**PURPLE: Improving Text-to-SQL with Logical Operator Composition and Demonstration-Based Retrieval**

**Abstract:**  
This research paper introduces PURPLE, a framework designed to address common limitations found in general Large Language Models (LLMs) in the domain of Text-to-SQL translation. PURPLE enhances performance by focusing on logical operator composition and demonstration-based retrieval. Through a novel four-stage process, PURPLE significantly improves translation accuracy in Text-to-SQL tasks, achieving superior results compared to state-of-the-art models across various metrics. This paper examines PURPLE's architecture, its success in handling complex SQL queries, and discusses its impact on common LLM limitations, as well as its own shortcomings.

**1. General LLM Challenges in Text-to-SQL**

Traditional LLMs face key limitations that hinder their performance in translating natural language to SQL queries. These limitations include:

**Input Length Restriction:**  
Many LLMs struggle with long inputs, often required for complex SQL queries that involve multiple clauses and conditions. These models may either truncate inputs or fail to generate comprehensive queries when the input exceeds a certain length.

**Logical Operator Composition Knowledge:**  
General LLMs lack a robust understanding of SQL operators and logical structures, which are critical for generating accurate and complex SQL queries. As a result, the SQL queries produced by LLMs may miss important logical connections, resulting in lower execution accuracy (EX) and exact match (EM).

### 2. LLM-Based NL2SQL Approaches

LLMs have been applied to NL2SQL tasks through two primary approaches:

**Zero-shot Approach:**  
In zero-shot NL2SQL, LLMs perform translation without any pre-annotated examples, relying solely on the current input query (X\mathcal{X}X) and database schema (D\mathcal{D}D). Models like **C3** and **ChatGPT-SQL** follow this method. Though effective for simple tasks, it struggles with handling complex queries that involve multiple logical operations and nested structures.

**Few-shot Approach:**  
In few-shot NL2SQL, LLMs are provided with a few annotated examples (demonstrations) from datasets, allowing them to learn how to map queries to SQL based on the input. The training set provides cross-domain demonstrations that assist LLMs in generating more accurate SQL for more challenging tasks. This method is beneficial in scenarios where logical operator composition is crucial.

### 3. Introduction of PURPLE to Overcome LLM Challenges

PURPLE was developed to specifically address these limitations by enhancing the ability of LLMs to generate accurate SQL queries through a four-stage framework. It leverages logical operator composition and a novel demonstration-based retrieval system to improve performance.

### 4. PURPLE Framework: Four Stages

PURPLE operates through four critical stages that ensure it generates SQL queries more accurately than traditional LLM approaches:

**Stage 1: Pruning**  
In the first stage, PURPLE filters out irrelevant demonstrations from the dataset, which narrows down the pool of SQL templates. This prevents the model from being overwhelmed by unnecessary examples and focuses on the most relevant ones.

**Stage 2: Skeleton Creation**  
In this stage, PURPLE generates an initial query skeleton—an outline of the SQL structure based on the input query. The skeleton includes high-level components like the SELECT and WHERE clauses but leaves out specific conditions, which will be refined in later stages.

**Stage 3: Demonstration Selection**  
PURPLE selects demonstrations based on the logical similarity between SQL queries. Instead of relying purely on syntactic similarity, PURPLE evaluates the logical structure of the target query and selects examples that share similar operator compositions. This allows PURPLE to better handle complex SQL operations and nested queries.

**Stage 4: Adaptation**  
Finally, PURPLE adapts the selected demonstrations to the specific input query by refining the skeleton with the necessary conditions and operators. This stage ensures that the final query closely matches the user's intent, even for complex tasks.

### 5. Success of PURPLE

PURPLE significantly improves performance on various complexity levels of SQL queries:

**Execution Accuracy (EX):**  
PURPLE's focus on operator composition enables it to generate queries that execute correctly, even for complex SQL tasks. It shows superior performance across all SQL hardness levels (easy, medium, hard, and extra hard), outperforming other state-of-the-art models.

**Exact Match (EM):**  
In terms of exact match, PURPLE's systematic approach to skeleton generation and adaptation results in SQL queries that more accurately reflect the original intent of the input. This leads to a higher EM score than competing models.

**Test-Suite Accuracy (TS):**  
Test-suite accuracy evaluates how well the generated SQL performs across a variety of test cases and scenarios. PURPLE excels in this metric due to its logical operator composition, ensuring robust performance across different input structures.

**Complex SQL Handling:**  
PURPLE's greatest strength lies in its ability to handle extra-hard SQL queries by emphasizing the importance of operator composition knowledge, which other models often neglect.

### 6. Drawbacks and Areas for Improvement

Despite its successes, PURPLE does face a few limitations:

**Dependence on Demonstrations:**  
PURPLE relies heavily on the quality and relevance of demonstrations in the dataset. If the demonstration pool lacks sufficient variety or fails to cover certain edge cases, PURPLE's performance may degrade.

**Computational Efficiency:**  
While PURPLE improves accuracy, its multi-stage process, particularly pruning and demonstration selection, can be computationally intensive. This may limit its scalability to very large datasets or real-time applications.

### Conclusion

In summary, PURPLE overcomes many of the core challenges faced by general LLMs in Text-to-SQL tasks through its unique focus on logical operator composition and demonstration retrieval. While it outperforms existing models across key metrics like Execution Accuracy, Exact Match, and Test-Suite Accuracy, future improvements could focus on reducing its computational overhead and expanding the demonstration pool to further enhance its versatility.