the relation-aware transformer module. I only ablate the new relations I introduce to model the interaction (shown in Figure 6.2), but I keep the Question-Schema and Schema-Schema relations introduced in (Wang et al., 2020). For each such variant, I train for 20,000 steps on the synthetic dataset then continue training on SPLASH until step 100,000. I also train an ablated variant that does not use

Model	**Correction Accuracy (%)**
NL-EDIT	41.17
– Feedback	19.81
– Explanation	26.80
– Question	38.27
– Interaction Relations	35.35
– Synthetic Feedback	35.01

Table 6.3: Correction accuracy on SPLASH Test of NL-EDIT versus variants with one ablated component each. Each component of NL-EDIT helps boosting the accuracy of the full model with 95% statistical significance.

the **synthetic feedback** where I train for 100,000 steps only on SPLASH. For all variants, I choose the checkpoint with the largest correction accuracy on the dev set and report the accuracy on the SPLASH test set.

The results in Table 6.3 confirm the effectiveness of each component in my model. Ablating any of the components significantly reduces the accuracy of the full model according to McNemar tests with significance level of 0.05. The model is able to correct 19.8% of the examples without the feedback. I noticed that the ablated-feedback model almost reaches that accuracy only after training on the synthetic data with very minor improvement ($< 1\%$) after training on SPLASH. Only using the question and the explanation, the model is able to learn about a set of systematic errors that parsers make and how they can be corrected (Gupta et al., 2017; Yin and Neubig, 2019).

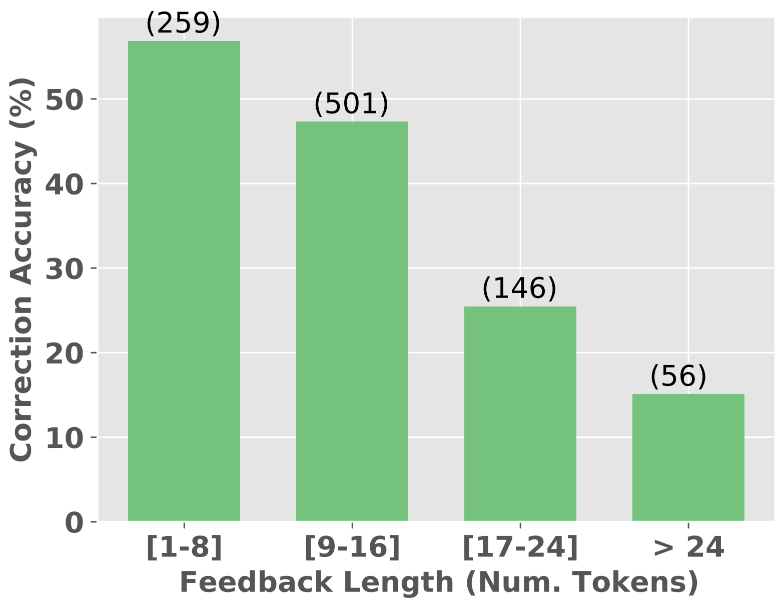


Figure 6.4: Breakdown of the correction accuracy on SPLASH test set by the feedback length. The number of examples in each group is shown on top of the bars.

6.5.2 Error Analysis

In Figures 6.4, 6.5 and 6.6, I breakdown the correction accuracy by the feedback and explanation lengths (in number of tokens) and by the reference edit size (number of required edit operations to fully correct the initial parse). The accuracy drops significantly when the reference edit size exceeds two (Figure 6.6), while it declines more gradually as the feedback and explanation increase in length. I manually (Examples in Table 6.4) inspected the examples with longer feedback than 24, and found that 8% of them the feedback is long because it describes how to rewrite the whole query rather than being limited to only the edits to be made. In the remaining 92%, the initial query had several errors (edit size of 5.5 on average) with the corresponding feedback enumerating all of them.

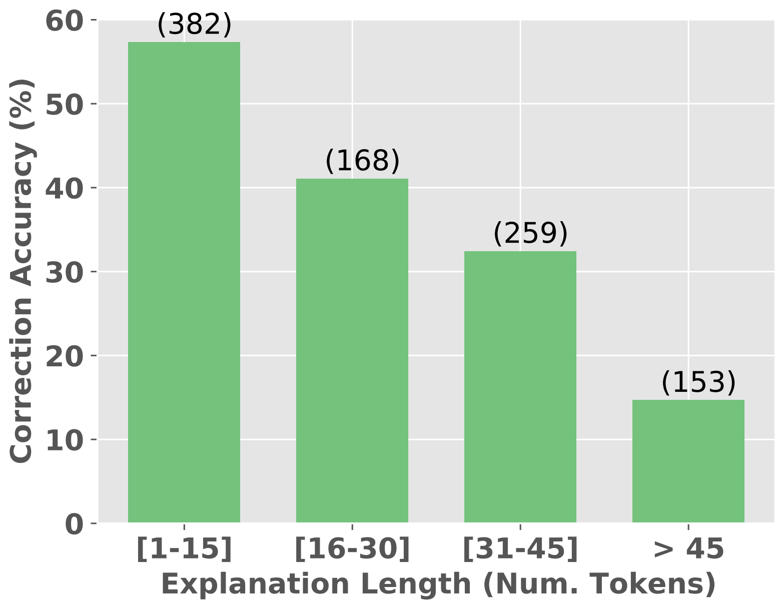


Figure 6.5: Breakdown of the correction accuracy on SPLASH test set by the explanation length. The number of examples in each group is shown on top of the bars.

Figure 6.7 shows how the number of errors (measured in edit size) changes after correction. The figure shows that even for examples with a large number of errors (four and five), the model is still able to reduce the number of errors in most cases. I manually inspected the examples with only one error that the model failed to correct. I found 15% of them have either wrong or non-editing feedback and in 29% the model produced the correct edit but with additional irrelevant ones. The dominant source of error in the remaining examples is because of failures with linking the feedback to the schema (Examples in Table 6.5).

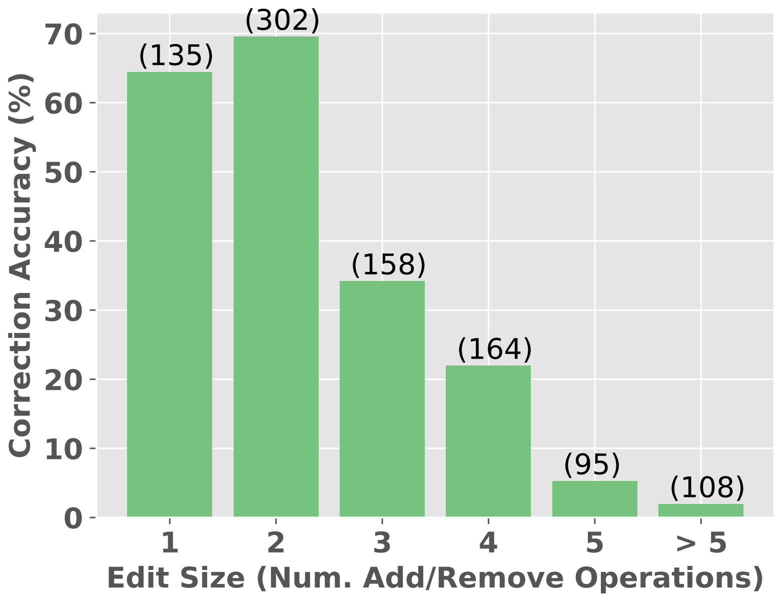


Figure 6.6: Breakdown of the correction accuracy on SPLASH test set by the size of the reference edit (number of add or remove operations). The number of examples in each group is shown on top of the bars.

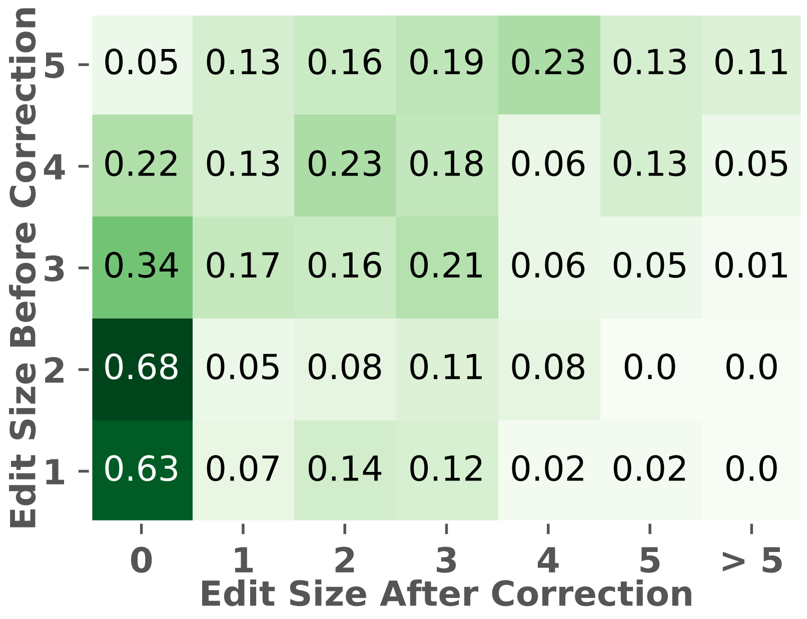


Figure 6.7: Transitions in edit size after correction. For each edit size of the initial parse (rows), we show the distribution of the edit size after correction.

Long Feedback Not Describing an Edit:
"you should determine the major record format from the orchestra table and make sure it is arranged in ascending order of number of rows that appear for each major record format."
Long Feedback Describing an Edit:
"replace course id (both) with degree program id, first courses with student enrolment, course description with degree summary name, second courses with degree programs."

Table 6.4: Example long feedback that NL-EDIT struggles with. Top: The feedback describes a rewriting of the query rather than how to edit it. Bottom: The initial query has several errors and the feedback enumerates how to edit all of them.

6.5.3 Cross-Parser Generalization

So far, I have been using SPLASH for both training and testing. The erroneous parses (and corresponding feedback) in SPLASH are based on the SEQ2STRUCT parser (Section 2.3.4). Recent progress in model architectures (Wang et al., 2020) and pre-training (Yin et al., 2020; Yu et al., 2021a) has led to parsers that already outperform SEQ2STRUCT by more than 30% in parsing accuracy.[1] Here, I ask whether NL-EDIT that I train on SPLASH (and synthetic feedback) can generalize to parsing errors made by more recent parsers without additional parser-specific training data.

I follow the same crowdsourcing process used to construct SPLASH (Section 5.2.3) to

[1]https://yale-lily.github.io/spider

Adding extra edits:

Ques.: Which city and country is the Alton airport at?

Initial: `SELECT City, Country FROM airports WHERE AirportName = 'Alton' AND Country = 'USA'`

Feedback: remove "and country equals USA" phrase.

Predicted: `<where> remove Country equals </where> <where> remove AirportName equals </where>`

Gold: `<where> remove Country equals </where>`

Failing to link feedback and schema:

Ques.: What are the full names of all left handed players, in order of birth date?

Initial: `SELECT first_name, last_name FROM players ORDER BY birth_date Asc`

Feedback: make sure that player are left handed.

Predicted: `<where> add birth_date equals </where>`

Gold: `<where> add hand equals </where>`

Table 6.5: Example failure cases of NL-EDIT.

	SEQ2STRUCT	EDITSQL	TaBERT	RAT-SQL
Correction Test Sets Summary				
Number of Examples	962	330	267	208
Average Feedback Length	13.1	13.5	12.9	12.2
Average Explanation Length	26.4	28.3	32.2.9	34.0
Semantic Parsing Accuracy (%)				
Error Correction	41.1	28.0	22.7	21.3
No Interaction	41.3	57.6	65.2	69.7
End-to-End	61.6	66.6	71.1	74.0
Δ w/ Interaction	+**20.3**	+**8.9**	+**5.9**	+**4.3**

Table 6.6: Evaluating the zero-shot generalization of NL-EDIT to different parsers (EDITSQL, TaBERT, and RAT-SQL) after training on SPLASH that is constructed based on the SEQ2STRUCT parser. Top: Summary of the dataset constructed based on each parser. Feedback and explanation length is the number of tokens. Bottom: The **Error Correction** accuracy on each test set and the end-to-end accuracy of each parser on the full SPIDER dev set with and without interaction. **Δ w/ Interaction** is the gain in end-to-end accuracy with the interaction added. All such gains are statistically significant with a significance level of 0.05.

collect three new test sets based on three recent text-to-SQL parsers: EDITSQL, TaBERT, and RAT-SQL (see Section 2.3.4 for an overview for each parser). Following the same process for constructing SPLASH test set, I run each parser on SPIDER dev set and only

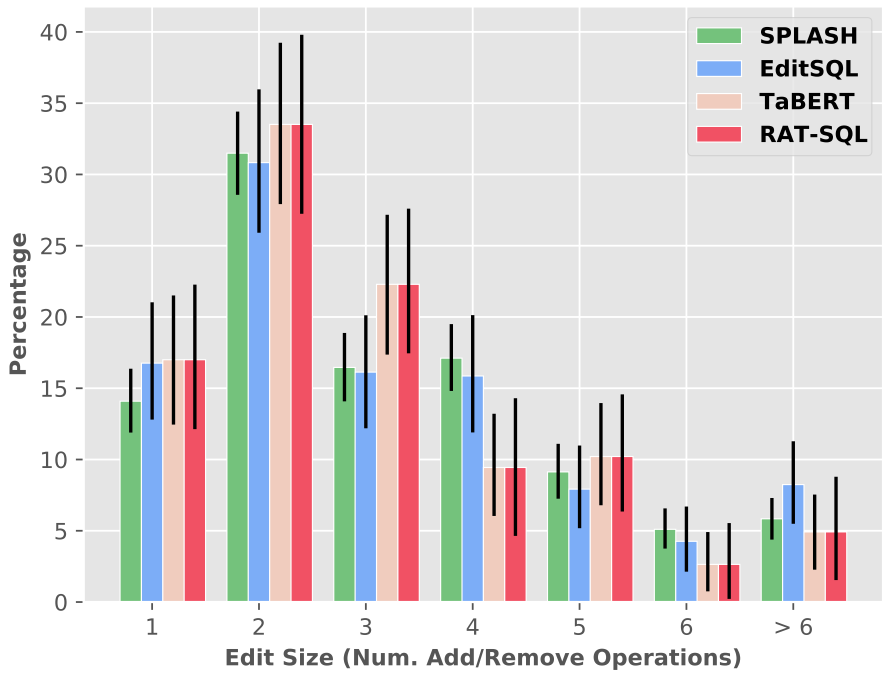


Figure 6.8: Distribution of Edit Size per example in SPLASH compared to the generalization test sets constructed based on EDITSQL, TaBERT, and RAT-SQL. The black bars are 95% confidence intervals estimated with the percentile method using 10,000 bootstrap samples (Efron and Tibshirani, 1994).

collect feedback for the examples with incorrect parses that can be explained using their SQL explanation framework. Table 6.6 (Top) summarizes the three new test sets and compares them to SPLASH test set. I note that the four datasets are based on the same set of questions and databases (SPIDER dev).

Table 6.6 (Bottom) compares the parsing accuracy (measure by exact query match (Section 2.3.4) of each parser when used by itself **(No Interaction)** to integrating it with NL-EDIT. I report both the accuracy on the examples provided to NL-EDIT (**Error Correction**) and the **End-to-End** accuracy on the full SPIDER dev set. NL-EDIT significantly (according to McNemar tests with significance level of 0.05) boosts the accuracy of all parsers, but with a notable drop in the gains as the accuracy of the parser improves. To explain that, in Figure 6.8 I compare the distribution of reference edit size across the four datasets. The figure does not show any significant differences in the distributions that would lead to such a drop in accuracy gain. Likewise, the distributions of the feedback lengths are very similar (the mean is shown in Table 6.6). As parsers improve in accuracy, they tend to make most of their errors on complex SQL queries. Although the number of errors with each query does not significantly change (Figure 6.8), I hypothesize that localizing the errors in a complex initial parse, with a long explanation (Table 6.6), is the main generalization bottleneck that future work needs to address.

6.6 Conclusions

In this chapter, I introduced the NL-EDIT model, a data augmentation method, and analysis tools for correcting semantic parse errors in text-to-SQL through natural language feedback. Compared to the strong baselines I introduced in Chapter 5, NL-EDIT improves the correction accuracy by 16% and boosts the end-to-end parsing accuracy by up to 20% with only one turn of feedback. Still there is a significant gap between NL-EDIT (the state-of-the-art model for the correction task) and the estimated upper-bound. Also, my analysis showed that there is also a generalization bottleneck that limits NL-EDIT's ability to correct parsing mistakes made by more recent parsers.

Overall, this chapter together with the previous chapter clearly demonstrate the potential of improving semantic parsers (hence, question answering) by enabling users to interact with machines through natural language. In the next chapter, I summarize the outcome and results of this thesis that I also use to draw a research agenda for further advancing collaborative human–computer NLP.

Chapter 7: Conclusions and Future Work

Artificial intelligence (AI) is currently at the stage of wide deployment in real life applications. Crucially, users need to trust the AI systems they are using and stay in-control. Moreover, AI models are not perfect and occasionally struggle or make mistakes. When we team up users with AI, they will verify the correctness of AI outputs and, if they are in-control, interact with the AI to provide corrective feedback when needed. Collaborative human–computer AI is the realistic setup for a wide deployment of AI systems.

The goal of this thesis is to create systems, methods and tools that help advance collaborative human–computer NLP. In particular, I focus on collaborations done through natural language interactions. With natural language, users can express their feedback with much flexibility and richness without being limited to the interaction capabilities that systems support (e.g., options in a graphical interface). With only one turn of natural language feedback, I show that the accuracy of question answering systems can be boosted by up to 20%. In this chapter, I summarize the contributions this thesis makes (Section 7.1), discuss limitations of the work I presented in the previous chapters (Section 7.2), then I outline future research directions based on the findings of the thesis (Sections 7.3 and 7.4).

7.1 Summary of Contributions

Natural language can convey rich information and when it is naturally occurring it can be a valuable source for training various machine learning models. In Chapter 3, I showed that the naturally occurring bilingual text can be very useful for learning general purpose vector representations of sentences. I compare three forms of natural language supervision for that goal: 1) bilingual sentences, 2) bilingual phrases that are automatically extracted from bilingual sentences, and 3) monolingual paraphrases which are also extracted from bilingual text, but require an order of magnitude larger corpus to learn reliable paraphrases. On English semantic textual similarity benchmarks that span several domains and genres, I show that learning with bilingual paraphrases is superior to the two others forms. That finding 1) demonstrates the effectiveness of learning from natural language supervision, and 2) offers a much resource-efficient approach for learning vector representations of sentences.

To enable human–computer interaction through natural language, machines should be equipped with methods for understanding user utterances in conversational contexts. In Chapter 4, I study conversational question answering. I present baseline methods for incorporating previously asked questions and their answers to improve the accuracy of answering interrelated questions in the open-domain QA setup (Section 2.2). Then, I introduce question-in-context rewriting to reduce context-dependent questions to stand-alone questions that can be answered by existing QA models. I create CANARD: a benchmark dataset that can be used for evaluating and analyzing rewriting models. Recent work has shown that the rewriting task and CANARD are useful for various setups of conversational QA (Vakulenko et al., 2020, 2021; Chen et al., 2021).

In Chapter 5, I create an interactive semantic parsing system that answers questions about data stored in relational databases by translating questions into SQL queries. In that system, users inspect the SQL queries produced by the system and interact with the system by provide feedback in free-form natural language that describes any mistakes with the queries and possibly how they can be corrected. To enable laypeople to inspect a SQL parse, I develop an approach for explaining SQL queries in simple natural language steps. Then, I construct SPLASH: a benchmark dataset that consists of erroneous SQL queries paired with human-authored natural language feedback. Using SPLASH, I introduce the task of semantic parse correction with natural language feedback, and present results of a set of baseline models. SPLASH will help the development and evaluation of improved models for the correction task. Also, the SQL explanation module will inspire the development of explainable semantic parsing systems e.g., (Narechania et al., 2021; Xu et al., 2021, inter alia).

In Chapter 6, I present NL-EDIT: an improved model for the correction task. I use the inputs to the correction task (question, initial parse, schema and feedback) to construct a graph that I encode with the relation-aware transformer model (Section 2.1.4). I also create a SQL editing language that I parse the feedback into rather than parsing the feedback in full SQL queries. Then, I introduce an approach for automatically generating synthetic feedback examples that I train NL-EDIT on (in addition to SPLASH) to further improve its correction accuracy. Then, I switch to creating analysis tools for better understanding the behaviour of correction models in general. I present fine-grained evaluation measures that take into account partial corrections. I also extend SPLASH and create new tests sets that evaluate the generalization of correction models to parsing errors made by more recent parsers.

7.2 Limitations

In Chapter 3, I evaluate bilingual supervision models and compare them to monolingual paraphrases with only a word averaging composition function. While previous work on using monolingual paraphrases (Wieting et al., 2016) show that word averaging outperforms other composition functions (e.g., LSTMs) in similar evaluation settings, it is not really clear whether the same result holds when using bilingual supervision. Also, while the models I evaluate can be trained on any language pair, all my experiments are limited to using Spanish as supervision to learn English representations. It is not clear how switching to a different supervision language improves or degrades the results. It is also not clear whether learning representations for languages other than English can benefit from bilingual supervision or not. Finally, I only show that bilingual supervision outperforms monolingual paraphrases when both are initialized randomly. It is not clear whether the same conclusion holds when both supervisions are used to train well-initialized representations.

In Chapter 4, I use QBLINK (Elgohary et al., 2018) to evaluate models for open-domain conversational QA. While information about previously asked questions and their answers can still be useful for answering questions, the individual QBLINK questions are fully specified (Figure 2.4), and can be answered without information about previous questions. So far, QBLINK is the only available dataset for *entity-centric* conversational QA. In Section 4.2, I study question in-context rewriting. My experiments only evaluate the output of the rewriting models in comparison to the reference rewrites. I do not evaluate the extent to which question rewriting helps QA.

In Chapters 5 and 6, I follow the assumptions of SPIDER (Yu et al., 2018c) about the complexity of SQL queries. In particular, SPIDER queries contains at most one sub-query. My SQL Edit scheme (Section 6.1) is based on that assumption. I also use exact set match (SPIDER's official measure) to compare inferred SQL queries to the reference queries. As discussed in Section 2.3.4, exact set match introduces false negatives and recent work (Zhong et al., 2020) has introduced alternative measures.

Also, I note that my SQL explanations are not supported for all queries in SPIDER. The explanation templates that I implemented only covers 85% of SPIDER's queries. I evaluated the usefulness of those explanations based on the correctness of the crowdsourced feedback. Still, an explicit user study that focuses on the quality and understandability of the explanations can provide more insights that guide future revisions of such explanations.

Finally, the NL-EDIT model I present in Chapter 6 assumes that the feedback always describes an edit. As shown in the analyses in Sections 5.3 and 6.5, that assumption does not always hold in SPLASH (See the first example in Table 6.4). Ideally, the correct model should also be able to process non-editing feedback (e.g., a corrected explanation) as well as feedback that describes edits to be made to the initial explanation.

7.3 Future Research on Collaborative Human–Computer NLP

Drawing from the conclusions and outcomes of this thesis, In the following sections I outline broad research directions for further advancing human–computer NLP in general.

7.3.1 User-Centered Explanations

The ability of users to provide useful feedback on the output of a given model is largely determined by how well the model communicates its reasoning process to users. It is hard to imagine what useful feedback a user would provide when they ask a QA system "How many hotels are in Arlington?" and only get "Ten" without any explanation of how the system reached that answer. An essential component for building collaborative NLP (and Artificial Intelligence in general) is *explainable* models.

Studying the interpretability of NLP and machine learning models has received a lot of attention in the past few years (Belinkov et al., 2020). Such efforts have mainly focused on understanding what models learn (Ettinger et al., 2018; Feng et al., 2018), how they operate internally (Wiegreffe and Pinter, 2019), and what their limitations are (Ribeiro et al., 2020). While very useful for model developers, it is not clear whether such methods can be useful for end users or not. For collaborative NLP, we need to design and evaluate explanation methods based on their usefulness to end users.

An important insight about the explanations I used for my semantic parsing work (Chapters 5 and 6) is that since the models produce a logical form that is executed to deterministically produce the answer, I only had to explain the logical forms and that is enough *for users* to verify the correctness of the produced answer. I did not need to explain how the black-box neural model produced the logical forms.

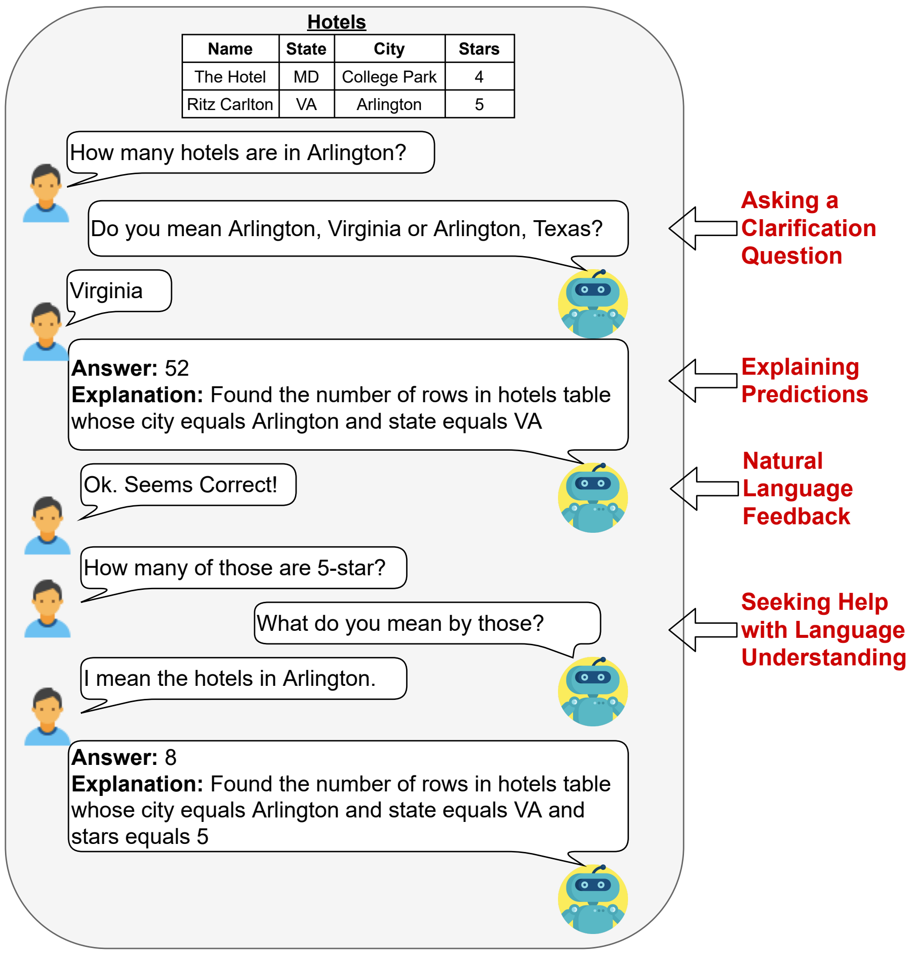
I envision a general framework for producing user-centered explanations while still retaining the accuracy of neural models. In that framework the task is split into two steps: 1) Infer a logical form (executable steps), 2) Deterministically execute the inferred logical forms to produce the result. Under that framework, we only need to explain the logical forms to users and have them interact with (e.g., provide feedback on) the explanations. Also, deep neural (black-box) models can still be used to infer the logical forms.

Besides semantic parsing (which directly fits in the envisioned framework), there are several other NLP tasks that can benefit from user interaction e.g., summarizing, machine translation, and information extraction. Future work needs to explore how to explain the outputs of models for such tasks to users. Following the framework I propose, we need to cast each task of interest as *parsing into a logical form*. For example, abstractive summarizing can be cast in a logical form that consists of two operations: `EXTRACT`ing key sentences and `COMPRESS`ing them. Similarly, the QED formalism for explainable question answering (Lamm et al., 2020) helps place machine reading comprehension within that framework.

Finally, in my interactive semantic parsing work, I chose to explain the inferred SQL queries in the form of natural language steps. The effectiveness of such explanations was only evaluated based on their usefulness to crowdworkers who were paid to make effort to understand the explanations and assess their correctness. Future work still needs to conduct more realistic user studies to judge whether natural language explanations are the most convenient to users of text-to-SQL systems and whether queries can be explained in other forms than (or in addition to) natural language steps.

7.3.2 Designing and Evaluating Interaction Mechanisms

The interaction mechanism I adopt in Chapters 5 and 6 is based on generating a full initial parse that users inspect and correct with natural language feedback. A complementary mechanism is to *enable parsers to ask clarification questions* while generating the parse (Figure 7.1). The parser can ask users to clarify ambiguous questions as "Do you mean Arlington, Virginia or Arlington Texas?" or ask for assistance with language understanding, e.g., resolving coreferences and ellipsis encountered in conversational questions as "What do you mean by 'those' ?". In Chapter 6, I found that even when the correction model was not able to fully fix the initial parse, it was still able to fix at least a subset of



Name	State	City	Stars
The Hotel	MD	College Park	4
Ritz Carlton	VA	Arlington	5

How many hotels are in Arlington?

Do you mean Arlington, Virginia or Arlington, Texas?

Virginia

Answer: 52
Explanation: Found the number of rows in hotels table whose city equals Arlington and state equals VA

Ok. Seems Correct!

How many of those are 5-star?

What do you mean by those?

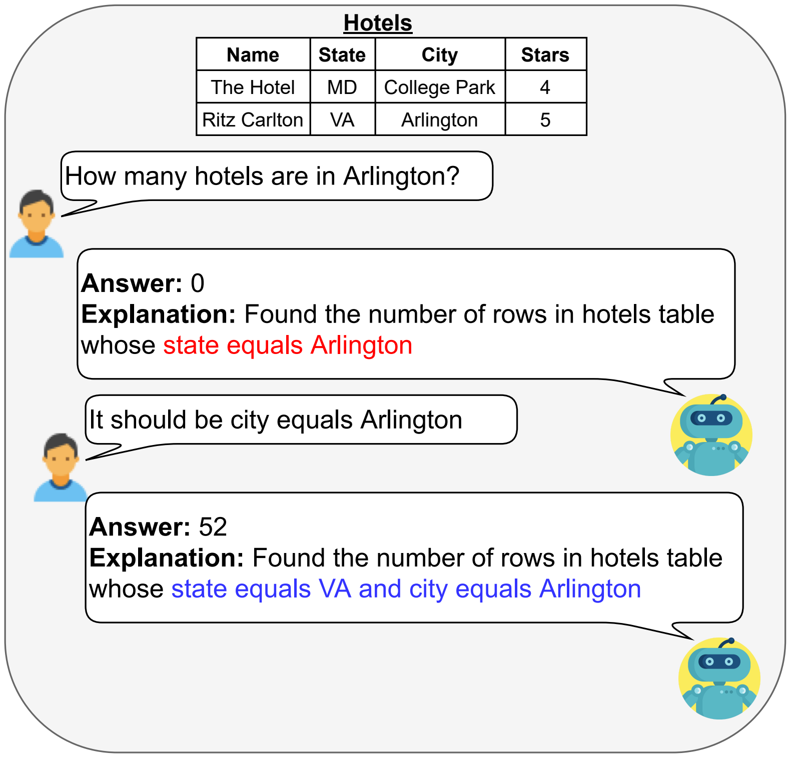
I mean the hotels in Arlington.

Answer: 8
Explanation: Found the number of rows in hotels table whose city equals Arlington and state equals VA and stars equals 5

Asking a Clarification Question
Explaining Predictions
Natural Language Feedback
Seeking Help with Language Understanding

Figure 7.1: Multiple interaction mechanisms that a text-to-SQL system supports. The system is able to decide the best mechanism to use in each situation. Future work needs to develop and evaluate such mechanisms and develop methods for combining them in one system.

the errors in more than 72% of cases. Enabling users to provide *multiple turns of feedback* can significantly improve the overall end-to-end semantic parsing accuracy. Future work needs to study the effectiveness of different interaction mechanisms and explore methods for potentially employing multiple mechanisms for the same task in ways that lead to the best overall user experience (Figure 7.1).



Hotels

Name	State	City	Stars
The Hotel	MD	College Park	4
Ritz Carlton	VA	Arlington	5

How many hotels are in Arlington?

Answer: 0
Explanation: Found the number of rows in hotels table whose state equals Arlington

It should be city equals Arlington

Answer: 52
Explanation: Found the number of rows in hotels table whose state equals VA and city equals Arlington

Figure 7.2: Example interaction with a text-to-SQL system in which the feedback is used to correct a mistake with the initial parse. The underlying parser itself needs to be updated and learn from the given feedback that Arlington is a state not a city.

7.3.3 Learning from Interactions

In the interactive semantic parsing framework I introduce, I only focused on correcting the parse of a given question (the task-at-hand). But the collected interaction logs that consist of triplets of a question, an initial parse, and natural language feedback can be used to improve the parser itself over time hence, reduce the errors it makes. In Figure 7.2, the work I presented in Chapters 5 and 6 uses the provided feedback "It should be city equals Arlington" to revise the initial parse. But the base parser does not really benefit from that feedback—it will keep making the same mistake over and over again. Future work is needed to develop methods that **improve semantic parsers with logs of natural language feedback**. A very promising direction is to employ and extend recent meta-learning algorithms for training and fine-tuning models with noisy supervision (Shu et al., 2019). The methods to be developed will potentially have a broader impact as natural language becomes widely used for providing feedback in other NLP tasks. More generally, any development of an interaction mechanism should be accompanied with corresponding methods for improving the base model (that produces initial predictions) over time through such form of interaction.

7.3.4 Testbed for Human-in-the-Loop Adaptation

To further encourage more research efforts on interactive language learning, future work can create **a testbed and evaluation measures** in the same spirit of popular benchmarks for NLP such as GLUE (Wang et al., 2018a). For a given task, it is impractical to assume that we can construct a large training set for each domain, language, and set of relevant linguistic phenomena. While a large body of work studies automatically reusing existing datasets for different settings (through e.g., domain adaptation methods or multilingual representations), future work can also explore **human-in-the-loop adaptation** methods. Such methods start with initial models trained on existing datasets and occasionally ask for human assistance with unseen test examples that they struggle with and, over time, incorporate the provided assistance to improve the initial model. For example, a conversational QA system (Section 2.2.2) can start with a model trained only to answer stand-alone questions, ask for stand-alone rewrites (Section 4.2) of questions it cannot understand in context, and use the provided rewrites to improve its ability to answer future conversational questions. The same idea applies to generalizing to new languages and domains: the system can ask for paraphrases or translation (even if non-fluent) as meaning annotation (Chapter 3) for words or phrases it struggles to understand. The testbed I envision defines a set of adaptation scenarios and provides methods under evaluation with simulated user responses to the questions they ask while monitoring the improvement in accuracy that each method achieves as a function of the number of user interactions. That testbed will significantly encourage and speed up the development of methods that continuously acquire language skills through human interaction.

7.4 Other Future Research on Interactive Semantic Parsing

So far, my work on interactive semantic parsing has been focused on answering individual questions. Future work can focus on integrating the natural language feedback mechanism inside conversational semantic parsing settings (Section 2.3.3). In that case, the system needs to distinguish and process differently new context-dependent questions from feedback utterances on the inferred parses. Conversational semantic parsing can also benefit from the SQL editing language that I introduce in Section 6.1: Instead of generating a SQL query after each user utterance, the system can generate an edit to a previously generated query.

The correction accuracy of the NL-EDIT model I present in Chapter 6 reaches up to 41% which is still too far from the estimated upper-bound of 81% (Section 5.4). More research still needs to be conducted to improve correction models. Possible avenues for improving are 1) better modeling the interaction between the inputs, 2) exploring different patterns for encoder-decoder attention, and 3) considering different methods for training with synthetic data, e.g., curriculum learning (Bengio et al., 2009).

Finally, recall that SPLASH is based on SPIDER (Section 5.2.3). As I point out in

Section 2.3.4, SPIDER does not provide the most realistic evaluation of text-to-SQL systems. Recent efforts (Suhr et al., 2020; Deng et al., 2021) propose more challenging/realistic evaluation setups (see Section 2.3.4 for details). Future work can study the effectiveness of the interaction framework and models I present in this thesis on other evaluation setups and datasets than SPIDER.

Bibliography

Eugene Agichtein, Eric Brill, and Susan Dumais. 2006. Improving web search ranking by incorporating user behavior. In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpioa, Montse Maritxalar, Rada Mihalcea, et al. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In Proceedings of the International Workshop on Semantic Evaluation.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In Proceedings of the International Workshop on Semantic Evaluation.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In Proceedings of the International Workshop on Semantic Evaluation.

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In Proceedings of the International Workshop on Semantic Evaluation.

C. Alberti, Kenton Lee, and Michael Collins. 2019. A BERT baseline for the natural questions. ArXiv, abs/1901.08634.

Vamshi Ambati, Stephan Vogel, and Jaime Carbonell. 2010. Active learning and crowd-sourcing for machine translation. In Proceedings of the Language Resources and Evaluation Conference.

Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. AI Magazine, 35.

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. Transactions of the Association for Computational Linguistics.

Ion Androutsopoulos, Graeme D Ritchie, and Peter Thanisch. 1995. Natural language interfaces to databases–an introduction. Natural language engineering, 1.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection.

Jimmy Ba, J. Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. ArXiv, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In Proceedings of the International Conference on Learning Representations.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In Proceedings of the Association for Computational Linguistics.

Yonatan Belinkov, Sebastian Gehrmann, and Ellie Pavlick. 2020. Interpretability and analysis in neural NLP. In Proceedings of the Association for Computational Linguistics.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. In Proceedings of Empirical Methods in Natural Language Processing.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. ArXiv, abs/2004.05150.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. The journal of machine learning research, 3.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In Proceedings of the International Conference of Machine Learning.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In Proceedings of Empirical Methods in Natural Language Processing.

Ben Bogin, Jonathan Berant, and Matt Gardner. 2019. Representing schema structure with graph neural networks for text-to-SQL parsing. In Proceedings of the Association for Computational Linguistics.

SRK Branavan, David Silver, and Regina Barzilay. 2012. Learning to win by reading manuals in a monte-carlo framework. Journal of Artificial Intelligence Research, 43.

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Wojciech Gajewski, Andrea Gesmundo, Neil Houlsby, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In Proceedings of the International Conference on Learning Representations.

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica S. Lam. 2020. Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking. In Proceedings of the Association for Computational Linguistics.

Giovanni Campagna, Silei Xu, Mehrad Moradshahi, Richard Socher, and Monica S. Lam. 2019. Genie: A generator of natural language semantic parsers for virtual assistant commands. In Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In Association for Computational Linguistics.

Zhi Chen, Lu Chen Hanqi Li, Ruisheng Cao, Dan Ma, Mengyue Wu, and Kai Yu. 2021. Decoupled dialogue modeling and semantic parsing for multi-turn text-to-sql. ArXiv, abs/2106.02282.

Jianpeng Cheng, Siva Reddy, and Mirella Lapata. 2018. Building a neural semantic parser from a domain ontology. ArXiv, abs/1812.10037.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In Proceedings of Empirical Methods in Natural Language Processing.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In Association for Computational Linguistics.

Edgar F Codd. 1974. Seven steps to rendezvous with the casual user. IBM Corporation.

Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2019. Cast 2019: The conversational assistance track overview. In In Proceedings of the Text Retrieval Conference.

Van Dang and Bruce W Croft. 2010. Query reformulation using anchor text. In Proceedings of ACM International Conference on Web Search and Data Mining.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. In Proceedings of the International Conference on Learning Representations.

Pradeep Dasigi, Nelson F. Liu, Ana Marasovic, Noah A. Smith, and Matt Gardner. 2019. QUOREF: A reading comprehension dataset with questions requiring coreferential reasoning. In Proceedings of Empirical Methods in Natural Language Processing.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In Conference of the North American Chapter of the Association for Computational Linguistics.

Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2021. Structure-grounded pretraining for text-to-SQL. In Conference of the North American Chapter of the Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Conference of the North American Chapter of the Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In Proceedings of the Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In Proceedings of the Association for Computational Linguistics.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In Association for Computational Linguistics.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In Proceedings of the Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Conference of the North American Chapter of the Association for Computational Linguistics.

Bradley Efron and Robert J Tibshirani. 1994. An introduction to the bootstrap. CRC press.

Ahmed Elgohary and Marine Carpuat. 2016. Learning monolingual compositional representations via bilingual supervision. In Proceedings of the Association for Computational Linguistics.

Ahmed Elgohary, Saghar Hosseini, and Ahmed Hassan Awadallah. 2020. Speak to your parser: Interactive text-to-SQL with natural language feedback. In Proceedings of the Association for Computational Linguistics.

Ahmed Elgohary, Christopher Meek, Matthew Richardson, Adam Fourney, Gonzalo Ramos, and Ahmed Hassan Awadallah. 2021. NL-EDIT: Correcting semantic parse errors through natural language interaction. In Conference of the North American Chapter of the Association for Computational Linguistics.

Ahmed Elgohary, Denis Peskov, and Jordan Boyd-Graber. 2019. Can you unpack that? learning to rewrite questions-in-context. In Proceedings of Empirical Methods in Natural Language Processing.

Ahmed Elgohary, Chen Zhao, and Jordan Boyd-Graber. 2018. Dataset and baselines for sequential open-domain question answering. In Proceedings of Empirical Methods in Natural Language Processing.

Allyson Ettinger, Ahmed Elgohary, Colin Phillips, and Philip Resnik. 2018. Assessing composition in sentence vector representations. In Proceedings of International Conference on Computational Linguistics.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for NLP.

Shi Feng, Eric Wallace, Alvin Grissom II, Pedro Rodriguez, Mohit Iyyer, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretation difficult. In Proceedings of Empirical Methods in Natural Language Processing.

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 Association for Computational Linguistics.

Nicholas FitzGerald, Julian Michael, Luheng He, and Luke Zettlemoyer. 2018. Large-scale QA-SRL parsing. In Proceedings of the Association for Computational Linguistics.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In Conference of the North American Chapter of the Association for Computational Linguistics.

Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2018. Dialog-to-Action: Conversational question answering over a large-scale knowledge base. In Proceedings of Advances in Neural Information Processing Systems.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In Proceedings of the Association for Computational Linguistics.

Rahul Gupta, Soham Pal, Aditya Kanade, and Shirish Shevade. 2017. Deepfix: Fixing common c language errors by deep learning. In Association for the Advancement of Artificial Intelligence.

Izzeddin Gur, Semih Yavuz, Yu Su, and Xifeng Yan. 2018. DialSQL: Dialogue based structured query generation. In Proceedings of the Association for Computational Linguistics.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In Computer Vision and Pattern Recognition.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In Proceedings of Empirical Methods in Natural Language Processing.

Luheng He, Julian Michael, Mike Lewis, and Luke Zettlemoyer. 2016b. Human-in-the-loop parsing. In Proceedings of Empirical Methods in Natural Language Processing.

Gary G Hendrix, Earl D Sacerdoti, Daniel Sagalowicz, and Jonathan Slocum. 1978. Developing a natural language interface to complex data. ACM Transactions on Database Systems, 3.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. In Proceedings of the Association for Computational Linguistics.

Jonathan Herzig, P. Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. TAPAS: Weakly supervised table parsing via pre-training. In Proceedings of the Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation, 9.

Hsin-Yuan Huang, Eunsol Choi, and Wen tau Yih. 2019. FlowQA: Grasping flow in history for conversational machine comprehension. In Proceedings of the International Conference on Learning Representations.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In Proceedings of the Association for Computational Linguistics.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In North American Association for Computational Linguistics.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In Proceedings of the Association for Computational Linguistics.

Mohit Iyyer, Wen tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In Proceedings of the Association for Computational Linguistics.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In Proceedings of the Association for Computational Linguistics.

Karen Sparck Jones and Julia R Galliers. 1995. Evaluating natural language processing systems: An analysis and review, volume 1083.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Proceedings of the Association for Computational Linguistics.

Siddharth Karamcheti, Edward Clem Williams, Dilip Arumugam, Mina Rhee, Nakul Gopalan, Lawson L.S. Wong, and Stefanie Tellex. 2017. A tale of two DRAGGNs: A hybrid approach for interpreting action-oriented and goal-oriented instructions. In Proceedings of the First Workshop on Language Grounding for Robotics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In Proceedings of Empirical Methods in Natural Language Processing.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors.

Guillaume Klein, Yoon Kim, Yuntian Deng, Vincent Nguyen, Jean Senellart, and Alexander Rush. 2018. OpenNMT: Neural machine translation toolkit. In Proceedings of Association for Machine Translation in the Americas.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Proceedings of Empirical Methods in Natural Language Processing.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In MT summit.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In Proceedings of the Association for Computational Linguistics.

Igor Labutov, Bishan Yang, and Tom Mitchell. 2018. Learning to learn semantic parsers from natural language supervision. In Proceedings of Empirical Methods in Natural Language Processing.

Matthew Lamm, Jennimaria Palomaki, C. Alberti, D. Andor, Eunsol Choi, Livio Baldini Soares, and Michael Collins. 2020. Qed: A framework and dataset for explanations in question answering. ArXiv, abs/2009.06354.

Guillaume Lample and Alexis Conneau. 2019. Cross-lingual language model pretraining.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942.

Carolin Lawrence and Stefan Riezler. 2018. Improving a neural semantic parser by counterfactual learning from human bandit feedback. In Proceedings of the Association for Computational Linguistics.

M. Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. 2020. Pre-training via paraphrasing.

Fei Li and HV Jagadish. 2014. Constructing an interactive natural language interface for relational databases. In Proceedings of the VLDB Endowment.

Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2017a. Dialogue learning with human-in-the-loop. In Proceedings of the International Conference on Learning Representations.

Jiwei Li, Alexander H. Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2017b. Learning through dialogue interactions by asking questions. In Proceedings of the International Conference on Learning Representations.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. In Proceedings of the International Conference on Learning Representations.

Huan Ling and Sanja Fidler. 2017. Teaching machines to describe images via natural language feedback. In Proceedings of Advances in Neural Information Processing Systems.

Bing Liu, Gokhan Tür, Dilek Hakkani-Tür, Pararth Shah, and Larry Heck. 2018. Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. In Conference of the North American Chapter of the Association for Computational Linguistics.

Ye Liu, Chenwei Zhang, Xiaohui Yan, Yi Chang, and Philip S Yu. 2019a. Generative question refinement with deep reinforcement learning in retrieval-based QA system. In Proceedings of the ACM International Conference on Information and Knowledge Management.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. ArXiv, abs/1907.11692.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In Proceedings of Empirical Methods in Natural Language Processing.

Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. 2014. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. Proceedings of the International Workshop on Semantic Evaluation.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In Proceedings of the Association for Computational Linguistics.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. Psychometrika, 12.

Tomas Mikolov, Kai Chen, G. Corrado, and J. Dean. 2013a. Efficient estimation of word representations in vector space. In Proceedings of the International Conference on Learning Representations.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In Proceedings of Advances in Neural Information Processing Systems.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In Proceedings of the Association for Computational Linguistics.

Arpit Narechania, Adam Fourney, Bongshin Lee, and Gonzalo Ramos. 2021. DIY: Assessing the correctness of natural language to sql systems. In International Conference on Intelligent User Interfaces.

Khanh Nguyen, Hal Daumé III, and Jordan Boyd-Graber. 2017. Reinforcement learning for bandit neural machine translation with simulated human feedback. In Proceedings of Empirical Methods in Natural Language Processing.

Fredrik Olsson. 2009. A literature survey of active machine learning in the context of natural language processing.

Aäron van den Oord, S. Dieleman, H. Zen, K. Simonyan, Oriol Vinyals, A. Graves, Nal Kalchbrenner, A. Senior, and K. Kavukcuoglu. 2016. WaveNet: A generative model for raw audio. arXiv preprint arXiv:1609.03499.

Sheena Panthaplackel, Pengyu Nie, Milos Gligoric, Junyi Jessy Li, and Raymond Mooney. 2020. Learning to update natural language comments based on code changes. In Proceedings of the Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In Proceedings of the Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In Proceedings of Empirical Methods in Natural Language Processing.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In Conference of the North American Chapter of the Association for Computational Linguistics.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In International Conference on Intelligent User Interfaces.

Chen Qu, Liu Yang, Minghui Qiu, W. Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. 2019a. BERT with history modeling for conversational question answering. In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval.

Chen Qu, Liu Yang, Minghui Qiu, Yongfeng Zhang, Cen Chen, W. Bruce Croft, and Mohit Iyyer. 2019b. Attentive history selection for conversational question answering. In Proceedings of the ACM International Conference on Information and Knowledge Management.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation and semantic parsing. In Proceedings of the Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. OpenAI Blog, 1.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In Proceedings of Empirical Methods in Natural Language Processing.

Sudha Rao and Hal Daumé III. 2018. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In Proceedings of the Association for Computational Linguistics.

Pushpendre Rastogi, Arpit Gupta, Tongfei Chen, and Lambert Mathias. 2019. Scaling multi-domain dialogue state tracking via query reformulation. In Conference of the North American Chapter of the Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. CoQA: A conversational question answering challenge. Transactions of the Association for Computational Linguistics.

Marco Túlio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. In Proceedings of the Association for Computational Linguistics.

Pedro Rodriguez, Shi Feng, Mohit Iyyer, He He, and Jordan Boyd-Graber. 2019. Quizbowl: The case for incremental question answering. ArXiv.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in bertology: What we know about how bert works. Transactions of the Association for Computational Linguistics, 8.

Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktaschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. Interpretation of natural language rules in conversational machine reading. In Proceedings of Empirical Methods in Natural Language Processing.

Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In AAAI Conference on Artificial Intelligence.

Gerard Salton and Chris Buckley. 1987. Term weighting approaches in automatic text retrieval. Technical report, Cornell University.

Torsten Scholak, Raymond Li, Dzmitry Bahdanau, Harm de Vries, and Chris Pal. 2020. DuoRAT: Towards simpler text-to-SQL models. ArXiv preprint arXiv:2010.11119.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In Proceedings of the Association for Computational Linguistics.

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In Proceedings of the Software Demonstrations of the Conference of the European Chapter of the Association for Computational Linguistics.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In Proceedings of the International Conference on Learning Representations.

Iulian Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C. Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In Association for the Advancement of Artificial Intelligence.

Burr Settles. 2010. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play. ArXiv preprint., 1801.04871/1801.04871.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In Conference of the North American Chapter of the Association for Computational Linguistics.

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

Richard Shin, Miltiadis Allamanis, Marc Brockschmidt, and Oleksandr Polozov. 2019. Program synthesis and semantic parsing with learned code idioms. In Proceedings of Advances in Neural Information Processing Systems.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-Weight-Net: Learning an explicit mapping for sample weighting.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, A. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of Empirical Methods in Natural Language Processing.

Shashank Srivastava, Igor Labutov, and Tom Mitchell. 2018. Zero-shot learning of classifiers from natural language quantification. In Proceedings of the Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2020. Seq2Edits: Sequence transduction using span-level edit operations. In Proceedings of Empirical Methods in Natural Language Processing.

Hui Su, Xiaoyu Shen, Rongzhi Zhang, Fei Sun, Pengwei Hu, Cheng Niu, and Jie Zhou. 2019. Improving multi-turn dialogue modelling with utterance ReWriter. In Proceedings of the Association for Computational Linguistics.

Yu Su, Ahmed Hassan Awadallah, Madian Khabsa, Patrick Pantel, and Michael Gamon. 2017. Building natural language interfaces to web APIs. In Proceedings of the ACM International Conference on Information and Knowledge Management.

Yu Su, Ahmed Hassan Awadallah, Miaosen Wang, and Ryen W White. 2018. Natural language interfaces with fine-grained user interaction: A case study on web APIs. In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval.

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. 2020. Exploring unexplored generalization challenges for cross-database semantic parsing. In Proceedings of the Association for Computational Linguistics.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In Conference of the North American Chapter of the Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In Proceedings of Advances in Neural Information Processing Systems.

Swabha Swayamdipta, Ankur P Parikh, and Tom Kwiatkowski. 2018. Multi-mention learning for reading comprehension with neural cascades. In International Conference on Learning Representations.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In Proceedings of the Association for Computational Linguistics.

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In Proceedings of the Association for Computational Linguistics.

Svitlana Vakulenko, S. Longpre, Zhucheng Tu, and R. Anantha. 2020. A wrong answer or a wrong question? an intricate relationship between question reformulation and answer selection in conversational question answering. In International Workshop on Search-Oriented Conversational AI (SCAI).

Svitlana Vakulenko, Shayne Longpre, Zhucheng Tu, and Raviteja Anantha. 2021. Question rewriting for conversational question answering. In Proceedings of ACM International Conference on Web Search and Data Mining.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of Advances in Neural Information Processing Systems.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In Proceedings of Advances in Neural Information Processing Systems.

Eric Wallace, Pedro Rodriguez, Shi Feng, Ikuya Yamada, and Jordan Boyd-Graber. 2019. Trick me if you can: Human-in-the-loop generation of adversarial question answering examples. In Transactions of the Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018a. Glue: A multi-task benchmark and analysis platform for natural language understanding. In BlackboxNLP@EMNLP.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In Proceedings of the Association for Computational Linguistics.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018b. Evidence aggregation for answer re-ranking in open-domain question answering. In International Conference on Learning Representations.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In Proceedings of the Association for Computational Linguistics.

Keenon Werling, Arun Tejasvi Chaganty, Percy S Liang, and Christopher D Manning. 2015. On-the-job learning with bayesian decision theory. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Proceedings of Advances in Neural Information Processing Systems, pages 3465–3473. Curran Associates, Inc.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. In Proceedings of Empirical Methods in Natural Language Processing.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. Transactions of the Association for Computational Linguistics.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In Proceedings of the International Conference on Learning Representations.

W. A. Woods, Ronald M Kaplan, and Bonnie L. Webber. 1972. The lunar sciences natural language information system: Final report. BBN Report 2378.

Y. Wu, M. Schuster, Z. Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, M. Krikun, Yuan Cao, Q. Gao, Klaus Macherey, J. Klingner, Apurva Shah, M. Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Y. Kato, Taku Kudo, H. Kazawa, K. Stevens, George Kurian, Nishant Patil, W. Wang, C. Young, J. Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, G. Corrado, Macduff Hughes, and J. Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. ArXiv, abs/1609.08144.

Jingjing Xu, Yuechen Wang, Duyu Tang, Nan Duan, Pengcheng Yang, Qi Zeng, Ming Zhou, and Xu SUN. 2019. Asking clarification questions in knowledge-based question answering. In Proceedings of Empirical Methods in Natural Language Processing.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on freebase via relation extraction and textual evidence. In Association for Computational Linguistics.

Peng Xu, Wenjie Zi, H. Shahidi, 'Akos K'ad'ar, Keyi Tang, Wei Yang, Jawad Ateeq, Harsh V. Barot, Meidan Alon, and Yanshuai Cao. 2021. Turing: an accurate and interpretable multi-hypothesis cross-domain natural language database interface. In Proceedings of the Association for Computational Linguistics.

Silei Xu, Sina Semnani, Giovanni Campagna, and Monica Lam. 2020. AutoQA: From databases to QA semantic parsers with only synthetic training data. In Proceedings of Empirical Methods in Natural Language Processing.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Proceedings of Empirical Methods in Natural Language Processing.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In Proceedings of the Association for Computational Linguistics.

Ziyu Yao, Xiujun Li, Jianfeng Gao, Brian Sadler, and Huan Sun. 2019a. Interactive semantic parsing for if-then recipes via hierarchical reinforcement learning. In Association for the Advancement of Artificial Intelligence.

Ziyu Yao, Yu Su, Huan Sun, and Wen-tau Yih. 2019b. Model-based interactive semantic parsing: A unified framework and a text-to-SQL case study. In Proceedings of Empirical Methods in Natural Language Processing.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In Proceedings of the Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2019. Reranking for neural semantic parsing. In Proceedings of the Association for Computational Linguistics.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In Proceedings of the Association for Computational Linguistics.

Kang Min Yoo, Youhyun Shin, and Sang goo Lee. 2019. Data augmentation for spoken language understanding via joint variational generation. In Association for the Advancement of Artificial Intelligence.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018a. Fast and accurate reading comprehension by combining self-attention and convolution. In Proceedings of the International Conference on Learning Representations.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2021a. GraPPa: Grammar-augmented pre-training for table semantic parsing. In Proceedings of the International Conference on Learning Representations.

Tao Yu, Michihiro Yasunaga, Kai Yang, Rui Zhang, Dongxu Wang, Zifan Li, and Dragomir Radev. 2018b. SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In Proceedings of Empirical Methods in Natural Language Processing.

Tao Yu, Rui Zhang, Heyang Er, Suyi Li, Eric Xue, Bo Pang, Xi Victoria Lin, Yi Chern Tan, Tianze Shi, Zihan Li, Youxuan Jiang, Michihiro Yasunaga, Sungrok Shim, Tao Chen, Alexander Fabbri, Zifan Li, Luyao Chen, Yuwen Zhang, Shreya Dixit, Vincent Zhang, Caiming Xiong, Richard Socher, Walter Lasecki, and Dragomir Radev. 2019a. CoSQL: A conversational text-to-SQL challenge towards cross-domain natural language interfaces to databases. In Proceedings of Empirical Methods in Natural Language Processing.

Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2021b. SCoRe: Pre-training for context representation in conversational semantic parsing. In Proceedings of the International Conference on Learning Representations.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018c. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In Proceedings of Empirical Methods in Natural Language Processing.

Tao Yu, Rui Zhang, Michihiro Yasunaga, Yi Chern Tan, Xi Victoria Lin, Suyi Li, Irene Li Heyang Er, Bo Pang, Tao Chen, Emily Ji, Shreya Dixit, David Proctor, Sungrok Shim, Vincent Zhang Jonathan Kraft, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019b. Sparc: Cross-domain semantic parsing in context. In Proceedings of the Association for Computational Linguistics.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. Swag: A large-scale adversarial dataset for grounded commonsense inference. In Proceedings of Empirical Methods in Natural Language Processing.

Luke Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classiication with probabilistic categorial grammars. In Proceedings of Uncertainty in Artificial Intelligence.

Luke S Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In Proceedings of the Association for Computational Linguistics.

Rui Zhang, Tao Yu, He Yang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019. Editing-based sql query generation for cross-domain context-dependent questions. In Proceedings of Empirical Methods in Natural Language Processing.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and W. Dolan. 2020. Dialogpt: Large-scale generative pre-training for conversational response generation. In Proceedings of the Association for Computational Linguistics.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-SQL with distilled test suites. In Proceedings of Empirical Methods in Natural Language Processing.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. arxiv preprint, arxiv/1709.00103.