FANDA: A Novel Approach to Perform Follow-up Query Analysis

Qian Liu[†*], Bei Chen[§], Jian-Guang Lou[§], Ge Jin[◇*], Dongmei Zhang[§]
[†]Beihang University, Beijing, China
[§]Microsoft Research, Beijing, China
[◇]Peking University, Beijing, China
qian.liu@buaa.edu.cn; {beichen, jlou, dongmeiz}@microsoft.com; elvisking@pku.edu.cn

Abstract

Recent work on Natural Language Interfaces to Databases (NLIDB) has attracted considerable attention. NLIDB allow users to search databases using natural language instead of SQL-like query languages. While saving the users from having to learn query languages, multi-turn interaction with NLIDB usually involves multiple queries where contextual information is vital to understand the users' query intents. In this paper, we address a typical contextual understanding problem, termed as follow-up query analysis. In spite of its ubiquity, follow-up query analysis has not been well studied due to two primary obstacles: the multifarious nature of follow-up query scenarios and the lack of high-quality datasets. Our work summarizes typical follow-up query scenarios and provides a new FollowUp dataset with 1000 query triples on 120 tables. Moreover, we propose a novel approach FANDA, which takes into account the structures of queries and employs a ranking model with weakly supervised max-margin learning. The experimental results on FollowUp demonstrate the superiority of FANDA over multiple baselines across multiple metrics.

1 Introduction

Natural Language Interfaces to Databases (NLIDB) relieve users from the burden of learning about the techniques behind the queries. They allow users to query databases using natural language utterances, which offers a better interactive experience compared to conventional approaches. By using semantic parsing techniques, utterances are automatically translated to executable forms (e.g. Structured Query Language or SQL) to retrieve answers from databases. The majority of the previous studies on NLIDB assumes that queries are context-independent and analyzes them separately. However, if we want to make NLIDB systems conform to users' mental models, it is vital to take contextual information into account. As users often pose new queries based on the past turns during a multi-turn interaction with the NLIDB system in a conversation. For example, given a query utterance, *"Show the sales in 2017."*, the user can simply say *"How about 2018?"* instead of the complete query *"Show the sales in 2018."*. In fact, Bertomeu et al. (2006) point out that, in their Wizard-of-Oz experiment, up to 74.58% queries follow immediately after the question they are related to.

In this paper, we focus on immediate follow-up queries, and we formulate the follow-up query analysis problem here. Consider a context-independent question and a question immediately following it, respectively named as *precedent query* and *follow-up query*. Generally speaking, the follow-up query, like *"How about 2018?"* in the above example, would be too ambiguous to be parsed into executable SQL by itself. Therefore, follow-up query analysis aims to generate a *fused query*, which resolves the ambiguity of the follow-up query in the context of its precedent query. Compared to the follow-up query, the fused query reflects users' intent explicitly and facilitates better downstream parsing. In reality, there are various scenarios for follow-up queries, which can make the problem challenging. Setlur et al. (2016) introduce the scenarios of single queries in their realistic system, inspired by which, we summarize typical scenarios of follow-up queries in Table 1. For instance, the follow-up query *"Compare it with Bill Collins."* aims to perform a comparison, and *"Show their number."* belongs to the scenario of calculation and statistics.

Several attempts have been made to analyze follow-up queries in specific datasets. For example, in the air travel domain, the ATIS dataset collects user queries, including follow-up ones, and their corresponding SQL from a realistic flight planning system (Dahl et al. 1994). Using this dataset, Miller et al. (1996) employ a fully statistical model with semantic frames; Zettlemoyer and Collins (2009) train a semantic parser using context-independent data and generate context-dependent logical forms; and Suhr, Iyer, and Artzi (2018) present a relatively complex sequence-to-sequence model. While the ATIS dataset is realistic, it is limited to a particular domain. All these methods are specific to it and hard to transfer across datasets. More recently, the Sequential Question Answering (SQA) dataset is proposed along with a search-based method (Iyyer, Yih, and Chang 2017). However, SQA focuses on relatively simple follow-up scenarios, where the answer to follow-up queries is always a subset of the answer to the precedent query.

While some of the previous efforts somehow follow the idea of semantic parsing methods, typical analysis scenarios, such as compare, group and sort, are not covered in ATIS or SQA. The lack of public high-quality and richer

[*]Work done during an internship at Microsoft Research.

Scenario	Example	
Analytics	Precedent	: In 1995, is there any network named CBC?
	Follow-up	: Any TSN?
	Fused	: In 1995, is there any network named TSN?
Compare	Precedent	: How much money has Smith earned?
	Follow-up	: Compare it with Bill Collins.
	Fused	: Compare money Smith earned with Bill Collins.
Calc & Stats	Precedent	: List all universities founded before 1855.
	Follow-up	: Show their number.
	Fused	: Show the number of all universities founded before 1855.
Extremum	Precedent	: Which stadium has the most capacity?
	Follow-up	: Which get the highest attendance?
	Fused	: Which stadium get the highest attendance?
Filter	Precedent	: How many roles are from studio paramount?
	Follow-up	: List all titles produced by that studio.
	Fused	: List all titles produced by studio paramount.
Group	Precedent	: Show the industry which has the most companies?
	Follow-up	: Show in different countries.
	Fused	: Show the industry which has the most companies in different countries.
Sort	Precedent	: Show all chassis produced after the year 1990.
	Follow-up	: Sort them by year.
	Fused	: Show all chassis produced after the year 1990 and sort by year.
Search	Precedent	: What position did Sid O'Neill play?
	Follow-up	: Which players else are in the same position?
	Fused	: Which players play in the position of Sid O'Neill excluding Sid O'Neill?

Table 1: Typical follow-up scenarios.

datasets makes the problem even more challenging. Taking all the aforementioned limitations into account, we build a new dataset and present a well-designed method for natural language follow-up queries. Our major contributions are:

- We build a new dataset named FollowUp [1],which contains 1000 query triples on 120 tables. To the best of our knowledge, it is the first public dataset that contains various kinds of follow-up scenarios.
- We propose a novel approach, **F**ollow-up **AN**alysis for **DA**tabases (FANDA), to interpret follow-up queries. FANDA considers the structures of queries and employs a ranking model with weakly supervised learning. It is parser-independent and can transfer across domains.
- We conduct experimental studies on the FollowUp dataset. Multiple baselines and metrics are utilized to demonstrate promising results of our model.

2 Follow-up Query Dataset

We create FollowUp dataset with the purpose of offering a high-quality dataset for research and evaluation. We utilize tables from WikiSQL dataset (Zhong, Xiong, and Socher 2017), as they are realistic extracted from the web. Tables with identical columns are joined, from which we randomly select 120 tables with at least 8 rows and 1 numerical column. Data is collected by crowdsourcing of 8 workers, using the format of the triple (precedent query, follow-up query, fused query). The collection is completed through two phases. Firstly, workers write context-independent precedent queries according to the tables. To avoid monotonous queries, we provide several generic prompts to workers such as *"Require sort with an obvious order."*, as Pasupat and Liang (2015). Secondly, given precedent queries, workers write follow-up queries and the equivalent fused queries. We fastidiously provide 10 examples for each follow-up scenario, so that workers can imitate the examples to write different kinds of follow-up scenarios.

FollowUp dataset contains 1000 triples on 120 tables with a vocabulary of size about 2000. All the example triples in Table 1 are from the proposed FollowUp dataset, which has great diversity in follow-up scenarios. Instead of SQL, we collect fused queries in natural language because SQL queries require workers equipped with more expertise. Natural language queries also allow methods for follow-up query analysis to be independent of the semantic parser. Furthermore, this kind of annotation format is embraced by several works on interactive question answering (Raghu et al. 2015; Kumar and Joshi 2016; 2017).

3 Follow-up Analysis for Database

In this section, we present a novel approach FANDA, by which the semantics of follow-up queries can be interpreted with precedent query. Taking the precedent query x and the follow-up query y as inputs, our goal is to obtain a complete fused query z. It has the same meaning with the follow-up query y and can be processed by a downstream semantic parser all alone. Note that x, y and z are all natural language utterances. As the fused query z always overlaps a great deal with the precedent and follow-up queries, it is natural to consider sequence-to-sequence based models. However, they are uninterpretable and require lots of training data, giving no consideration to the semantic structures of

[1] Available at https://github.com/SivilTaram/FollowUp

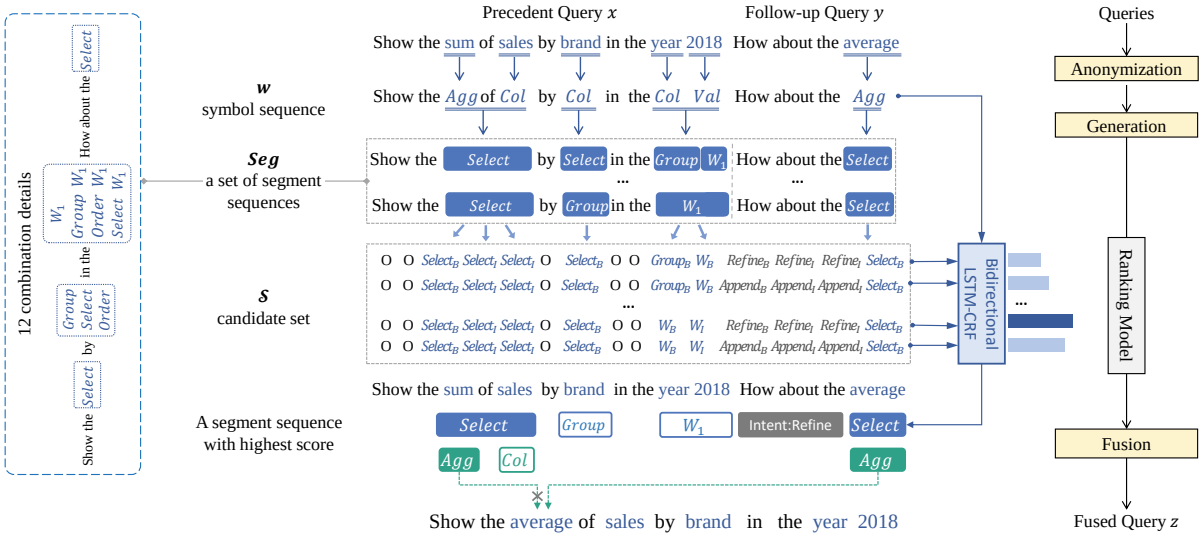

Figure 1: Illustration of FANDA.

queries. In fact, x and y always have conflicting semantic structures. For example, given *"Show the sales in 2017."* and *"How about 2018 ?"*, *"2018"* conflicts with *"2017"*, and only *"2018"* should be kept. Therefore, in FANDA, we carefully consider two-level structures: *symbol*-level and *segment*-level. Symbols are for words, and segments are for phrases related to SQL clauses. Three components, shown as yellow boxes in Figure 1, are devised to obtain the new fused query: (1) **Anonymization**. Symbols are well-designed to simplify the queries, producing the symbol sequences. (2) **Generation**. Segments are presented with compositional deduction rules, producing the best segment sequence. (3) **Fusion**. Fusing x and y using the relationships among two-level structures, producing the fused query z.

3.1 Anonymization

In query utterances, the words can be divided into two types: analysis-specific words and rhetorical words. Analysis-specific words indicate the parameters of SQL clauses explicitly, while rhetorical words form the sentence patterns. As shown in Figure 1, in the precedent query *"Show the sum of sales by brand in the year 2018"*, the words *"sum"*, *"sales"*, *"brand"*, *"year"* and *"2018"* are likely to be analysis-specific words, while the others are rhetorical words. As shown in Table 2, we predefine 8 types of *symbol* for different analysis-specific words. Given a query, anonymization is to recognize all analysis-specific words in it, and replace them with the corresponding symbols to construct a symbol sequence. Following the example in Figure 1, the symbol sequence corresponding to x should be "Show the Agg of Col by Col in the Col Val".

The symbols Col and Val are table-related, while the others are language-related. For table-related symbols, the analysis-specific words can be found from the corresponding table of each query. Note that all the numbers and dates belong to Val. Replacing column names and cell values by

Symbol	Meaning	Examples
Col	Column Name	sale, country
Val	Cell Value	2018, Australia
Agg	Aggregation	sum, maximum, count
Com	Comparison	more, later, before
Dir	Order Direction	descending, ascending
Per	Personal Pronoun	it, he, them
Pos	Possessive Pronoun	its, his, their
Dem	Demonstrative	that, those, other

Table 2: Symbols for analysis-specific words.

Col and Val is the key to equip FANDA with the ability to transfer across tables. For language-related symbols, $\{Per, Pos, Dem\}$ are for pronouns, while the others are for different kinds of SQL operators. Agg corresponds to aggregation function, Com stands for comparison operator, and Dir indicates the direction of ORDERYBY. The meanings of language-related symbols are limited to a narrow space, so it is viable to enumerate the most common analysis-specific words empirically. For example, $Pos \in$ {their, its, his, her} and $Agg \in$ {average, sum, count, maximum, $\cdots$}. Both of the precedent query x and the follow-up query y are anonymized, and the resulting symbol sequences are denoted by $\hat{x}$ and $\hat{y}$ respectively. [2]

3.2 Generation

The symbol of an analysis-specific word reflects its intrinsic semantics, but ignores the content around it. Supposing we have parsed the query *"Show the sum of sales by brand in the year 2018"* into a SQL statement. Although both *"brand"* and *"year"* are with the same symbol Col, they belong to

[2] If an analysis-specific word belongs to multiple symbols, several symbol sequences will be obtained. For example, "those" can be Per or Dem.

Segment	Rule	Segment	Rule
$Select$	$[Agg + [Val]] + Col$	$Group$	Col
$Order$	$[Dir] + Col$	P_1	Per
W_1	$[Col] + [Com] + Val$	P_2	Pos
W_2	$Col + Com + Col$	P_3	$Dem + Col$

Table 3: Segment types and compositional deduction rules. Square brackets indicate optional symbols.

different SQL clauses. Along with the adjacent Val *"2018"*, *"year"* forms a clause WHERE year = 2018. As there are rhetorical words like *"by"* around *"brand"*, it forms a clause GROUPBY brand. Hence, inspired by SQL clauses, we design the structure *segment* to combine symbols and capture the effect of rhetorical words. Each segment can be deduced by one or more adjacent [3] symbols according to the compositional deduction rule. Table 3 shows the well-defined 8 types of segments, along with their compositional deduction rules. W and P stand for $Where$ and $Pronoun$ respectively. Concatenating the symbol sequences $\hat{x}$ and $\hat{y}$ as a whole, the goal of generation is to obtain the correct segment sequence for it. However, there are multiple ways to combine symbols, making it problematic to acquire the correct segment sequence. Therefore, it is cast into a ranking problem. Firstly, symbols are combined to generate all possible segment sequences. Then, a ranking model is built to score these segment sequences and pick the best one as output.

The compositional deduction rules originate from SQL clause syntax. For example, in Figure 1, "*sum of sales* (Agg of Col)" can make up a $Select$ segment, relevant to a SQL clause SELECT SUM(sales). There can also be multiple choices. "*year* (Col)" can be $Select$, $Group$ and $Order$ alone, or be W_1 together with "*2018* (Val)". To make the rules more robust, we leave out the order of symbols. For instance, both ($[Dir]$, Col) and (Col, $[Dir]$) can be composed into segment $Order$.

As the precedent query has a complete structure, the compositional deduction rules can be applied directly. However, ellipsis exists in the follow-up query, so all the symbols in the first 5 rules become optional. Just as the follow-up case *"How about the average"* in Figure 1, segment $Select$ can be deduced by a single Agg without Col. Moreover, symbols in different queries cannot combine. Concatenating $\hat{x}$ and $\hat{y}$, we can generate multiple segment sequences and obtain the set Seg. For the examples in Figure 1, there are 12 resulting segment sequences in Seg, as shown in the left blue dashed box. Then, a ranking model is built to pick the best segment sequence in Seg, which will be introduced in detail in Section 4.

3.3 Fusion

Based on the symbol sequence and the best segment sequence, the fused query z can be obtained. Breaking down the best segment sequence into two parts, one part corresponds to x, and the rest corresponds to y. There are two steps to accomplish the fusion. The first is to find conflicting segment pairs between the two parts. Conflicting means segments have the same or incompatible semantics. Generally speaking, segments of the same type conflict with each other. For instance, the two $Select$ segments conflict in the second case of Figure 2. However, there are particular cases. For W_1, segments conflict only if their inner symbols Val are in the same column. It is the structured characteristic of tables that leads to incompatible semantics among these W_1 segments. As shown in the first case of Figure 2, instead of *"1995"*, *"TSN"* only conflicts with *"CBC"*, for they are both in column *"Network"*. For pronouns related segments, P_1, P_2 and P_3, we empirically design some semantic conflicting rules to resolve them, without considering ambiguities. For instance, P_3 ($Dem + Col$) conflicts with W_1 which describes the same column, such as *"that year"* and *"1995"* in Figure 2. P_1 (Per) conflicts with all words except those already in conflicting pairs, resulting in nested queries in z.

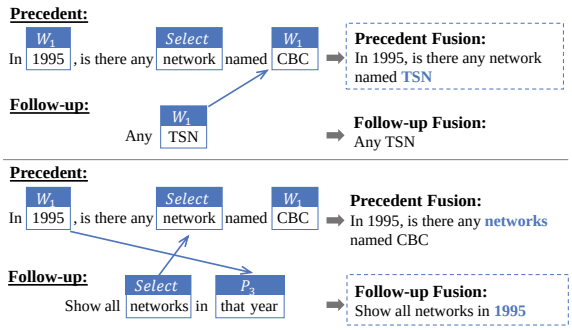

Figure 2: Two fusion cases.

The second step is to perform fusion on these conflicting segment pairs. Generally, we fuse two segments by replacing one with the other. As indicated by different arrow directions in Figure 2, we replace *"CBC"* with *"TSN"*, and replace *"that year"* with *"1995"*. When there is no pronoun, the replacement is symbol-level. Taking the example in Figure 1, "*sum of sales* (Agg of Col)" and "*average* (Agg)" are both $Select$ segments and conflict with each other. Then only Agg *"sum"* is replaced by *"average"*. Although replacement can be applied in most scenarios, it is not suitable for scenarios of compare, where the conflicting segments are presented for side-by-side comparison. Consider a query sequence *"How much money has Smith earned? How about Bill Collins?"*, *"Smith"* should be replaced by *"Bill Collins"*. However, given *"How much money has Smith earned? Compare with Bill Collins."*, *"Bill Collins"* should be added to *"Smith"*. To distinguish the two different situations, we define two intents for follow-up queries: $Append$ for compare and $Refine$ for others. Thus, the output of ranking model turns into the best segment sequence and intent. There are various methods to perform intent classification, and we choose to regard them as two special segments. Finally, precedent fusion and follow-up fusion are obtained. We pick the follow-up fusion as output z if it is different from the follow-up query. Otherwise, we choose the prece-

[3] Adjacent means that there is nothing but rhetorical words between two symbols and their distance in word level is less than a window size (4 in our experiments).

dent fusion, as shown in the blue dotted boxes in Figure 2.

4 Ranking Model

As mentioned, in the process of generation, a ranking model is employed to pick the best segment sequence from Seg. In this section, we introduce how the ranking model works, followed by its learning process with weak supervision.

Intent As previously stated, every follow-up query has an intent. We regard the two intent, $Refine$ and $Append$, as special segments. The intent of a follow-up query is related to its sentence pattern, which we believe contains all the rhetorical words before the first analysis-specific word. As in Figure 1, the sentence pattern *"How about the"* is labeled as intent. Specifically, if there is no word before the first analysis-specific word, the intent is set as $Refine$.

Mapping Inspired by named entity recognition (Sang 2002), we regard segment sequence ranking as the problem of tag sequence ranking. For simplicity, $\{W_1, W_2\}$ are unified into W and $\{P_1, P_2, P_3\}$ are unified into P. An additional segment O, designed for $Others$, is employed for words without existing segments. Moreover, O can also be deduced by the symbols $\{Per, Pos, Dem\}$ in the situation where the pronouns are ambiguous, such as *"that"* used as a conjunction. Employing the IOB (Inside, Outside, Beginning) format (Ramshaw and Marcus 1999), we map Seg into a set of tag sequences termed candidate set $\mathcal{S}$.

One segment sequence usually maps to two tag sequences. As shown in Figure 1, the first two tag sequences are both from the first segment sequence, but have different intent tags $(Refine_B, Refine_I, Refine_I)$ and $(Append_B, Append_I, Append_I)$. The one with higher score represents the final intent of the follow-up query.

Ranking Let $\boldsymbol{w} = (w_1, w_2, \cdots, w_N)$ denote the concatenation of symbol sequences $\hat{x}$ and $\hat{y}$. The candidate set $\mathcal{S} = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \cdots, \boldsymbol{s}_K\}$ contains tag sequence candidates, and the tag sequence can be denoted as $\boldsymbol{s}_k = (t_1^k, t_2^k, \cdots, t_N^k)$. Our goal is to find the best candidate $\boldsymbol{s}^*$, that is:

$$\boldsymbol{s}^* = \arg\max_{\boldsymbol{s}\in\mathcal{S}} g(\boldsymbol{s}|\Theta), \tag{1}$$

where $g(\cdot|\Theta)$ is a score function given parameter set Θ. To this end, we perform tag sequence candidates ranking using a bidirectional LSTM-CRF model (Huang, Xu, and Yu 2015) with weakly supervised max-margin learning. For each $w_i (i = 1, \cdots, N)$, the model computes a hidden state $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$, then the forward hidden state is:

$$\overrightarrow{\mathbf{h}}_i = \overrightarrow{\mathbf{LSTM}}\big(\phi(w_i); \overrightarrow{\mathbf{h}}_{i-1}\big), \tag{2}$$

where ϕ is an embedding function initialized using Glove (Pennington, Socher, and Manning 2014). Let T denote the number of tags, and $\mathbf{f}_i$ denote the T-dimensional network score vector for w_i, which can be computed as:

$$\mathbf{f}_i = \mathbf{h}_i \mathbf{W}, \tag{3}$$

where $\mathbf{W}$ is the learned matrix. Let $\mathbf{A}$ denote the $T \times T$ transition matrix of CRF layer, and the entry $\mathbf{A}_{uv}$ is the probability of transferring from tag u to v. Let θ denote the parameters of network in LSTM. Given $\Theta = \{\mathbf{A}, \mathbf{W}, \theta\}$, the score function for candidate $\boldsymbol{s}_k$ is defined as the sum of two parts: transition score by CRF and network score by bidirectional LSTM, which can be formulated as:

$$g(\boldsymbol{s}_k|\Theta) = \sum_{i=1}^{N} (\mathbf{A}_{t_{i-1}^k t_i^k} + \mathbf{f}_i[t_i^k]), \tag{4}$$

where t_i^k is the corresponding tag of w_i in candidate $\boldsymbol{s}_k$.

Weakly Supervised Learning Finally, we introduce how the bidirectional LSTM-CRF model is learned. As mentioned in Section 2, it is too expensive to annotate SQL, as well as tags. Hence, we utilize the gold fused query in natural language to learn, leading to the weak supervision.

For each tag sequence candidate $\boldsymbol{s}_k \in \mathcal{S}$, we can perform fusion based on its corresponding segment sequence and intent (Section 3.3) , and obtain a natural language query z_k. Let z^* denote the gold fused query. To compare z_k and z^*, we process them by anonymization (Section 3.1), while the pronouns are ignored. Then we check their symbols. If they have the same symbols with the same corresponding words, we call them *symbol consistent* and put $\boldsymbol{s}_k$ in the positive set $\mathcal{P}$; otherwise, they are symbol inconsistent and $\boldsymbol{s}_k$ is put in the negative set $\mathcal{N}$. As we can see, $\mathcal{S} = \mathcal{P} \cup \mathcal{N}$. However, the tag sequences in $\mathcal{P}$ are not all correct. After fusion and anonymization, the sequences with wrong tags may result in symbol consistence by chance. Only one tag sequence in $\mathcal{S}$ may be correct, and the correct one is always in $\mathcal{P}$. As symbol consistence is the requirement of correctness on tags. Therefore, we calculate the scores of all tag sequences in $\mathcal{S}$, and select the highest ones from $\mathcal{P}$ and $\mathcal{N}$:

$$\widehat{\boldsymbol{s}}_p = \arg\max_{\boldsymbol{s}\in\mathcal{P}} g(\boldsymbol{s}|\Theta), \quad \widehat{\boldsymbol{s}}_n = \arg\max_{\boldsymbol{s}\in\mathcal{N}} g(\boldsymbol{s}|\Theta). \tag{5}$$

Then a max-margin learning method is employed to encourage a margin of at least Δ between $\widehat{\boldsymbol{s}}_p$ and $\widehat{\boldsymbol{s}}_n$. Considering various lengths of different inputs, normalization factors are added to the scores. The hinge penalty is formulated as:

$$\max(0, \Delta - \frac{g(\widehat{\boldsymbol{s}}_p|\Theta)}{|\widehat{\boldsymbol{s}}_p|} + \frac{g(\widehat{\boldsymbol{s}}_n|\Theta)}{|\widehat{\boldsymbol{s}}_n|}), \tag{6}$$

where $\Delta > 0$ is a hyperparameter.

5 Experiments

We evaluate our methods on the proposed FollowUp dataset, and split the 1000 triples following the sizes 640/160/200 in train/development/test. All the query utterances are preprocessed by anonymization (Section 3.1). In the process of anonymization, dates and numbers are extracted for Val using *Spacy*[4], and person entities are recognized to handle personal pronouns. Moreover, for recognition of *Col* and *Val*, a simple matching algorithm is applied without considering synonyms.

We use three metrics to compare the two natural language queries: the output queries of our methods and the gold fused queries from the dataset. (1) **Symbol Accuracy**. It is the proportion of the output queries that are symbol consistent

[4]https://spacy.io/

	Model	Symbol Acc (%)	BLEU (%)
Dev	SEQ2SEQ	0.63 ± 0.00	21.34 ± 1.14
	COPYNET	17.50 ± 0.87	43.36 ± 0.54
	S2S+ANON	18.75 ± 0.95	41.22 ± 0.33
	COPY+ANON	25.50 ± 2.47	51.45 ± 0.93
	FANDA	**49.00** ± 1.28	**60.14** ± 0.98
Test	CONCAT	22.00 ± –	52.02 ± –
	E2ECR	27.00 ± –	52.47 ± –
	SEQ2SEQ	0.50 ± 0.22	20.72 ± 1.31
	COPYNET	19.30 ± 0.93	43.34 ± 0.45
	S2S+ANON	18.80 ± 1.77	38.90 ± 2.45
	COPY+ANON	27.00 ± 4.32	49.43 ± 1.11
	FANDA	47.80 ± 1.14	59.02 ± 0.54
	– Intent	35.30 ± 0.44	55.01 ± 0.86
	– Ranking	24.30 ± 6.70	52.92 ± 2.24
	+ Pretrain	**48.20** ± 1.02	**59.87** ± 0.43

Table 4: The results of symbol accuracy and BLEU scores.

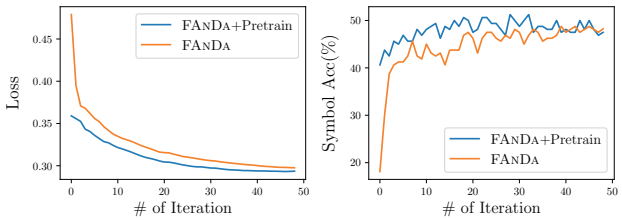

Figure 3: Convergence process on development set.

Model	Execution Accuracy (%)
CONCAT	25.24
E2ECR	27.18
COPY+ANON	40.77
FANDA	**60.19**

Table 5: The results of execution accuracies.

(mentioned in Section 4) with the gold fused ones. It is used to measure the retention of critical information, but without considering the order of symbols. (2) **BLEU**. It automatically assigns a score to each output query based on how similar it is to the gold fused query (Papineni et al. 2002). (3) **Execution Accuracy**. To further evaluate the validity of the output queries, we parse them into SQL and evaluate the execution accuracy manually. Specifically, we use COARSE2FINE (Dong and Lapata 2018), the state-of-the-art semantic parser on WikiSQL, to parse all the 200 gold fused queries in the test set, and take the 103 successful ones. Then the execution accuracy of the 103 corresponding output queries is calculated. The other 97 triples are excluded due to the incapability of the parser.

Our baselines include: (1) **CONCAT**: a simple method that directly concatenates precedent and follow-up queries; (2) **E2ECR**: an end-to-end neural coreference resolution method. We perform evaluation using an already trained model provided by (Lee et al. 2017) as it requires different training data; (3) **SEQ2SEQ**: sequence-to-sequence model with attention (Bahdanau, Cho, and Bengio 2015); (4) **COPYNET**: sequence-to-sequence model with copying mechanism (Gu et al. 2016). For SEQ2SEQ and COPYNET, the input are the concatenation of the precedent and follow-up queries, and the features include word embeddings, part-of-speech tags, table information and so on; (5) **S2S+ANON**: SEQ2SEQ with anonymized inputs; (6) **COPY+ANON**: COPYNET with anonymized inputs. For S2S+ANON and COPY+ANON, inputs are anonymized using Col and Val. For example, the utterance "*In 1995, is there any network named CBC? Any TSN?*" is anonymized as "*In Val#1, is there any Col#1 named Val#2? Any Val#3?*".

5.1 Follow-up Results

Table 4 shows symbol accuracies and BLEU scores on both development and test sets, where we run each experiment five times and report the averages. FANDA–Intent means FANDA equips all follow-up queries with intent $Refine$ without the intent classification; FANDA–Ranking is to exclude the ranking model and select a segment sequence from Seg randomly; and FANDA+Pretrain is to annotate the tags of 100 triples from train set and pre-train the bidirectional LSTM-CRF with supervised learning. We can observe that FANDA significantly outperforms all the baselines, which demonstrates the effectiveness of our model. S2S+ANON and COPY+ANON get better results than SEQ2SEQ and COPYNET respectively, which demonstrates the importance of anonymization. The process of anonymization in FANDA is well-designed and indispensable. Moreover, FANDA–Intent performs worse than FANDA, which shows the reasonability of distinguishing different intents. The bad performance of FANDA–Ranking with a large standard deviation demonstrates the necessity of the ranking model. Unsurprisingly, FANDA+Pretrain performs the best with the help of manual annotations. As shown in Figure 3, pre-training with supervision can speed up the convergence, while the weakly supervised FANDA has the competitive results.

Symbol accuracy is more convincing than BLEU, as the correctness of symbols is a prerequisite of correct execution. Table 5 reports the execution accuracies on 103 test triples. Due to the workload of checking the executable results manually, we only include baselines CONCAT, E2ECR, and the best baseline COPY+ANON. Results demonstrate the superiority of FANDA over baselines in understanding context and interpreting the semantics of follow-up queries. It also shows that FANDA is parser-independent and can be incorporated into any semantic parser to improve the context understanding ability cost-effectively.

5.2 Closer Analysis

Figure 4 shows the partial transition matrix **A** of CRF layer in the ranking model. We observe that transition score from Tag_B to Tag_I is evidently higher than others, suggesting that CRF layer has learned which combinations of tags are more reasonable ($Tag \in \{Select, W, Group, P, Order\}$). Figure 5 shows the evolution of candidate scores in $\mathcal{S}$ of a specific case, as training iteration progresses. In the coordinate system, each point represents a candidate. To scale the scores from different iterations into a unified space, we nor-

No	Case Analysis		
1	Precedent	: What is the result, when the home team score is 2.4.6?	Follow-up : What is the date?
	Gold Fusion	: What is the date, when the home team score is 2.4.6?	
	COPY+ANON	: What is the date, the home team score is 2.4.6?	
	FANDA	: What is the date, when the home team score is 2.4.6?	
2	Precedent	: Which is the draw number of Lowry?	Follow-up : How about Laura?
	Gold Fusion	: Which is the draw number of Laura?	
	COPY+ANON	: Which is the draw number of Lowry?	
	FANDA	: Which is the draw number of Laura?	
3	Precedent	: What are the names when elevation is feet?	Follow-up : Of those, whose GNIS feature is 1417308?
	Gold Fusion	: Of names when elevation is feet, whose GNIS feature is 1417308?	
	COPY+ANON	: What are the names when elevation is 1417308, whose GNIS feature is feet?	
	FANDA	: What are the names when elevation is feet whose GNIS feature is 1417308?	

Table 6: Cases analysis of COPY+ANON and FANDA.

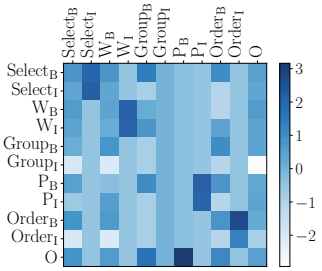

Figure 4: Transition matrix in CRF layer.

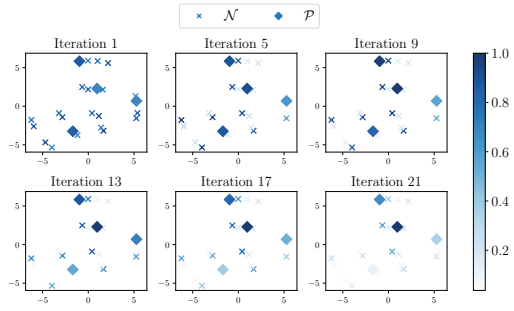

Figure 5: The evolution of candidate scores.

malize them to range $[0, 1]$. In Iteration 1, FANDA assigns different but similar scores to all candidates in $\mathcal{P}$ and $\mathcal{N}$ with random initialization. From Iteration 5 to 21, the score distribution becomes increasingly skewed. The growing gap between the highest score and others verifies the effectiveness of max-margin learning. And from Iteration 13 to the end, the candidate with the highest score remains unchanged, indicating the stability of our weakly supervised learning.

Finally, we analyze three real cases in Table 6 and show the results of the generative model COPY+ANON and our model FANDA. In case 1, both two models perform well. COPY+ANON puts *"date"* in the position of *"result"* according to the same context *"what is the"*, indicating that generative models work well in the situation where a substantial overlap exists between precedent query and follow-up query. FANDA can also deal with the situation, as bidirectional LSTM-CRF assigns *"result"* and *"date"* as $Select$ segment, and then *"result"* is replaced with *"date"*.

However, COPY+ANON performs worse than FANDA mainly in two situations. The first situation is that there is no overlap. As in Case 2, COPY+ANON makes a mistake of ignoring *"Laura"*, which should be used to replace *"Lowry"*, indicating the weak reasoning ability of COPY+ANON. COPY+ANON only uses a learning-based approach, while FANDA takes one step further by introducing the table structure to make judgments. The reason why FANDA replaces *"Lowry"* with *"Laura"* is that they are both in column $Artist$. The second situation is that there is an ambiguous overlap. As in Case 3, there is a general word *"is"* in front of both *"feet"* and *"1417308"*. Influenced by this, COPY+ANON confuses the positions of *"feet"* and *"1417308"*. FANDA can solve the problem because it regards *"elevation is feet"* and *"GNIS feature is 1417308"* as separate segments.

6 Related Work

From the perspective of semantic parsing, our work is related to the analysis of context-independent queries, such as statistical parsing (Popescu et al. 2004; Poon 2013) and sequence-to-sequence based methods (Jia and Liang 2016; Iyer et al. 2017; Dong and Lapata 2018). Specifically, Palakurthi et al. (2015) utilize a CRF model to classify attributes into different SQL clauses, similar to our ranking model. From the perspective of follow-up analysis, there are multiple researches on context-sensitive conversation, such as open-domain response generation using neural networks (Sordoni et al. 2015), conversation agent using reinforcement learning (Shah et al. 2018), contextual question understanding for retrieval system (Ren et al. 2017), and non-sentential utterance resolution in question answering (Kumar and Joshi 2016; 2017), which is similar to our baseline S2S+ANON. Our work is also related to coreference resolution. The recent methods based on deep learning achieve the state-of-the-art performances (Long, Pasupat, and Liang 2016; Clark and Manning 2016; Lee et al. 2017), from which we choose one as our baseline E2ECR. Moreover, several interactive visual analysis systems (Setlur et al. 2016; Dhamdhere et al. 2017; Hoque et al. 2018) take context into account.

7 Conclusion and Future Work

For the purposes of research and evaluation, we create the FollowUp dataset that contains various follow-up scenarios. A novel approach, FANDA, is presented for follow-up query analysis, which considers the structures of queries and employs a ranking model with weakly supervised max-margin learning. The experimental results demonstrate the effectiveness of our model. For future work, we are interested in extending our method to multi-turns and multi-tables.

Acknowledgments

We thank Yihong Chen, Börje Karlsson, and the anonymous reviewers for their helpful comments.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Bertomeu, N.; Uszkoreit, H.; Frank, A.; Krieger, H.-U.; and Jörg, B. 2006. Contextual phenomena and thematic relations in database QA dialogues: results from a Wizard-of-Oz experiment. In *HLT-NAACL*.

Clark, K., and Manning, C. D. 2016. Improving coreference resolution by learning entity-level distributed representations. In *ACL*.

Dahl, D. A.; Bates, M.; Brown, M.; Fisher, W. M.; Hunicke-Smith, K.; Pallett, D. S.; Pao, C.; Rudnicky, A. I.; and Shriberg, E. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *NAACL*.

Dhamdhere, K.; McCurley, K. S.; Nahmias, R.; Sundararajan, M.; and Yan, Q. 2017. Analyza: Exploring data with conversation. In *IUI*.

Dong, L., and Lapata, M. 2018. Coarse-to-Fine decoding for neural semantic parsing. In *ACL*.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*.

Hoque, E.; Setlur, V.; Tory, M.; and Dykeman, I. 2018. Applying pragmatics principles for interaction with visual analytics. *IEEE Transactions on Visualization and Computer Graphics*.

Huang, Z.; Xu, W.; and Yu, K. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*.

Iyer, S.; Konstas, I.; Cheung, A.; Krishnamurthy, J.; and Zettlemoyer, L. 2017. Learning a neural semantic parser from user feedback. In *ACL*.

Iyyer, M.; Yih, W.-t.; and Chang, M.-W. 2017. Search-based neural structured learning for sequential question answering. In *ACL*.

Jia, R., and Liang, P. 2016. Data recombination for neural semantic parsing. In *ACL*.

Kumar, V., and Joshi, S. 2016. Non-sentential question resolution using sequence to sequence learning. In *COLING*.

Kumar, V., and Joshi, S. 2017. Incomplete follow-up question resolution using retrieval based sequence to sequence learning. In *SIGIR*.

Lee, K.; He, L.; Lewis, M.; and Zettlemoyer, L. 2017. End-to-end neural coreference resolution. In *EMNLP*.

Long, R.; Pasupat, P.; and Liang, P. 2016. Simpler context-dependent logical forms via model projections. In *ACL*.

Miller, S.; Stallard, D.; Bobrow, R.; and Schwartz, R. 1996. A fully statistical approach to natural language interfaces. In *ACL*.

Palakurthi, A.; Ruthu, S. M.; Akula, A. R.; and Mamidi, R. 2015. Classification of attributes in a natural language query into different SQL clauses. In *RANLP*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

Pasupat, P., and Liang, P. 2015. Compositional semantic parsing on semi-structured tables. In *ACL-IJCNLP*.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Poon, H. 2013. Grounded unsupervised semantic parsing. In *ACL*.

Popescu, A.-M.; Armanasu, A.; Etzioni, O.; Ko, D.; and Yates, A. 2004. Modern natural language interfaces to databases: Composing statistical parsing with semantic tractability. In *COLING*.

Raghu, D.; Indurthi, S.; Ajmera, J.; and Joshi, S. 2015. A statistical approach for non-sentential utterance resolution for interactive QA system. In *SIGDIAL*.

Ramshaw, L. A., and Marcus, M. P. 1999. Text chunking using transformation-based learning. *ACL*.

Ren, G.; Malik, M.; Ni, X.; Ke, Q.; and Bhide, N. 2017. Conversational/multiturn question understanding. In *ICTIR*.

Sang, E. F. T. K. 2002. Introduction to the CoNLL-2002 shared task. In *COLING*.

Setlur, V.; Battersby, S. E.; Tory, M.; Gossweiler, R.; and Chang, A. X. 2016. Eviza: A natural language interface for visual analysis. In *UIST*.

Shah, P.; Hakkani-Tur, D.; Liu, B.; and Tur, G. 2018. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *HLT-NAACL*.

Sordoni, A.; Galley, M.; Auli, M.; Brockett, C.; Ji, Y.; Mitchell, M.; Nie, J.-Y.; Gao, J.; and Dolan, B. 2015. A neural network approach to context-sensitive generation of conversational responses. In *HLT-NAACL*.

Suhr, A.; Iyer, S.; and Artzi, Y. 2018. Learning to map context-dependent sentences to executable formal queries. In *NAACL*.

Zettlemoyer, L. S., and Collins, M. 2009. Learning context-dependent mappings from sentences to logical form. In *ACL-IJCNLP*.

Zhong, V.; Xiong, C.; and Socher, R. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *CoRR*.