Improving NL-to-Query Systems through Re-ranking of Semantic Hypothesis

Pius von Däniken[1] and **Jan Deriu**[1] and **Eneko Agirre**[2] and
Ursin Brunner[1] and **Mark Cieliebak**[1] and **Kurt Stockinger**[1]
[1] ZHAW Zurich University of Applied Sciences
{vode,deri,brrn,ciel,stog}@zhaw.ch
[2] HiTZ Center - Ixa, University of the Basque Country UPV/EHU
e.agirre@ehu.eus

Abstract

Natural Language-to-Query systems translate a natural language question into a formal query language such as SQL. Typically the translation results in a set of candidate query statements due to the ambiguity of natural language. Hence, an important aspect of NL-to-Query systems is to rank the query statements so that the most relevant query is ranked on top. We propose a novel approach to significantly *improve the query ranking and thus the accuracy of such systems*. First, we use existing methods to translate the natural language question (NL_{in}) into k query statements and rank them. Then we translate each of the k query statements back into a natural language question (NL_{gen}) and use the *semantic similarity* between the original question NL_{in} and each of the k generated questions NL_{gen} to *re-rank the output*. Our experiments on two standard datasets, OTTA and Spider, show that this technique improves even strong state-of-the-art NL-to-Query systems by up to 9 percentage points. A detailed error analysis shows that our method correctly down-ranks queries with missing relations and wrong query types. While this work is focused on NL-to-Query, our method could be applied to any other semantic parsing problems as long as a text generation method is available.

1 Introduction

NL-to-Query describes the task of translating natural language questions to meaningful representations, such as logical forms, executable code, or structured query languages like SQL. The application of neural networks and the introduction of larger datasets (Yin and Neubig, 2017; Yu et al., 2018; Brunner and Stockinger, 2021) has increased performance, but the task is far from solved.

Re-ranking of candidate query statements allows introducing additional information in the process (Yin and Neubig, 2019). For a given natural language question (NL_{in}), neural networks keep a beam search and produce k candidate query statements (QS). Our analysis shows that an oracle selecting the correct query among the top-scoring 15 candidates would improve the performance of publicly available systems by up to 10 accuracy points on the Spider benchmark (Yu et al., 2018).

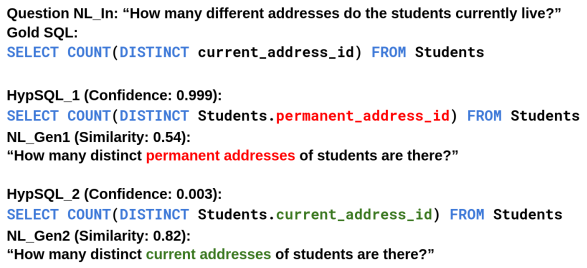


Figure 1: Example illustrates how semantic similarity is used to extract the correct hypothesis. `NL_In` is the input question, `Gold SQL` is the gold SQL query, `HypSQL_1` and `HypSQL_2` are generated by an NL-to-Query system (with confidence scores), and `NL_Gen1` and `NL_Gen2` are back-translated from the `HypSQL` statements, with scores by a similarity system. See text for further details.

Inspired by the success of back-translation in machine translation (Sennrich et al., 2016), we propose to *re-rank the candidate queries* according to the semantic similarity between the original question NL_{in} and the k synthetic questions NL_{gen} obtained via back-translating each of the k candidate queries into natural language. Figure 2 depicts the pipeline of our proposed system.

Figure 1 shows an example from the Spider dataset. For the question "How many different addresses do the students currently live?". The highest-ranked query according to the beam search ranking is HypSQL_1 with a confidence score of 0.999. However, this query returns the *permanent* addresses, which does not refer to the correct attribute, which would be the *current* addresses. In the example, the second hypothesis (i.e., HypSQL_2) has a much lower confidence of 0.003

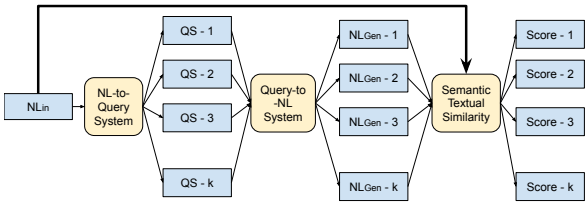


Figure 2: Pipeline of our system. NL_{in} = original natural language, QS = query statement, NL_{gen} = generated natural language.

although it fits the input question perfectly. On the other hand, the *semantic similarity* score between NL_{in} and the generated questions NL_{gen} shows a different picture: The back-translation of the correct hypothesis, i.e., NL_{gen2}, has a higher semantic similarity (0.82) than the back-translation of the incorrect hypothesis (0.54). Hence, semantic similarity would help to identify the correct query. This paper makes the following contributions:

- We present a novel method to improve NL-to-Query systems using re-ranking according to *Query-to-NL back-translation and semantic similarity*.
- We showcase improvements in two datasets using three systems, around $5-9$ points in OTTA (Deriu et al., 2020) and $2-3$ points in Spider (Yu et al., 2018).
- The error analysis shows that our method down-ranks hypotheses with missing relations or with incorrect query types.

2 Related Work

NL-to-Query (also referred to as Natural Language to Databases NLIDB) describes the task of translating natural language questions into structured queries (e.g., SQL). Most current approaches are based on sequence-to-sequence architectures (Yin and Neubig, 2017; Dong and Lapata, 2018; Suhr et al., 2018; Deriu et al., 2020), where the encoder is a recurrent neural network that generates a hidden representation of the natural language question, and the decoder is a recurrent neural network that generates the query. Alternatively, some approaches combine symbolic reasoning with information retrieval techniques (Sen et al., 2020). For a more in-depth treatment, we refer the reader to Affolter et al. (2019) and Odzcan et al. (2020).

In this work, we focus on the translation from natural language questions to database queries, where most recent approaches were proposed in the context of the text-to-SQL Spider dataset (Yu et al., 2018)[1]. Instead of working directly on SQL, some authors propose to use simpler and more general abstract syntax trees. For instance, Deriu et al. (2020) propose to use so-called Operation Trees, which we also used for this work.

Hypothesis Re-ranking is the task of creating an alternative ranking of k candidate solutions for a given task. The k candidates are usually the output of a beam search. In our case, the candidates are queries for the given natural language question. However, the problem of hypothesis re-ranking arises in many different generation tasks, not only NL-to-Query. For instance, Dušek and Jurcicek (2016) train a re-ranking network to score the generated hypotheses of their natural language generation model. Alternatively, (Deriu and Cieliebak, 2018; Agarwal et al., 2018) trained classifiers to predict the correctness of the hypotheses produced by their natural language generation system and select the hypothesis with the highest correctness score. Most of these approaches are developed in the field of natural language generation from structured data. For code generation, Yin and Neubig (2019) perform re-ranking by reconstructing the original utterance for the generated code. They use the reconstruction error as a measure for re-ranking. We are not aware of prior research on using textual semantic similarity to re-rank hypotheses in the field of NL-to-Query or Semantic Parsing in general.

Semantic Textual Similarity assesses to what degree two chunks of text are similar, usually on a 0-5 scale, which ranges from unrelated (0) to semantically equivalent (5) (Agirre et al., 2013). The advent of transformer-based models such as RoBERTa (Liu et al., 2019) has improved automatically assessing semantic textual similarity. Recently (Kane et al., 2020) introduced *NUBIA (NeUral Based Interchangeability Assessor for Text Generation)*. It extracts features from RoBERTa and GPT-2 (Radford et al., 2019) and fine-tunes a fully connected neural network to output a score between 0 and 1, indicating how interchangeable two input sentences are. Throughout this work, we will use *NUBIA* to automatically score the similarity between a natural question (NL_{in}) and a back-translated question (NL_{gen}).

Query-to-NL has the goal of translating a struc-

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

tured query into natural language and to provide a lay user with an explanation of the meaning of the query. A simple approach is to define production rules applied to the nodes of the abstract syntax tree (AST) of the query. Systems based on this idea have been developed for SQL (Koutrika et al., 2010), SPARQL (Ngonga Ngomo et al., 2013), Operation Trees (von Däniken, 2021), and queries expressed in lambda calculus (Wang et al., 2015). There are also systems based on neural networks such as (Xu et al., 2018). In this work, we leverage one of those systems to post-process the output of an NL-to-Query system. Others have also used query explanations to incorporate corrective feedback from the user in the NL-to-Query workflow (Elgohary et al., 2020; Labutov et al., 2018; Yao et al., 2019, 2020).

3 Method: Similarity for Re-ranking

The proposed method works in three steps (see also Figure 2): first, the NL-to-Query system translates the natural language input NL_{in} into a set of k candidate query statements QS - called our hypotheses. This is achieved by applying beam search during the decoding stage of a recurrent neural network. In the second step, each of the k hypotheses QS is translated back into natural language NL_{gen} using a Query-to-NL system. In the last step, each of the k back-translations NL_{gen} is compared to the original input using an off-the-shelf semantic textual similarity algorithm. We use the semantic similarity score to rank the hypotheses. For each NL_{in}, the top-scoring hypothesis is returned as the answer of the system.

3.1 Ranking Hypotheses based on Semantic Textual Similarity

Let NL_{in} be the user input (i.e., the natural language question) and $H = \{QS_1, ..., QS_k\}$ be the set of k hypotheses, i.e. candidate query statements QS_i, that are the output of the NL-to-Query system. In most cases, this set is the result of applying beam search for decoding. However, other approaches result in a set of hypotheses, for instance an ensemble of different NL-to-Query systems. In this work, we focus only on beam search-based hypothesis sets. Thus, each of the hypotheses has a confidence score c_i, which is used to rank the set of hypotheses, i.e., the candidate queries. We refer to this ranking as *Confidence*.

In a second step, each of the hypotheses QS_i is back-translated into a natural language question NL^i_{gen} using a Query-to-NL engine. Thus, we end up with a set of back-translated hypotheses $H_Q = \{NL^1_{gen}, ..., NL^k_{gen}\}$.

In a third step, we compute for each back-translated hypothesis the semantic textual similarity score with the user input NL_{in}, i.e., $s_i = \mathit{SemSim}(NL_{in}, NL^i_{gen})$. The set of hypotheses can be ranked according to the semantic similarity scores. We refer to this ranking as *Semantic*.

3.2 Weighting Strategies

Since the two rankings, *Confidence* and *Semantic* may disagree on the top hypothesis in some cases (as we have shown in the example in Figure 1), we combine the two scores c_i and s_i into a new ranking. For this, we propose the following weighting strategies:

Equal Weighting. The naive strategy is based on simply multiplying the two scores, that is $m_i^{equal} = c_i * s_i$, and we rank the set of hypotheses according to m_i^{equal}. We refer to this ranking as *Equal Weighting*.

Calibrated Weighting. Since the confidence scores and the semantic similarity scores have different distributions, the influence of each score in the *Equal Weighting* is not equal. For instance, in some cases, the influence of c_i is stronger than s_i and vice-versa. To counteract this effect, we decided to *calibrate both scores before multiplying*. A calibrated score should reflect the proportion of correct hypotheses selected, e.g., when a calibrated system assigns a score of 0.8 to a hypothesis, this hypothesis will be correct in 80% of the cases.

We use *Platt Scaling* (Platt, 2000) to calibrate both scores. This works by training a logistic regression model on the outputs of a model to transform these outputs into probability distributions. More precisely, for the confidence scores and the semantic scores respectively, a logistic regression model is trained. For this, we have to set aside a few hypotheses (more details later on). For the confidence calibration, a logistic regression model is trained on a set of pairs of confidence score and a label that indicates if the query is correct, i.e., $\mathbf{D} = \{(c_i, I^i_{\text{corr}}\}$. Analogously, we train a logistic regression model for the semantic similarity score s_i. Thus, the calibrated scores can be interpreted as the probability of the query being correct, i.e., $c_i^{calib} = Pr(I^i_{\text{corr}} = 1|c_i)$ and $s_i^{calib} = Pr(I^i_{\text{corr}} = 1|s_i)$. We call the score after

calibration c_i^{calib} and s_i^{calib} and the resulting mixed score $m_i^{calib} = c_i^{calib} * s_i^{calib}$. The resulting ranking is called *Calibrated Weighting*.

Learned Weighting. A natural extension of the calibration idea is to *train a logistic regression on both scores* at the same time, instead of independently. That is, we train a logistic regression model on pairs of confidence and semantic-scores[2], i.e., $\mathbf{D} = \{((c_i, s_i), I^i_{\text{corr}}\}$. This way, the model can learn the mixed proportions directly. Thus, $m_i^{learned} = Pr(I^i_{\text{corr}} = 1|c_i, s_i)$. For this, we again have to set aside a few hypotheses. We use the predicted probabilities from the logistic regression model to rank hypotheses and call the resulting ranking *Learned Weighting*.

Threshold Weighting. We observed that the confidence scores c_i are high in most cases in which the *Confidence* ranking yields a correct query. In many cases where the *Confidence* ranking yields wrong queries, the confidence scores are low. However, the *Semantic* scores tend to be higher. Thus, we propose the following strategy: If the maximum confidence score of the hypotheses set is above a threshold, we use the *Confidence* ranking, otherwise, we use the *Semantic* ranking. We refer to this ranking as *Threshold Weighting*. The threshold is calculated by first determining the 90th percentile over the confidence scores of all training hypotheses and then finding the lowest confidence of a correct hypothesis that lies above that.

Upper Bounds. To determine the theoretical upper bounds of our approach, we introduce two oracles. The first oracle selects the correct hypothesis from the candidates if there is one. The second oracle selects the correct hypothesis between the two top-ranked hypotheses by *Confidence* or *Semantic* if there is one. The first oracle determines the potential of re-ranking in general (we refer to it as *Oracle*). The second oracle determines the maximum contribution that the semantic similarity could do to Confidence (we refer to it as *Oracle-Sem*).

4 Experimental Setup

In this section, we describe the experimental setup, the datasets, the NL-to-Query models, the Query-to-NL model, and the semantic textual similarity model.

[2]Using more features, e.g., the length of the generated query or m_i^{equal} did not yield any improvements.

4.1 Datasets

We analyzed our approach on two different datasets used as benchmarks for evaluating NL-to-Query systems: Spider (Yu et al., 2018) and OTTA (Deriu et al., 2020). Both datasets contain complex queries and cover large amounts of attributes of the databases. Spider contains around 10K queries against 200 different databases. The dataset is used to study NL-to-SQL translations. OTTA contains around 3.8K queries over 5 databases. OTTA is used to study translations from NL-to-OT (Operation Trees) which are similar to abstract syntax trees (AST), i.e., an intermediate query language can be translated to other query languages such as SQL or SPARQL. OTTA contains more complex queries with longer join paths than Spider. From the OTTA corpus, we used only queries against the databases *Moviedata* and *Chinook* since they contain the largest amounts of queries. Details about the queries used for each dataset are given below.

4.2 NL-to-Query Models

We applied publicly available machine learning models trained for the datasets, which produce queries with filter values in the WHERE-clauses as otherwise there would be placeholder tokens in the back-translations. For all models, we use a beam size[3] of $k = 15$. For the OTTA corpus, we used the pre-trained *GrammarNet* by (Deriu et al., 2020). The output of *GrammarNet* is a set of Operation Tree (OT) hypotheses, which represent the query. OTs can be translated to SQL and executed on an SQL database. For each of the two domains in OTTA (i.e., *Moviedata* and *Chinook*), we use a specifically trained *GrammarNet*. We refer to these models as *GrammarNet-Moviedata* and *GrammarNet-Chinook*. For the Spider dataset, we apply two strong NL-to-SQL systems that are publicly available. The first system is *BridgeV2* (Lin et al., 2020), which returns a set of hypothesis SQL queries from a beam search decoder. We refer to this model as *Spider-BridgeV2* [4]. The second system is *ValueNet* by Brunner and Stockinger (2021), which also returns a set of SQL hypotheses from a beam search decoder[5]. We refer to this model as

[3]In preliminary experiments, we noted that using a larger beam size does not impact the scores significantly.

[4]We chose these systems for their strong performance, code availability and quality of code.

[5]The API provided by the authors included confidence scores based on the sum per-token-confidence instead of average. We approximated the average by dividing the provided score by the number of characters in the SQL hypothesis.

Spider-ValueNet.

4.3 Query-to-NL Model

For back-translating queries to natural language, we use the *Operation Tree-to-Text (OT3)* system kindly made available by von Däniken (2021). It translates OTs into natural language questions in a rule-based manner, which ensures that most OTs are translated correctly, i.e., no nodes are left out or added during translation. OT3 is domain-agnostic, which allows it to be easily adapted to a new domain by just defining domain-specific metadata, i.e., the canonical names of the tables, attributes and types. The main advantage of OT3 is the ability to express relationships naturally, which results in more fluent back-translations. There are currently some limitations with the state-of-the-art Query-to-NL models, which do not handle more complex constructs [6] well. Thus, we perform the evaluation only on the queries that are handled by OT3. More details can be found in Appendix A.

4.4 Semantic Textual Similarity Model

In order to compute textual semantic similarity between two questions, i.e., between NL_{in} and NL_{gen}, we apply NUBIA (Kane et al., 2020), a pre-trained model that scores a pair of sentences based on their interchangeability. We use NUBIA[7] out-of-the-box without any fine-tuning.

4.5 Mixed Strategies

For the *Calibrated Weighting*, the *Learned Weighting*, and the *Threshold Weighting* rankings, labeled data points are needed for setting up the mixed strategy. The samples are used to train the logistic regression models for the *Calibrated Weighting* and the *Learned Weighting*. We use the implementation provided by *scikit-learn* (Pedregosa et al., 2011) with balanced class weights and all other parameters as default. For the *Threshold Weighting*, these samples are used to determine the threshold for when to select the *Confidence* ranking or the *Semantic* ranking. We use k-fold cross-validation with k[8] chosen such that there are 20 samples in each fold[9] for training the strategies for each split. We report accuracies averaged over the k test splits for all strategies.

[6] E.g., GroupBy, SetOperations, or Nested Quieries

[7] https://github.com/wl-research/nubia

[8] Concretely, $k = 22$ for Spider, $k = 11$ for Moviedata, and $k = 12$ for Chinook.

[9] This results in $20 * 15 = 300$ data points for training the logistic regression models.

5 Results

As explained in the previous section, we evaluate the effectiveness of our approach over two different datasets consisting of 22 databases using three different systems, as shown in Table 1. We evaluate the systems using the *component equality* proposed by Yu et al. (2018). We can see that for all datasets one of our *re-ranking approaches outperforms the baseline* without re-ranking up to 9%. We will now analyze our re-ranking approaches in more detail.

Semantic Re-ranking. In all cases, except for *Chinook*, the *Semantic-based* re-ranking performs worse than the baseline system ranking (Confidence), showing that our method alone has not enough information to select the correct hypothesis.

Mixed Re-ranking (i.e. Equal, Calibrated, Learned, Threshold). On the contrary, the combination of the Confidence and Similarity scores improves over *Confidence* alone in all mixed strategies (with a minor exception for Threshold for ValueNet in Spider). The improvement ranges from $2-3\%$ on *Spider* to $5-9\%$ on *OTTA*. These results show that our method injects new information and improves over the base systems. In all cases, the simple *Equal Weighting* performs well, making it a great default mixed strategy. The results or other mixed strategies are better in some cases, although the best mixed strategy varies in each column. For instance, for *Spider-Bridge* the *Threshold Weighting* strategy works best, yielding an improvement of 2.56 points in accuracy.

Oracle. The difference between *Confidence* and *Oracle*, i.e. the optimal re-ranking, lies at around 18% for both *OTTA* subcorpus and $8-10\%$ for Spider, depending on the system. The differences in margins between *Spider* and *OTTA* can be explained by the fact that the *Spider*-based models achieve higher *Confidence* accuracies, which decreases the margin for improvement.

Oracle-Sem. The difference between the best mixed strategy and *Oracle-Sem* is around 3 points. Thus, there is a potential improvement of around 3 points left for all systems using semantic similarity. However, the difference between the *Oracle-Sem* score and the *Oracle* score differs between the *Spider*-based systems and the *OTTA*-based systems. While the difference in the *Spider*-based systems is between 3 to 4 points, the difference for the *OTTA*-based systems is between 6 to 7 points.

Dataset System	OTTA-Chin. GrammarNet	OTTA-Movie GrammarNet	Spider Bridge	Spider ValueNet
Confidence	42.89	52.24	71.46	74.31
Semantic	48.16 (+5.27)	45.25 (−6.99)	62.70 (−8.76)	68.01 (−6.30)
Equal	**51.84** (+8.95)	59.23 (+6.99)	73.03 (+1.57)	76.83 (+2.52)
Calibrated	51.44 (+8.55)	**59.60** (+7.36)	73.78 (+2.32)	**77.22** (+2.91)
Learned	51.48 (+8.59)	59.44 (+7.20)	73.93 (+2.47)	77.09 (+2.78)
Threshold	46.90 (+4.01)	54.71 (+2.47)	**74.05** (+2.59)	71.30 (−3.01)
Oracle-Sem	54.94 (+12.05)	62.38 (+10.14)	77.30 (+5.84)	80.35 (+6.04)
Oracle	61.32 (+18.43)	69.98 (+17.74)	81.12 (+9.66)	83.12 (+8.81)

Table 1: Accuracy of our approach for translating NL questions to OTs and SQL, respectively, using three different systems and two different datasets. The deltas with respect to the *Confidence* ranking (baseline) are shown in parentheses. *Oracle-Sem* and *Oracle* are theoretical upper bounds.

6 Discussion

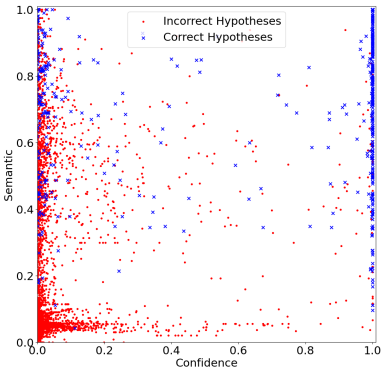
Based on the results, we see that including semantic similarity for re-ranking works better than using the *Confidence* scoring only. In this section, we explore the potential and limitations of this approach in more detail.

6.1 Confidence Score vs. Semantic Similarity Score

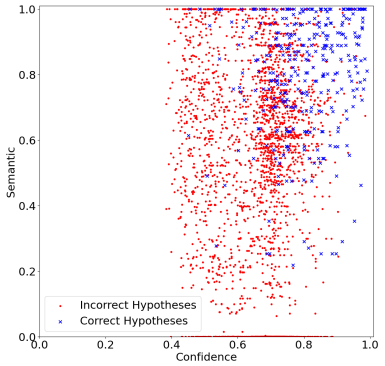
To better understand the results, we analyze the relationship between the confidence scores and the semantic scores. In Figure 3, the confidence scores are plotted against the semantic similarity scores, where blue dots denote correct hypotheses, and red dots denote incorrect ones. We perform the analysis on the *Bridge* system over Spider and the GrammarNet system over *Moviedata*, as they show the clearest difference in score distributions.

First, we note that the distributions for the two systems look different. For *Bridge* the confidence scores mostly lie at the edges, either at 0.0 or 1.0. The *Moviedata* confidence scores are more evenly distributed between 0.4 and 1.0. On the other hand, the semantic similarity scores are evenly distributed in both cases.

Second, we note that for the *Bridge* system, confidence scores close to 1.0 are reliable, i.e., a hypothesis with confidence close to 1.0 tends to be correct. On the other hand, correct hypotheses with low confidence tend to have higher semantic scores (see upper left corner). This explains the strong performance of *Threshold Weighting* for *Bridge*. For *Moviedata*, the picture is different. The correct samples tend to have both high confidence and high semantic scores (upper right corner). Thus, the other weighing strategies tend to perform well, while *Threshold Weighting* under-performs.



(a) Spider-BridgeV2



(b) Moviedata-GrammarNet

Figure 3: Confidence scores and semantic similarity scores for hypotheses produced by *Spider-BridgeV2* and *Moviedata-GrammarNet*. Every cross corresponds to a hypothesis. Blue indicates correct hypotheses and red incorrect ones.

Third, we note that semantic scoring alone is not sufficient. For *Bridge*, the semantic score tends to score correct hypotheses as low as the incorrect ones (see lower part). However, it works well for finding incorrect hypotheses. Although the distributions for *Bridge* and *Moviedata* have great differences, our approach works in both cases.

Error Type	Missing Join			
Original Question	List all singer names in concerts in year 2014.			
Ranking	**SQL**	**Back-translated Question**	c_i	s_i
Gold	SELECT T2.name FROM singer_in_concert AS T1 JOIN singer AS T2 ON T1.singer_id = T2.singer_id JOIN concert AS T3 ON T1.concert_id = T3.concert_id WHERE T3.year = 2014	What are the names of singers who performed in concerts whose year is 2014?	-	-
Baseline	SELECT singer.Name FROM singer_in_concert JOIN singer ON singer_in_concert.Singer_ID = singer.Singer_ID WHERE singer.Song_release_year = 2014 (**missing table "concert"**)	What are the names of singers who were released in 2014 who performed in concerts?	0.020	0.792
Semantic	SELECT singer.Name FROM singer_in_concert JOIN singer ON singer_in_concert.Singer_ID = singer.Singer_ID JOIN concert ON singer_in_concert.concert_ID = concert.concert_ID WHERE concert.Year = 2014	What are the names of singers who performed in concerts whose year is 2014?	0.015	0.823

Error Type	Wrongly added Filter			
Original Question	Find the pixel aspect ratio and nation of the tv channels that do not use English.			
Ranking	**SQL**	**Back-translated Question**	c_i	s_i
Gold	SELECT Pixel_aspect_ratio_PAR , country FROM tv_channel WHERE LANGUAGE $\neq$ 'English'	What are the aspect ratios and countries of tv channels whose language is not English?	-	-
Baseline	SELECT TV_Channel.Pixel_aspect_ratio_PAR, TV_Channel.Country FROM TV_Channel WHERE TV_Channel.Language $\neq$ "English"	What are the aspect ratios and countries of tv channels whose language is not english?	1.000	0.654
Semantic	SELECT TV_Channel.Pixel_aspect_ratio_PAR, TV_Channel.Country FROM TV_Channel WHERE TV_Channel.Language $\neq$ "English" AND TV_Channel.Country $\neq$ "English" (**wrong additional filter**)	What are the aspect ratios and countries of tv channels whose country is not english and whose language is not english?	0.008	0.673

Table 2: Examples of types of errors due to re-ranking. For each error type, we show the natural language question and the corresponding SQL gold standard. Next we show the top candidates according to the *Confidence* ranking and the *Semantic* ranking. c_i and s_i refer to confidence score of the NL-to-query translation and the similarity score between the natural language questions, respectively.

NL_{in}:	Whats the average track size of tracks purchased from 120 S Orange Ave?				
i	NL_{gen}	c_i	s_i	m_i^{equal}	OK
1	What is the average size of all tracks on invoice lines which are part of invoices?	0.669	0.49	0.327	F
2	What is the average size of all tracks on invoice lines which are part of invoices whose billing street is 120 S Orange Ave?	0.668	0.61	0.407	T
9	What is the average size of all tracks on Albums on invoice lines which are part of invoices whose billing street is 120 S Orange Ave?	0.632	0.3	0.1896	F
NL_{in}:	Which companies from Mexico produced their films in Mexico ?				
i	NL_{gen}	c_i	s_i	m_i^{equal}	OK
1	What are the names of companies which produced movies whose status is Mexico?	0.729	0.676	0.492	F
3	What are the names of companies which produced movies which were produced in countries whose name is Mexico?	0.712	0.751	0.534	T
5	What are the names of companies which produced movies whose name is Mexico?	0.664	0.741	0.492	F
NL_{in}:	What are the distinct template type descriptions for the templates ever used by any document?				
i	NL_{gen}	c_i	s_i	m_i^{equal}	OK
1	What are the distinct descriptions of template types for templates?	0.494	0.686	0.338	F
2	What are the distinct descriptions of template types for templates used for documents?	0.091	0.973	0.166	T
3	Show me everything about template types.	0.031	0.133	0.050	F

Table 3: Illustrative examples of the impact of re-ranking. We show three original questions (NL_{in}) and the corresponding back-translated examples (NL_{gen}). Value i denotes the rank in the *Confidence* ranking, c_i is the confidence score of the decoder, s_i is the similarity score, m_i^{equal} is the combination of c_i and s_i, OK indicates whether the generated query is correct (T = true, F = false).

6.2 Error Analysis: Confidence vs. Semantic Ranking

To better understand the differences between the *Semantic* and *Confidence* rankings, we analyze the cases in which one of the two ranking schemes returns a correct query, and the other one does not. This analysis is performed on the *Bridge* output where in 19.2% of the cases, only one of the two ranking schemes returns the correct hypothesis. In 25% of the cases in which only the *Confidence*

ranking returns a correct query, the *Semantic* ranking returned a query with a redundant *WHERE*-clause, and in 20% of cases, the *Semantic* ranking returned a wrong attribute in the projection. This suggests that the *Semantic* ranking is not stable against redundant information in the query and slight variations in the return attributes.

In the cases where only the *Semantic* ranking returns a correct query, the query returned by the *Confidence* ranking contains missing or redundant *Join*-clauses in 47% of cases and wrong query types in 21% of cases. This suggests that the *Semantic* ranking's strength lies in detecting missing relations and detecting wrong query types (i.e., SUM instead of COUNT).

In Table 2 two examples of errors are shown. The first example shows a missing join operation of *Confidence*. In particular, the table "concert" is missing in the SQL statement. In this case the confidence score of the wrong *Confidence* query, i.e. $c_i = 0.02$, is higher than the confidence of the correct *Semantic* query, i.e. 0.015. On the other hand, the semantic textual similarity score s_i of the correct *Semantic* query, i.e., 0.823, is higher than the score of the incorrect *Confidence* query, i.e., 0.792. We note that although the confidence score of the incorrect query is the highest of all hypotheses, it is a low score. Usually, the confidence scores are around 1.0.

The second example shows the problem of an additional filter (TV_Channel.Country $\neq$ "English"), which confuses the semantic similarity score. The *Confidence* ranking selects the correct query with high confidence, i.e. 1.0. However, the semantic score of the incorrect *Semantic* query, i.e., 0.673, is higher than the semantic score of the correct query, i.e., 0.654.

This phenomenon motivates the *Threshold Weighting*. The reason is that high confidence scores from the NL-to-Query system are more trustworthy than the semantic scores. However, in cases where the NL-to-Query system is not confident, the semantic score performs well. The automatically determined threshold in our experiments lies at around 0.9.

6.3 Qualitative Analysis

In Table 3, we show examples of the different rankings. We show three representative examples of a 15-best list. In the first example, we note that the hypothesis with the best confidence score, i.e., c_1 = 0.669, is incorrect. The second best hypothesis, according to the confidence score, is correct and has a very similar score to the hypothesis placed first (0.669 *vs.* 0.668). The hypothesis that is placed 9^{th} adds an unnecessary relation. However, the confidence score is still close to the hypothesis placed first. The semantic score, on the other hand, is more accurate. The correct hypothesis is placed 1^{st} with a large margin (0.61 *vs.* 0.49) and an even larger difference with the score of the 9^{th} place. Finally, the combined score m_2^{equal} of 0.407 clearly identifies result 2 as the correct one.

The second example shows a similar pattern: the first hypothesis with a confidence score c_1 of 0.729 is obviously wrong. The second hypothesis, which is correct, has a slightly lower confidence score c_2 of 0.712. The *Semantic* score s_3 of 0.751 ranks the set of hypotheses correctly. However, *Semantic* re-ranking alone is not enough since the 5^{th} ranked example has a very high semantic similarity score while being incorrect. In this case the *Equal Weighing* approach m_i^{equal} helps differentiating: While s_3 and s_5 are very close, m_3^{equal} and m_5^{equal} have a bigger margin.

The last example shows a case where the *Equal Weighting* does not work. Although the semantic score s_2 of 0.933 works to find the correct answer, the confidence score c_2 of 0.091 of the correct hypothesis is much lower than the confidence of the incorrect hypothesis, $c1$ of 0.494. In this case, the *Threshold Weighting* would work well as it relies on s_i for the cases where the maximum confidence score is too low.

7 Conclusion

We proposed a novel approach to improve semantic NL-to-Query systems based on back-translating the generated query into a natural language question, and re-ranking the top hypothesis of the NL-to-Query system according to the semantic similarity of the generated questions with regard to the original question. Our approach improves over strong, publicly available systems by up to 3 percentage points on the Spider dataset and up to 9 points on the OTTA dataset.

Our results clearly show the potential of back-translation for improving NL-to-Query systems, and it could be applied to more general semantic parsing problems as long as a generation method is available.

Acknowledgements

This work was supported by the European Union's Horizon 2020 research and innovation program under grant agreement No. 863410.

References

Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal*, 28(5):793–819.

Shubham Agarwal, Marc Dymetman, and Eric Gaussier. 2018. Char2char generation with reranking for the e2e nlg challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 451–456.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Angela Bonifati, Wim Martens, and Thomas Timm. 2017. An analytical study of large SPARQL query logs. *Proc. VLDB Endow.*, 11(2):149–161.

Ursin Brunner and Kurt Stockinger. 2021. Valuenet: A natural language-to-sql system that learns from database information. *International Conference on Data Engineering (ICDE)*.

Jan Deriu, Katsiaryna Mlynchyk, Philippe Schläpfer, Alvaro Rodrigo, Dirk von Grünigen, Nicolas Kaiser, Kurt Stockinger, Eneko Agirre, and Mark Cieliebak. 2020. A methodology for creating question answering corpora using inverse data annotation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 897–911, Online. Association for Computational Linguistics.

Jan Milan Deriu and Mark Cieliebak. 2018. Syntactic manipulation for generating more diverse and interesting texts. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 22–34.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Ondřej Dušek and Filip Jurcicek. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51.

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 58th Annual Meeting of the Association for Computational Linguistics*, pages 2065–2077, Online. Association for Computational Linguistics.

Hassan Kane, Muhammed Yusuf Kocyigit, Ali Abdalla, Pelkins Ajanoh, and Mohamed Coulibali. 2020. Nubia: Neural based interchangeability assessor for text generation.

Andreas Kokkalis, Panagiotis Vagenas, Alexandros Zervakis, Alkis Simitsis, Georgia Koutrika, and Yannis Ioannidis. 2012. Logos: A system for translating queries into narratives. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, page 673–676, New York, NY, USA. Association for Computing Machinery.

G. Koutrika, A. Simitsis, and Y. E. Ioannidis. 2010. Explaining structured queries in natural language. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 333–344.

Igor Labutov, Bishan Yang, and Tom Mitchell. 2018. Learning to learn semantic parsers from natural language supervision. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1676–1690, Brussels, Belgium. Association for Computational Linguistics.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP 2020, November 16-20, 2020*.

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

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak sparql: Translating sparql queries into natural language. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, page 977–988, New York, NY, USA. Association for Computing Machinery.

Fatma Odzcan, Abdul Quamar, Jaydeep Sen, Chuan Lei, and Vasilis Efthymiou. 2020. State of the art and open challenges in natural language interfaces to data. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, page 2629–2636, New York, NY, USA. Association for Computing Machinery.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer,

R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

John Platt. 2000. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv. Large Margin Classif.*, 10.

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

Jaydeep Sen, Chuan Lei, Abdul Quamar, Fatma Özcan, Vasilis Efthymiou, Ayushi Dalmia, Greg Stager, Ashish Mittal, Diptikalyan Saha, and Karthik Sankaranarayanan. 2020. Athena++: Natural language querying for complex nested sql queries. *Proc. VLDB Endow.*, 13(12):2747–2759.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2238–2249, New Orleans, Louisiana. Association for Computational Linguistics.

Pius von Däniken. 2021. Improving a semantic parser through user interaction. *Publikationen School of Engineering*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342, Beijing, China. Association for Computational Linguistics.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. SQL-to-text generation with graph-to-sequence model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 931–936, Brussels, Belgium. Association for Computational Linguistics.

Ziyu Yao, Yu Su, Huan Sun, and Wen-tau Yih. 2019. Model-based interactive semantic parsing: A unified framework and a text-to-SQL case study. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5447–5458, Hong Kong, China. Association for Computational Linguistics.

Ziyu Yao, Yiqi Tang, Wen-tau Yih, Huan Sun, and Yu Su. 2020. An imitation game for learning semantic parsers from user interaction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6883–6902, Online. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450, Vancouver, Canada. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2019. Reranking for neural semantic parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4553–4559, Florence, Italy. Association for Computational Linguistics.

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. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

A On Query-to-NL

While Query-to-Text is not a contribution of our work, we discuss and motivate our choice of OT3 as our Query-to-Text engine. We adapted OT3 to handle all the domains in the Spider development set, which comprises 20 databases. In order to handle SQL queries, we translate SQL queries into OTs using a rule-based approach. The main advantage over statistical methods is that we can be sure that the queries are correctly back-translated to text. This is due to the rule-based nature of OT3.

Sanity Check. In order to show that OT3 correctly renders the semantics of a query, we first perform a sanity check, where we backtranslated the gold-standard tree for a given question. Thus, we need to show that the original question and the back-translation are semantically equivalent. As negative examples, we also mix in randomly sampled human questions, thus the original question and the negative back-translation should never be semantically equivalent. We let humans annotate this data, that is, we showed humans pairs of original questions and either a positive or negative back-translation. In this setting, humans agree in 94% of cases with the parsing ground-truth. This shows that the synthetic questions are understandable and

generally maintain the semantics of the underlying OT. The experiments show that the synthetic questions are of high quality and can be used as basis for re-ranking.

Limitations. OT3 does not handle *GROUP BY*, *sub-queries* and *set operations*, thus, we discard these samples from the Spider and OTTA development sets, keeping 82% of *OTTA-Moviedata*, 76% of *OTTA-Chinook* and 43% of *Spider*. The reported results are on these subsets of the datasets. Note that several studies on natural language query logs (Bonifati et al., 2017; Affolter et al., 2019) show that typical queries in real-world applications are far less complex than the ones contained in the Spider dataset. Hence, not supporting GROUP BYs, sub queries or set queries is not a significant issue in a real-world scenario. Note that our method can still be applied to the full datasets, defaulting to the Confidence ranking when none of the hypotheses could be back-translated. The positive results are consistent, but the improvement is lower, correlated with the coverage. E.g. an overall improvement of 0.67% for the whole Spider (with the Bridge system using equal mixed re-ranking), which roughly corresponds to the 1.57% improvement obtained on the 43% subset of Spider which does not contain complex SQL operations.

Selection. The choice of OT3 is motivated by the fact that it renders relationships between entities naturally. For instance, the relationship between persons and movies, which is modelled via the cast table, is expressed as "Persons that play in movies". For instance, Logos (Kokkalis et al., 2012) expresses the same relationship as "Persons associated with movies", which is not natural and cannot be handled by our semantic textual similarity tool. We also evaluated statistical models, which suffer from hallucinations (i.e., adding text that is not semantically related to the query) and are generally unreliable. Thus, we are not aware of any Query-to-Text solution, that handles all types of queries (Group By, Set Operations, Nested Queries) such that the generated texts read naturally. Thus, OT3 has proved to be best suited for our task.

B On Evaluation

We adapted the *Component Equality* measure for operation trees (OTs) since we translate the SQL queries of the *Spider*-based systems to OTs. For OTs, this measure checks if the nodes of the predicted tree correspond to the nodes of the gold standard tree. This allows measuring query equality independently of the order of the nodes. Furthermore, we adapted this analysis also to measure if the *Join* attributes are rendered correctly. We decided against a result-based evaluation since it is impossible to reasonably evaluate queries that return an empty result set, often leading to over-estimating the quality of NL-to-Query systems. This happens often in cases where the result set is empty or for count questions. For Spider the databases are very small and do not contain much data, thus, queries tend to return empty results. For OTTA, which uses Yes/No questions, this problem is even more pronounced. Thus, the result-based evaluation is not reliable, and we opted for the component-based evaluation, which is now the standard evaluation for the Spider dataset.