We propose clause-level edit methods with Python-style representations to use language models of code for SQL correction.

Our Method: Clause-Level Editing and Python Representation

- Token-level edits represented with special tokens can be ambiguous
  - Use clause-level edits to mitigate ambiguity issues
- Most language models of code (e.g., CodeT5) are not pre-trained on SQL
  - Propose Python-style representations to better use language models of code

```
sql = {
    "select": "SELECT singer_id,country,age",
    "from": "FROM singer",
    "groupBy": "GROUP BY age"
}
```

```
sql["select"] = "SELECT name,country,age"
sql.pop("groupBy")
sql["orderBy"] = "ORDER BY age DESC"
```

Text-to-SQL Error Correction

- Recent text-to-SQL parsers can reach decent accuracy, but they are still not accurate enough in practice
- We study how to correct errors in parser-generated SQL queries with language models of code

```
SELECT singer_id, country, age FROM singer
GROUP BY age
```

```
sql = {
    "select": "SELECT singer_id,country,age",
    "from": "FROM singer",
    "groupBy": "GROUP BY age"
}
```

```
sql["select"] = "SELECT name,country,age"
sql.pop("groupBy")
sql["orderBy"] = "ORDER BY age DESC"
```

Experiments and Results

- Data synthesis: 5-fold cross-validation on Spider for each parser
- Fine-tune CodeT5 (and CoditT5) on our synthesized data with different representations

Results

- CodeT5 shows considerably lower zero-shot perplexity on our proposed representation
- CodeT5 consistently achieves statistically significant improvement using our proposed representation
- Simulating user interactions with CodeT5 can further improve text-to-SQL parser's accuracy