Enhancing Semantic Parsing Accuracy through Diverse Generation and Uncertainty Quantification: A Dual-Component Approach

Alex Najera, Vishnu Yarabarla

***Abstract*— This report is a study in the field of Machine Learning and Natural Language Processing, which focuses on the application of Weak Supervision and Uncertainty Quantification techniques to improve semantic parsing.**

Existing semantic parsers often struggle with the ambiguity and complexity inherent in natural language instructions or LLM outputs. This research proposes a semantic parsing architecture involving two complementary components to improve translation accuracy within task-oriented domains. Firstly, inspired by the AMA Research Paper, we will generate diverse semantic parsings and apply weak supervision to identify the most likely correct translation. Secondly, motivated by the ZeroTOP Research Paper, we will implement uncertainty quantification to trigger requests for user clarification when parser confidence is low.

We will begin this report with a brief overview of why semantic parsing is so important, and then segway into explaining the background topics which include the AMA paper which uses weak supervision and the ZeroTOP paper which talks about their rudimentary uncertainty threshold. Then, we will discuss how their ideas can be implemented and why they are so useful.

We will also talk about the Spider Dataset that we used which is a semantic parsing and text-to-SQL dataset. We used that dataset for comparison with our program when generating SQL from an English sentence.

Next, we discuss our different parsing approaches, including Stanza, which uses Stanford's CoreNLP, and its Dependency Parsing that is used to depict the relationship between the words in a sentence. We compare this with other leading parsing technologies, and our analysis demonstrates how each parsing strategy impacts the accuracy and efficiency of natural language understanding tasks, particularly in domain-specific applications like our SQL statements.

We will then discuss our conversion of Text-to-SQL which involves looking at the sentence and word relationships to generate valid SQL queries. From these queries, we apply Weak Supervision and Uncertainty Quantification to enhance our parsed results in our action space.

Our study explores the potential of applying machine learning methods on a natural language parsing task to enhance the final parse. Although we found a small increase in performance, we think this is a great start to improving parsing that could be greatly improved with more resources and time. Our paper establishes a strong foundation for future work, demonstrating that integrating our two special components would be a promising avenue for improving parsing accuracy.

Keywords—Weak Supervision, Uncertainty Quantification, Semantic Parsing

Introduction and Background Work

Semantic parsing is essential for bridging the gap between natural language and executable actions in task-based systems. However, traditional parsers often fail when dealing with complex language or limited context. This is a major bottleneck because even if LLMs advance to achieve near-perfect complex outputs, their practical use will be hindered by the inability to precisely parse them into executable actions. This proposal addresses this challenge by integrating two distinct modules within the parsing process, aiming to boost accuracy and robustness.

Recent research has explored the use of in-context learning or actively requesting feedback in order to improve LLM performance. These techniques include LLM-based example generation for prompt generation (weak supervision) and uncertainty quantification to enhance reliability. We will adapt and integrate these powerful approaches within a semantic parser architecture.

Weak Supervision

Weak supervision is an approach within the field of machine learning that aims to build predictive models using less reliable, less accurate, or "noisier" labels compared to what is typically used in conventional supervised learning. This approach is particularly valuable in scenarios where obtaining a large set of accurately labeled data is too costly, time-consuming, or otherwise impractical. Weak supervision leverages various forms of weaker, imperfect, or indirect sources of supervision to train models effectively, often integrating multiple weakly supervised sources to improve learning accuracy and robustness. Weak supervision encompasses several methods where the training labels are not obtained through direct or accurate annotations but are instead derived from heuristics, noisy labels, crowd-sourced data, or other indirect sources. These sources include Heuristic Rules, Distant Supervision, Crowd-Sourcing, Data Augmentation. We chose to follow the Heuristic Rules approach. Heuristic rules are simple, domain-specific rules used to label data automatically. These rules are typically quick to implement but may not cover all cases and can introduce biases. However, we made sure to be very specific with our heuristics to cover our cases and reduce how much bias we got.

RELATED WORK

AMA Paper

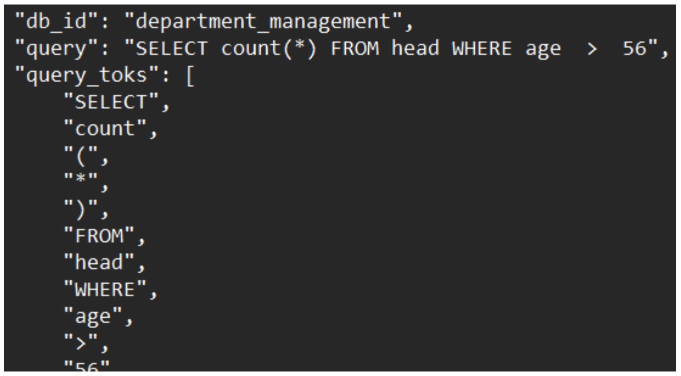
In the realm of semantic parsing, the AMA Paper introduces an approach using weak supervision to significantly enhance prompting accuracy. By using varied in-context demonstrations, the AMA approach has demonstrated a notable performance improvement of 10.2% in prompting accuracy. This methodology is particularly pertinent to our research as it provides a framework for generating diverse semantic parsings, which we adapt to improve translation accuracy within task-oriented parsing. It was also shown that specifically for natural language understanding, the AMA approach showed improvement in results showing promise to effectively improve parsers. The success of the AMA model in employing weak supervision serves as a foundational component of our proposed architecture.

ZeroTOP Paper

ZeroTOP introduces a zero-shot, task-oriented parsing approach that decomposes semantic parsing challenges into discrete question-answering components allowing for the LLM used to construct its target and observe its confidence per question. This method not only facilitates detailed understanding but also allows the system to assess its own confidence across various sub-tasks. By implementing a mechanism to identify unanswerable questions, ZeroTOP allows the model's ability to request further information instead of making inaccurate assumptions. Our research builds on this concept by incorporating uncertainty quantification to trigger user clarifications, thereby addressing one of the fundamental limitations identified in current semantic parsers.

DATASET

For our dataset, we used the Spider Dataset which is a cross-domain semantic parsing and text-to-SQL dataset. Spider includes complex SQL queries featuring multiple SQL components and clauses such as JOIN, GROUP BY, ORDER BY, and nested queries. It includes over 10,000 SQL queries with around 7,000 being used as training examples. Its SQL queries are more complex than almost all of the other datasets we looked into and includes highly relevant queries that we would use in a real world setting. This complexity comes as a Its cross-domain capabilities are perfect for our use since we want our parser to deal with a variety of english words and sentences to show the parser’s capabilities in identifying important information across various domains all the way from real estate to business management. It serves as a benchmark for assessing the performance of SQL generation, which we plan to do later in the paper. Many state-of-the-art models in semantic parsing report their results on Spider, making it a standard dataset for comparison. Researchers use Spider to develop models that can understand and transform natural language queries into SQL statements, which are then executable on a SQL database. This task requires both understanding the intent of the query and the structure of multiple databases. We informally used it in a simpler way to verify our generated SQL statements after running it through different parsers and ignored the database aspect of the dataset as it did not fit the scope of our project for our timeline.



```
"db_id": "department_management",
"query": "SELECT count(*) FROM head WHERE age  >  56",
"query_toks": [
    "SELECT",
    "count",
    "(",
    "*",
    ")",
    "FROM",
    "head",
    "WHERE",
    "age",
    ">",
    "56"
```

Fig. 1. Example Spider Dataset Entry

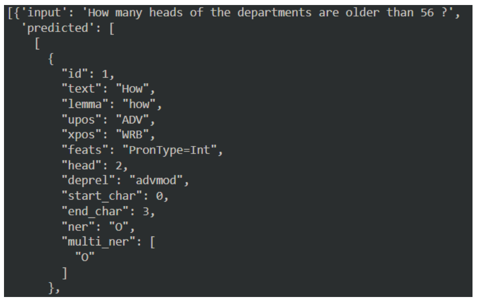
APPROACH

Our approach was to first parse our data through different parsers. From the parses, we will then convert the English sentence to an SQL statement. Since we have multiple parsers, we will turn each into an SQL statement and then run them through the weak supervisor in order to find the best SQL statement. The best option is rewarded which will give us the best option. If the uncertainty threshold is not met, then the user is prompted to enter more information to prevent uncertainty. This, in theory, should enhance our semantic parsing accuracy in this particular action space.

IMPLEMENTATION

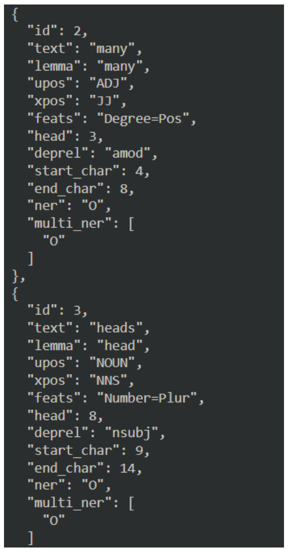
Parsing

To parse our data, we used three different types of parsers (Stanza, spaCy, and AllenNLP) in order to compare their performance. Stanza is Stanford’s natural language processing library which offers robust tokenization, part-of-speech tagging, lemmatization, dependency parsing, named entity recognition (NER), and more. It was built on PyTorch and is known for its accuracy and efficiency. Stanza's models are pre-trained on large corpora and are available in several languages, making it suitable for multilingual NLP tasks. However, we only need it to work in English right now. spaCy is similar, but is known for its speed, scalability, and extensive documentation. AllenNLP is our last parser which was specifically designed for natural language processing tasks. Similarly to Stanza, it is built on PyTorch but is well-known for its modularity and extensibility which allows for us to experiment with different architectures and techniques more easily. We chose these three as they are the most established parsers. Originally, we wanted to create our own parser but found that to be incredibly inefficient and impractical, especially considering that the established mainstream parsers are way ahead of anything else right now and would suffice for our purpose. The only way these parsers could be improved is with action space specific parsings. We had a mixed bag of results from these parsings. Here is an example of how Stanza parses a sentence:



```
[{'input': 'How many heads of the departments are older than 56 ?',
  'predicted': [
    [
      {
        "id": 1,
        "text": "How",
        "lemma": "how",
        "upos": "ADV",
        "xpos": "WRB",
        "feats": "PronType=Int",
        "head": 2,
        "deprel": "advmod",
        "start_char": 0,
        "end_char": 3,
        "ner": "O",
        "multi_ner": [
          "O"
        ]
      },
```

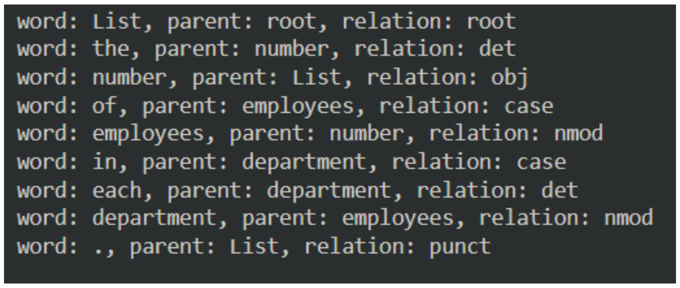
Fig. 2. Example of Stanza parsed sentence



```
{
  "id": 2,
  "text": "many",
  "lemma": "many",
  "upos": "ADJ",
  "xpos": "JJ",
  "feats": "Degree=Pos",
  "head": 3,
  "deprel": "amod",
  "start_char": 4,
  "end_char": 8,
  "ner": "O",
  "multi_ner": [
    "O"
  ]
},
{
  "id": 3,
  "text": "heads",
  "lemma": "head",
  "upos": "NOUN",
  "xpos": "NNS",
  "feats": "Number=Plur",
  "head": 8,
  "deprel": "nsubj",
  "start_char": 9,
  "end_char": 14,
  "ner": "O",
  "multi_ner": [
    "O"
  ]
```

Fig. 3. Example of Stanza parsed sentence

As you can see, there are several features that are determined for each word in the sentence. upos specifically refers to Universal Part-of-Speech which determines which part of speech that word is (verb, noun, adjective). This is the most important first step in our process as this determines how we are going to structure our SQL statement based on the original sentence. We also did dependency parsing which helps us determine how the words are related to other words in the sentence like shown below:



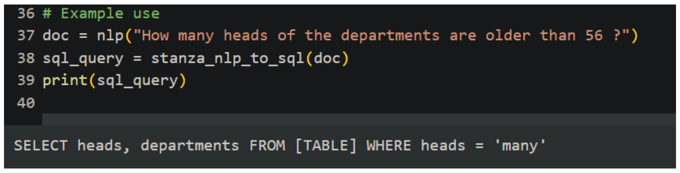
```
word: List, parent: root, relation: root
word: the, parent: number, relation: det
word: number, parent: List, relation: obj
word: of, parent: employees, relation: case
word: employees, parent: number, relation: nmod
word: in, parent: department, relation: case
word: each, parent: department, relation: det
word: department, parent: employees, relation: nmod
word: ., parent: List, relation: punct
```

Fig. 4. Example Dependency Parsing Results

Here, you can see each word having a relation and having a parent. We can use these to determine which word correlates to what other word which helps us understand the relationship these words have together in the sentence. The important factors include who, what, where, when, and why. The POS and dependency parsing help us determine what the SQL query should be doing as it includes information on who has control of what, whether it is an object of the sentence, etc.

Converting Parse into SQL

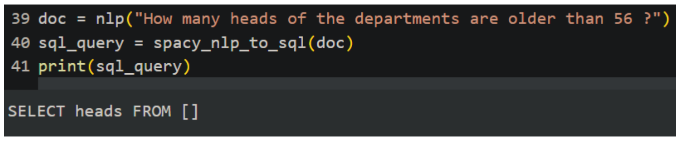
We originally wanted to develop a text-to-SQL parser designed to translate a natural prompt into executable SQL queries. Our main motivation to do this was that it simply was the most straightforward way to score and evaluate our parsers with and without our system. In theory, if we find improvement in the accuracy of the SQL queries, our project can be generalized for broader tasks. Despite the straightforward idea, this required us to develop our own methods to convert a parsed decomposed sentence into an SQL query which led to complications along the way. Complex parsings were difficult to interpret and translate, and the ambiguity of natural language meant the same sentence did not output an identical SQL query every time leading to inconsistent scores. We created different functions for converting to SQL for each parser since they have their own structures. While this was necessary to accommodate the different formats, it also has the downside of the process for converting not being exactly 1 to 1 for each parser. However, this was a risk we needed to take as it was the only way to move forward. This is definitely a place for improvement in the future.



```
# Example use
doc = nlp("How many heads of the departments are older than 56 ?")
sql_query = stanza_nlp_to_sql(doc)
print(sql_query)

SELECT heads, departments FROM [TABLE] WHERE heads = 'many'
```

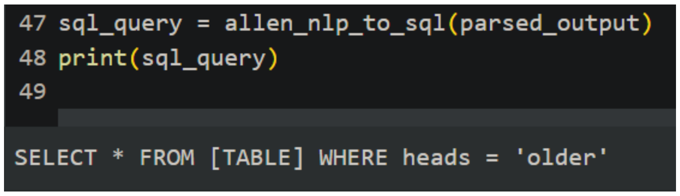
Fig. 5. Stanza Parsing



```
doc = nlp("How many heads of the departments are older than 56 ?")
sql_query = spacy_nlp_to_sql(doc)
print(sql_query)

SELECT heads FROM []
```

Fig. 6. Stanza Parsing

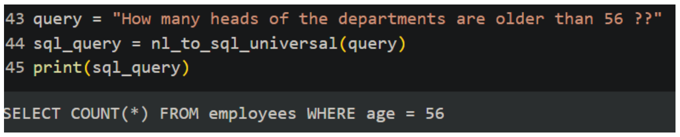


```
sql_query = allen_nlp_to_sql(parsed_output)
print(sql_query)

SELECT * FROM [TABLE] WHERE heads = 'older'
```

Fig. 7. Stanza Parsing

As you can see the results are not completely accurate, and in the case of spaCy in this particular example, it isn't even close to correct. The issue stems from our text to SQL conversion being too generalized. However, we made a function at some point that had an 'overfittting' approach to creating the query. This did come at the cost of only being useful for a certain type of query structure, so we ended up not using it. Although it yielded almost perfect results, it failed miserably in cases it was not meant for. We think the best balance is to have a bunch of these overfitting features for the function for different cases and have the best one be used depending on the case. But, that is for future work when more resources and time are available. Here is an example output from that overfitted response:



```
query = "How many heads of the departments are older than 56 ??"
sql_query = nl_to_sql_universal(query)
print(sql_query)

SELECT COUNT(*) FROM employees WHERE age = 56
```

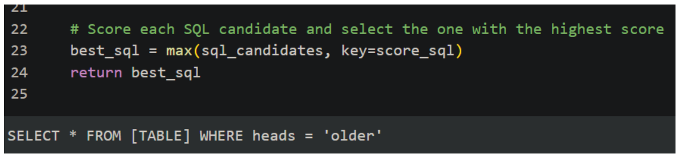
Fig. 8. An overfitting SQL conversion method that we ended up not using but provided promising results

It did better in recognizing the task (counting) and determined the age limit but failed in recognizing the correct reference to the department and the age comparison (equals instead of greater than).

Note that in our original 3 query figures above that we kept the table blank. This is due to our intentional decision to not include the table information from our dataset and allowing name recognition since it didn't fall in the scope of our project. This can be fixed in future work.

Applying Weak Supervision and Uncertainty Quantification

Weak Supervision has been proven to be a powerful tool to aggregate and refine noisy sources. Due to the nature of natural language ambiguity, most parsers and methods of understanding natural language will have noisy outputs. We applied weak supervision to the various parsings generated from the same prompt to improve general performance of parsers. Our Weak Supervision framework included the following heuristics: 1) Reward SQL queries that use key terms likely to be in the correct query. 2) Include a simplicity bias which penalizes overly complex queries. 3) Check if query terms appear in the SQL with simple context matching. We then scored each SQL candidate and selected the one with the highest score.



```
    # Score each SQL candidate and select the one with the highest score
    best_sql = max(sql_candidates, key=score_sql)
    return best_sql

SELECT * FROM [TABLE] WHERE heads = 'older'
```

Fig. 9. Results from Weak Supervision

Here you can see the results of weak supervision with the output being from AllenNLP's parser. While not completely correct, it can be seen as the most correct according to our specifications.

Uncertainty quantification has been utilized thoroughly in research, allowing the model to reassess and ensure its goal is accurately described. The natural ambiguity behind sentence parsing leads to the necessity of measuring the certainty of an output in order to get consistent results. To implement this idea, for each parsing outcome, the model generates a probability score that reflects its model's confidence. The system then checks the confidence based on historical performance i.e. if previous parsings had a certain average accuracy, we utilize this past performance to assess its confidence. When the confidence level of a parsing result falls below the threshold, the system automatically initiates the re-parsing process. If the confidence continues to remain low, the system then requests user input for clarification and explains in further detail the goal.

Results and Analysis

Results

The results of our study on improving semantic parsing through weak supervision and uncertainty quantification yielded promising outcomes, but with a huge room for improvement. Through our approach, we observed enhancements in the accuracy and robustness of semantic parsing models, showcasing the potential of weak supervision and uncertainty quantification techniques in NLP tasks. We personally found Stanza and AllenNLP to be the best parsers in context of parsing to determine SQL statements. However, it's clear that further optimizations are required to achieve better performance in this action space. While the results were satisfactory, they fell short of our expectations, indicating the need for refinement and fine-tuning in various aspects of the methodology. Our analysis is that addressing certain challenges, such as refining the weak supervision signals, optimizing model architectures, and fine-tuning uncertainty estimation methods, could lead to substantial improvements. Additionally, exploring complementary techniques and incorporating additional sources of supervision could contribute to enhancing the overall effectiveness of our proposed approach. Overall, while the results demonstrate promising progress, there is a clear opportunity for future research to build upon these findings.

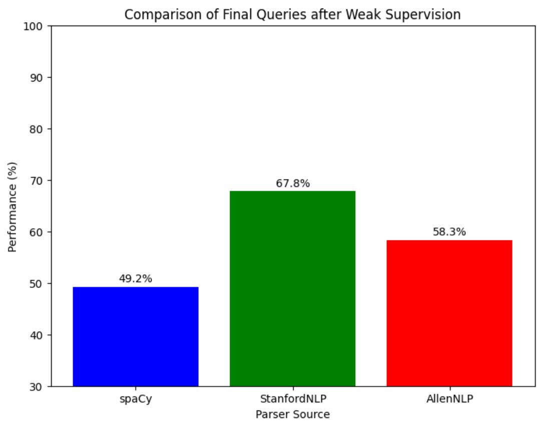


Fig. 10. Results from Weak Supervision

The above image shows our final query accuracy after weak supervision. spaCy did the worst as we mentioned.

Conclusion and Future Work

The work presented in this paper explored the potential of combining two existing approaches, weak supervision and uncertainty quantification, to improve the accuracy and robustness of semantic parsing within task-oriented domains. Drawing inspiration from the AMA and ZEROTOP papers, we aimed to develop a system that could generate diverse semantic parsings, filter the most likely translations, and proactively seek user clarification in situations where the parsing was uncertain. Utilizing various existing semantic parsers (AllenNLP, StanfordNLP, and SpaCy) as our baseline, we found that these parsers, while generally effective, struggled with the inherent ambiguity and complexity of natural language instructions or LLM outputs. This variability was consistent across the different parsers we tested, suggesting that the challenge is not tool-specific but rather intrinsic to the current state of semantic parsing technologies. One of the most significant challenges was the inherent difficulty in scoring and comparing the diverse semantic parsings generated, this aspect requires further refinement and potentially more universally applicable scoring. Scoring parsings based on the accuracy of the SQL query limited the scope of capabilities of our system. In the future, it would be helpful to evaluate our system's performance on a wider range of tasks and action spaces. The free version of Google Colab also posed a constraint, as the limited GPU resources hampered our ability to train and experiment with more complex testing. To further explore if our system can be an effective improvement, we would need to develop more advanced scoring mechanisms for weak supervision and finding the confidence threshold. Additionally, expanding our dataset beyond SQL query restrictions is needed to test a broader range of task-oriented domains. This would allow for a more thorough evaluation and significantly improve the applicability of our research. Investigating the feasibility to develop our own semantic parser and implement our ideas into the parser's core architecture could potentially lead to even more significant improvements in performance. While our scope of this project was limited, we hope this research is one more tool to bridge the gap between natural language and computational understanding and provide a better starting point for future work..

References

[1] Ask Me Anything: A simple strategy for prompting language models

[2] ZEROTOP: Zero-Shot Task-Oriented Semantic Parsing using Large Language Models