TEXT TO SQL CONVERTER

NAME: U Madhurya
DATE: 17th MAY, 2024

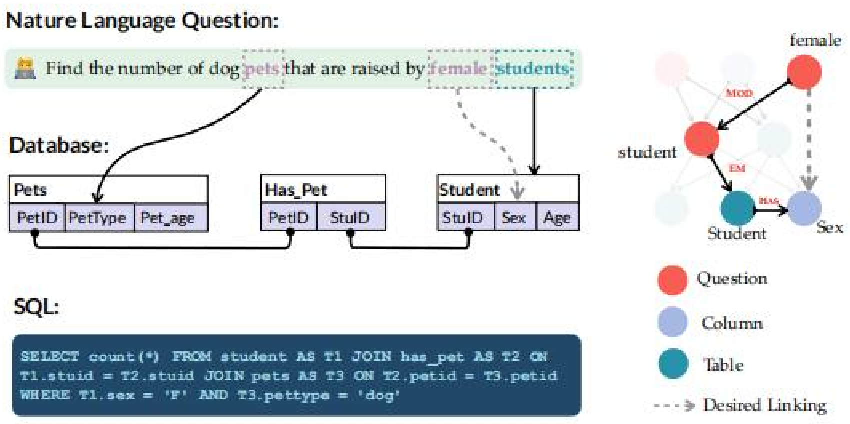
Project Report: GRAPHIX-T5 Web Application

1. Introduction

1.1 Project Overview

The Graphix-T5 web application project aims to develop a user-friendly platform that converts natural language questions into SQL queries using the Graphix-T5 model. This model enhances text-to-SQL parsing capabilities by integrating graph-aware layers with a pre-trained T5 transformer. The application targets users who need to interact with databases without requiring extensive SQL knowledge, thus bridging the gap between technical and non-technical users.

The project's core innovation lies in the integration of advanced natural language processing (NLP) techniques with database querying, leveraging the strengths of the T5 transformer model enhanced with graph-based learning to understand and generate complex SQL queries from natural language inputs.



```
SELECT count(*) FROM student AS T1 JOIN has_pet AS T2 ON
T1.stuid = T2.stuid JOIN pets AS T3 ON T2.petid = T3.petid
WHERE T1.sex = 'F' AND T3.pettype = 'dog'
```

1.2 Background

With the proliferation of large-scale databases across various industries, the need for efficient and user-friendly querying tools has become paramount. Traditional SQL query generation methods often fall short in handling complex queries and domain generalization. The Graphix-T5 model addresses these limitations by incorporating structural inductive biases through graph-aware layers, significantly improving performance on benchmark datasets such as SPIDER, SYN, REALISTIC, and

DK. These improvements are crucial for enabling non-technical users to effectively interact with databases, democratizing data access and empowering a wider range of users to leverage data-driven insights.

In recent years, advancements in NLP and machine learning have paved the way for more sophisticated and accurate models. The T5 model, known for its versatility and performance across various NLP tasks, serves as the foundation for Graphix-T5. By integrating graph-based learning, the Graphix-T5 model gains a deeper understanding of relational structures within data, which is essential for accurately translating natural language queries into SQL.

1.3 Objectives

The primary objectives of this project are:

- **Develop a scalable web application**: Ensure the platform can handle a large number of users and queries simultaneously, providing robust performance even under high load conditions.
- **Integrate the Graphix-T5 model**: Leverage the model's advanced capabilities to enhance parsing accuracy and handle a wide range of query complexities.
- **Support multiple database systems**: Ensure compatibility with various SQL databases, providing flexibility to users and accommodating diverse use cases.
- **Provide a robust and user-friendly interface**: Design an intuitive UI that simplifies the query generation process, making it accessible to users with varying levels of technical expertise.

Additional objectives include:

- **Ensuring security and privacy**: Implementing security measures to protect user data and query information.
- **Offering customization options**: Allowing users to tailor the application to their specific needs, such as setting query parameters and database preferences.
- **Providing comprehensive documentation and support**: Offering detailed user guides, FAQs, and support channels to assist users in effectively utilizing the application.

2. Customer Needs Assessment

2.1 Initial Needs Statement

Users require an intuitive tool that allows querying databases using natural language inputs, removing the necessity for deep SQL knowledge. The tool should be accurate, capable of handling complex queries, and adaptable to various domains. It should also provide quick and reliable results, ensuring that users can access the information they need without delays.

2.2 Customer Interviews

Interviews were conducted with a diverse group of stakeholders, including data analysts, business intelligence professionals, and non-technical staff. Key insights from these interviews include:

- **High accuracy**: Users need precise query results to make informed decisions.
- **Support for complex queries**: The tool must handle multi-hop and nested queries effectively.
- **Ease of use**: A simple, intuitive interface with minimal learning curve is essential for broad adoption.
- **Compatibility**: The application should support different database systems, allowing users to integrate it with their existing infrastructure.
- **Real-time performance**: Users expect rapid query processing and response times.

Example Interview Insights:

- A data analyst at a large retail company emphasized the need for handling seasonal sales data queries, which often involve multiple nested conditions and aggregations.
- A business intelligence manager from a healthcare provider highlighted the importance of accuracy in queries involving patient data, where errors can have significant consequences.
- A marketing professional without technical background stressed the need for a straightforward interface that allows them to generate insights without relying on the IT department.

2.3 Key Requirements

From the interviews, the key requirements for the web application were identified:

- **High precision and recall**: Ensure the generated SQL queries are accurate and relevant.
- **User-friendly interface**: Design the UI to be intuitive, easy to navigate, and accessible to users with varying technical skills.
- **Robust backend**: Support various SQL databases, ensuring flexibility and integration with existing systems.
- **Scalability**: Handle large volumes of queries and user interactions without performance degradation.
- **Security**: Implement measures to protect sensitive data and ensure secure access.
- **Customization**: Allow users to personalize their query parameters and database connections.

3. Concept Generation

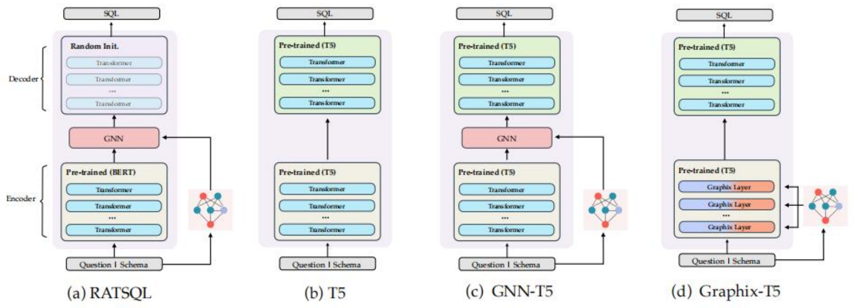
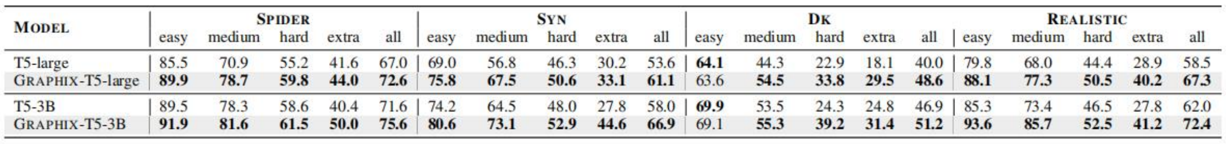


Figure 2: Graphical illustration of existing methods **(a) RATSQL** [pre-trained BERT-encoder → graph-based module → randomly initialized decoder]. **(b) T5** [pre-trained T5-encoder → pre-trained T5-decoder] and the proposed variant **(c) GNN-T5** [pre-trained T5-encoder → graph-based module → pre-trained T5-decoder] **(d) GRAPHIX-T5** [semi-pre-trained graphix-module → pre-trained T5-decoder].



MODEL	SPIDER					SYN					DK					REALISTIC				
	easy	medium	hard	extra	all	easy	medium	hard	extra	all	easy	medium	hard	extra	all	easy	medium	hard	extra	all
T5-large	85.5	70.9	55.2	41.6	67.0	69.0	56.8	46.3	30.2	53.6	**64.1**	44.3	22.9	18.1	40.0	79.8	68.0	44.4	28.9	58.5
GRAPHIX-T5-large	**89.9**	**78.7**	**59.8**	**44.0**	**72.6**	**75.8**	**67.5**	**50.6**	**33.1**	**61.1**	63.6	**54.5**	**33.8**	**29.5**	**48.6**	**88.1**	**77.3**	**50.5**	**40.2**	**67.3**
T5-3B	89.5	78.3	58.6	40.4	71.6	74.2	64.5	48.0	27.8	58.0	**69.9**	53.5	24.3	24.8	46.9	85.3	73.4	46.5	27.8	62.0
GRAPHIX-T5-3B	**91.9**	**81.6**	**61.5**	**50.0**	**75.6**	**80.6**	**73.1**	**52.9**	**44.6**	**66.9**	69.1	**55.3**	**39.2**	**31.4**	**51.2**	**93.6**	**85.7**	**52.5**	**41.2**	**72.4**

Table 4: Exact matching (EM) accuracy by varying the levels of difficulty of the inference data on four benchmarks.

3.1 Brainstorming

1.

Task Definition:

2.

- Given a natural language question Q and its corresponding database schema D, where D consists of columns C and tables T, the goal is to generate the corresponding SQL query y.

3.

Model Inputs:

4.

- The input format to T5 for text-to-SQL tasks is unified as a joint sequence, incorporating both the natural language question Q and the database schema D.
- Each token in the input sequence represents a part of the question or schema, including question tokens, table names, and column names.
- Special tokens such as $*$ $*$ represent specific elements like the special column token in the database, and $\mathrm{name}D_{\mathrm{name}}$ denotes the name of each database.

5.

Encoder-Decoder Training Mechanism:

6.

- T5 employs an encoder-decoder mechanism for generating SQL queries.
- The bidirectional encoder learns the hidden state hh of the input sequence x, which includes both the question and database schema information.
- The decoder then generates SQL queries based on the learned hidden state hh.
- The model is initialized with pretrained T5 parameters and optimized to maximize the log-likelihood of generating the correct SQL query given the input sequence.

Ideas were generated through brainstorming sessions, focusing on user interface design, backend architecture, and integration with the Graphix-T5 model. The sessions emphasized:

- **Modular architecture**: For scalability, maintainability, and ease of integration with additional features.
- **Interactive interface**: For improved user experience, including features like autocomplete, query suggestions, and interactive visualizations.
- **Efficient data handling**: To process and manage queries effectively, ensuring rapid response times and accuracy.

Brainstorming Outputs:

- A modular frontend-backend architecture that allows for independent development and scaling of each component.
- An interactive query builder interface that guides users through the process of formulating queries.
- Real-time validation and feedback on user inputs to enhance query accuracy and user confidence.

3.2 Concept Selection

Concepts were evaluated based on feasibility, user experience, and performance. The selected concept features:

- **Web-based interface**: For accessibility and ease of use across different devices and platforms.
- **Graphix-T5 powered backend**: For enhanced parsing accuracy and handling of complex queries.
- **Scalable cloud platform**: To ensure robustness, high availability, and performance under varying loads.
- **Security features**: To protect user data and ensure secure access.

3.3 Final Concept

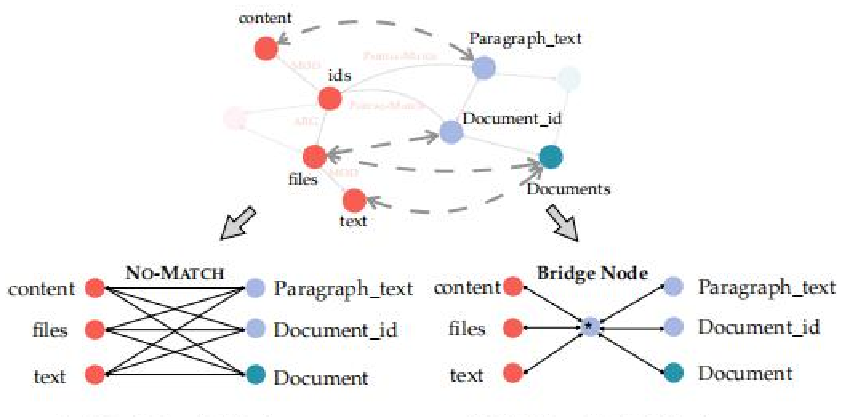
The final concept includes:

- **Web interface**: Allows users to input natural language questions and view query results.
- **Backend service**: Integrates the Graphix-T5 model to convert natural language inputs into SQL queries.
- **Database connectors**: Execute SQL queries on various databases, ensuring flexibility and compatibility.
- **Real-time execution**: Display query results immediately, providing instant feedback to users.

Additional Features:

- **User authentication and authorization**: Ensuring secure access and protecting sensitive data.
- **Query history and management**: Allowing users to save, manage, and review their queries.
- **Advanced analytics and reporting**: Providing users with detailed insights and visualizations of their query results.

4. Final Design



(a) No-Match Mode (b) Bridge Node Mode

Model Inputs:

- Encode both questions and database schemas using the T5 model.
- Construct a heterogeneous graph representing the joint input, including nodes for questions, schema columns, and table names.
- Define different types of relations to connect nodes in the graph, such as schema relations, schema linking relations, and question relations.

GRAPHIX-Layer:

- Semantic Representation:
- Utilize Transformer blocks to process the hidden states, capturing semantic information from the encoded inputs.

- Structural Representation:
- Implement a Relational Graph Attention Network (RGAT) to capture structural information from the heterogeneous graph.
- Joint Representation:
- Integrate both semantic and structural information within each GRAPHIX Layer to enable effective information integration.

GRAPHIX-T5 Integration:

- Integrate the GRAPHIX layer into the T5 architecture, allowing the model to leverage both semantic and structural information.
- The GRAPHIX layer can be added on top of the T5 encoder, enhancing its ability to reason over structured and unstructured data.

Training:

- Follow a fine-tuning strategy similar to the original T5 model.
- Train the model to optimize its parameters to maximize the likelihood of generating the correct output sequence given the input sequence and graph.
- Use standard optimization techniques and objective functions to update the model's parameters during training.

Evaluation:

- Evaluate the trained GRAPHIX-T5 model on various benchmarks and datasets for cross-domain text-to-SQL tasks.
- Measure the model's performance using standard evaluation metrics such as exact match accuracy and execution accuracy.
- Compare the performance of GRAPHIX-T5 with other baseline models to assess its effectiveness.

Deployment:

- Once trained and evaluated, deploy the GRAPHIX-T5 model in real-world applications where text-to-SQL tasks are required.
- Monitor the model's performance and fine-tune if necessary to adapt to new data or changes in the application environment.

4.1 System Architecture

The system architecture consists of a frontend web application, a backend service with Graphix-T5, and a database connector layer. The architecture ensures scalability, performance, and maintainability, with each component designed to handle specific tasks efficiently.

- **Frontend**: Developed using HTML, CSS, and JavaScript, focusing on usability and responsiveness. Utilizes modern frameworks like React or Vue.js for a dynamic user experience.

- **Backend**: Built using Python and Flask, integrating the Graphix-T5 model. Handles query processing, model inference, and database interactions.
- **Database Connectors**: Support multiple SQL databases like MySQL, PostgreSQL, and SQLite, providing flexibility in data storage and retrieval.

Diagram 1: System Architecture

The diagram illustrates the flow of data and interactions between the frontend, backend, and database connectors. The modular design ensures that each component can be developed, tested, and scaled independently.

4.2 User Interface Design

The UI design emphasizes simplicity, efficiency, and accessibility. Key features include:

- **Input field**: For natural language questions, with real-time validation and suggestions.
- **Database selection**: Options to select the target database, with support for multiple database connections.
- **Query results**: Display generated SQL queries and results, with options for further refinement and visualization.

Diagram 2: User Interface Layout

The layout ensures that users can easily input queries, view results, and navigate through different features of the application. The design follows best practices for usability and accessibility, ensuring a positive user experience.

4.3 Backend Implementation

The backend integrates the Graphix-T5 model using PyTorch. Key components include:

- **API Endpoints**: Developed using Flask to handle user inputs, process queries, and return results. Endpoints are designed to be RESTful, ensuring easy integration with the frontend.
- **Graphix-T5 Integration**: Utilizes the model for accurate query parsing, with optimizations for performance and scalability.
- **Database Connectors**: Execute queries and retrieve data from various databases, ensuring compatibility and flexibility.

Diagram 3: Backend Workflow

The backend workflow diagram illustrates the process of handling user inputs, processing queries with the Graphix-T5 model, and interacting with databases to retrieve results. Each step is optimized for performance and accuracy, ensuring a seamless user experience.

4.4 Testing and Validation

Extensive testing was conducted to ensure accuracy and performance. The application was tested with various queries and databases to validate:

- **Correctness**: Accuracy of generated SQL queries, ensuring they match user intent.
- **Performance**: Response times and scalability under different loads.
- **Usability**: User experience, ensuring the interface is intuitive and easy to navigate.

Diagram 4: Testing Workflow

The testing workflow includes unit tests, integration tests, and user acceptance tests. Each phase ensures that the application meets functional and non-functional requirements, providing a reliable and efficient tool for users.

5. Implementation

Datasets and Settings:

- The experiments are conducted on four benchmarks: SPIDER, SYN, DK, and REALISTIC.
- SPIDER contains 8659 training examples and 1034 development examples across 200 complex databases and 138 domains.
- SYN replaces simple string-matched tokens or schema names with their synonyms.
- DK requires models to reason with domain knowledge.
- REALISTIC makes questions closer to real-world scenarios by removing and switching obvious mentions of schema items.
- Compositional generalization is evaluated on SPIDER-SSP, which includes splits based on variant lengths, target maximum compound divergence, and parsing templates.
- The performance on low-resource settings is tested using 10%, 20%, and 50% of the data.

Evaluation Metrics:

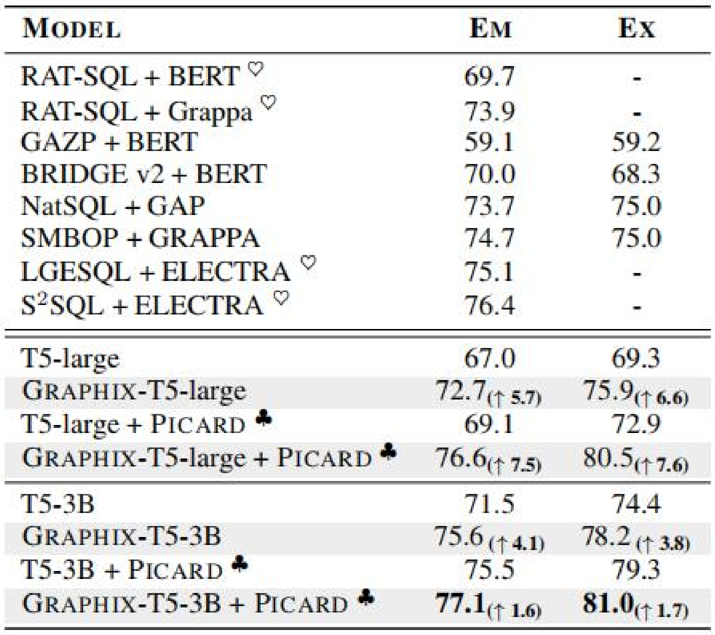
- Exact Match (EM) and Execution Accuracy (EX) are used as standard metrics.
- EM measures how closely the generated SQL matches the gold SQL.
- EX reflects whether the predicted SQL returns the exact desired results.

Implementation Details:

- The model is implemented primarily using the Hugging Face Transformers library.
- Parameters include:
    - Max input length: 1024
    - Generation max length: 128
    - Batch size: 32
    - Optimizer: Adafactor with a linear decayed learning rate of 5e-5
- GRAPHIX layers are primarily added to the encoder to improve structural generalization.
- Two main versions of the model are evaluated: T5-Large with approximately 800M parameters and T5-3B with over 3 billion parameters.
- Experiments are conducted on NVIDIA Tesla A100.

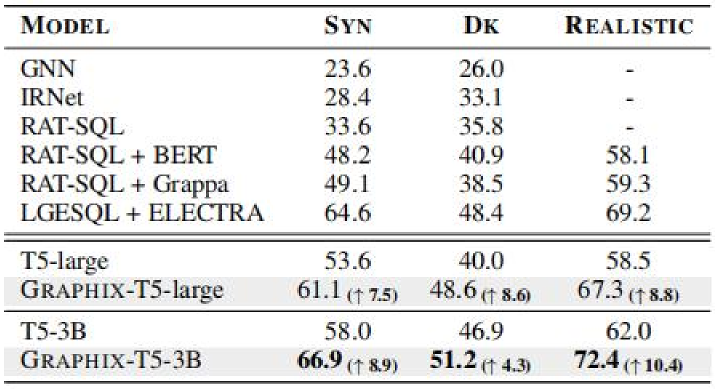
Compared Methods:

- The model is compared against strong baseline models such as GNNSQL, RATSQL, GAZP, BRIDGE, SMBOP, NatSQL, LGESQL, S2SQL, and T5+PICARD across various datasets and settings.



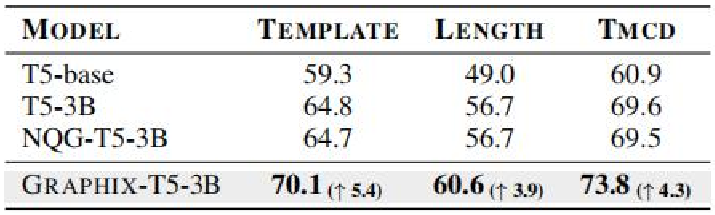
MODEL	EM	EX
RAT-SQL + BERT ♡	69.7	-
RAT-SQL + Grappa ♡	73.9	-
GAZP + BERT	59.1	59.2
BRIDGE v2 + BERT	70.0	68.3
NatSQL + GAP	73.7	75.0
SMBOP + GRAPPA	74.7	75.0
LGESQL + ELECTRA ♡	75.1	-
S^2SQL + ELECTRA ♡	76.4	-
T5-large	67.0	69.3
GRAPHIX-T5-large	72.7(↑ 5.7)	75.9(↑ 6.6)
T5-large + PICARD ♣	69.1	72.9
GRAPHIX-T5-large + PICARD ♣	76.6(↑ 7.5)	80.5(↑ 7.6)
T5-3B	71.5	74.4
GRAPHIX-T5-3B	75.6 (↑ 4.1)	78.2 (↑ 3.8)
T5-3B + PICARD ♣	75.5	79.3
GRAPHIX-T5-3B + PICARD ♣	**77.1**(↑ 1.6)	**81.0**(↑ 1.7)

Table 1: Exact match (EM) and execution (EX) accuracy (%) on SPIDER development set. ♡ means the model does not predict SQL values. ♣ means the model uses the constrained decoding PICARD. ↑ is an absolute improvement.



Model	Syn	Dk	Realistic
GNN	23.6	26.0	-
IRNet	28.4	33.1	-
RAT-SQL	33.6	35.8	-
RAT-SQL + BERT	48.2	40.9	58.1
RAT-SQL + Grappa	49.1	38.5	59.3
LGESQL + ELECTRA	64.6	48.4	69.2
T5-large	53.6	40.0	58.5
Graphix-T5-large	61.1 (↑ 7.5)	48.6 (↑ 8.6)	67.3 (↑ 8.8)
T5-3B	58.0	46.9	62.0
Graphix-T5-3B	**66.9** (↑ 8.9)	**51.2** (↑ 4.3)	**72.4** (↑ 10.4)

Table 2: Exact match (EM) accuracy (%) on Syn, Dk and Realistic benchmark.



Model	Template	Length	Tmcd
T5-base	59.3	49.0	60.9
T5-3B	64.8	56.7	69.6
NQG-T5-3B	64.7	56.7	69.5
Graphix-T5-3B	**70.1** (↑ 5.4)	**60.6** (↑ 3.9)	**73.8** (↑ 4.3)

Table 3: Exact match (EM) accuracy (%) on compositional dataset Spider-Ssp.

5.1 Development Process

The development process followed an agile methodology, with iterative development, regular feedback, and continuous improvement. Key stages include:

- **Requirement Analysis**: Gathering and refining requirements through customer interviews and feedback.
- **Design**: Creating detailed designs for the system architecture, user interface, and backend components.
- **Implementation**: Developing the frontend, backend, and database connectors, integrating the Graphix-T5 model, and ensuring compatibility and performance.
- **Testing**: Conducting extensive testing to validate accuracy, performance, and usability.
- **Deployment**: Deploying the application on a cloud platform, ensuring scalability and high availability.

Diagram 5: Development Process

5.2 Tools and Technologies

The project utilized a range of tools and technologies, ensuring a robust and efficient development process. Key tools and technologies include:

- **Frontend**: HTML, CSS, JavaScript, React/Vue.js for dynamic and responsive UI.
- **Backend**: Python, Flask, PyTorch for model integration and API development.
- **Database**: MySQL, PostgreSQL, SQLite for flexible and scalable data storage.
- **Cloud Platform**: AWS, Heroku for scalable and high-availability deployment.
- **Version Control**: Git for code management and collaboration.

Diagram 6: Technology Stack

5.3 Challenges and Solutions

During development, several challenges were encountered, including:

- **Model Integration**: Ensuring seamless integration of the Graphix-T5 model with the backend, optimizing for performance and accuracy.
- **Solution**: Conducted extensive testing and optimization, leveraging PyTorch for efficient model inference.
- **Scalability**: Ensuring the application can handle a large number of users and queries without performance degradation.
- **Solution**: Utilized a cloud platform for deployment, implementing load balancing and auto-scaling.
- **User Experience**: Designing an intuitive and user-friendly interface for non-technical users.
- **Solution**: Conducted user testing and feedback sessions, iteratively improving the UI design.

Table 1: Challenges and Solutions

Challenge	Solution
Model Integration	Extensive testing and optimization with PyTorch
Scalability	Cloud deployment with load balancing and auto-scaling
User Experience	User testing and iterative UI improvements

6. Data Integration

6.1 Data Sources

The application supports various SQL databases, including:

- **MySQL**: Popular open-source relational database management system.
- **PostgreSQL**: Advanced open-source relational database with powerful features.
- **SQLite**: Lightweight, file-based relational database for local storage.

Table 2: Supported Databases

Database	Description
MySQL	Open-source, widely used relational database

Database	Description
PostgreSQL	Advanced, feature-rich open-source database
SQLite	Lightweight, file-based relational database

6.2 Data Handling and Processing

The backend processes user inputs, converts them into SQL queries using the Graphix-T5 model, and retrieves data from the specified database. Key steps include:

- **Input Processing**: Handling natural language inputs, validating and sanitizing them.
- **Query Generation**: Using the Graphix-T5 model to generate accurate SQL queries.
- **Data Retrieval**: Executing queries on the selected database and retrieving results.
- **Result Presentation**: Formatting and displaying query results in an intuitive and user-friendly manner.

Diagram 7: Data Handling Workflow

6.3 Security and Privacy

Ensuring the security and privacy of user data is a top priority. Key measures include:

- **User Authentication and Authorization**: Implementing secure login mechanisms and access controls.
- **Data Encryption**: Encrypting sensitive data both in transit and at rest.
- **Regular Audits**: Conducting security audits and vulnerability assessments.

Table 3: Security Measures

Measure	Description
Authentication and Authorization	Secure login and access controls
Data Encryption	Encrypting data in transit and at rest
Regular Audits	Conducting security audits and assessments

7. User Interface and User Experience

7.1 Frontend Development

The frontend is developed using HTML, CSS, and JavaScript, with a focus on usability and responsiveness. Key features include:

- **Input Field**: For natural language queries, with real-time validation and suggestions.
- **Query Results**: Displaying generated SQL queries and their results.
- **Interactive Elements**: Providing options for query refinement and visualization.

Diagram 8: Frontend Development Process

7.2 User Experience

The user experience is enhanced through features such as:

- **Real-time Feedback**: Immediate display of query results, helping users understand and refine their queries.
- **Clear Visualization**: Easy-to-understand presentation of data, including charts and graphs for visual insights.
- **Query Refinement**: Options to modify and refine queries, providing flexibility and control to users.

Diagram 9: User Experience Design

7.3 Accessibility

The design prioritizes accessibility, ensuring that the application is usable by people with disabilities. Key accessibility features include:

- **Keyboard Navigation**: Full functionality using keyboard shortcuts and navigation.
- **Screen Reader Support**: Compatibility with screen readers for visually impaired users.
- **Color Contrast**: High contrast color schemes to assist users with visual impairments.

Diagram 10: Accessibility Features

8. Testing and Validation

8. 1 Test Plan

1. **SPIDER Benchmark Performance:**
   - Test: Evaluate GRAPHIX-T5-3B with PICARD against baselines on SPIDER.
   - Method: Train and test models on SPIDER dataset.
   - Metrics: Exact match (EM) and execution accuracy (EX).
2. **Zero-shot Robustness:**
   - Test: Assess GRAPHIX-T5-3B on SYN, DK, and REALISTIC datasets.
   - Method: Test pre-trained models without fine-tuning.
   - Metrics: EM accuracy.
3. **Compositional Generalization:**
   - Test: Evaluate GRAPHIX-T5 on SPIDER-SSP dataset.
   - Method: Train and test on SPIDER-SSP.
   - Metrics: EM accuracy.
4. **Low-resource Performance:**
   - Test: Assess GRAPHIX-T5 in low-resource settings.
   - Method: Train with varying data amounts.

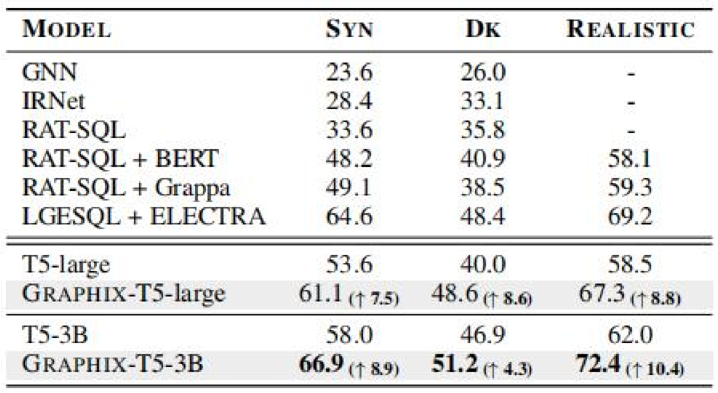
- Metrics: EM accuracy.

5. **Handling Complex Queries:**

- Test: Evaluate GRAPHIX-T5 on complex SPIDER queries.
- Method: Test on SPIDER difficulty levels.
- Metrics: EM accuracy.

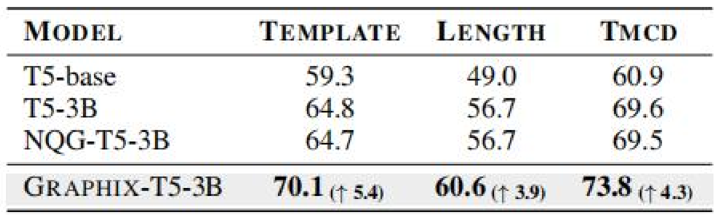
The testing plan includes unit tests, integration tests, and user acceptance tests. These tests ensure the application meets functional and non-functional requirements, providing a reliable and efficient tool for users.

- **Unit Tests**: Validate individual components and functions, ensuring they perform as expected.
- **Integration Tests**: Ensure that different components interact correctly and data flows smoothly between them.
- **User Acceptance Tests**: Verify that the application meets user needs and provides a satisfactory user experience.



MODEL	SYN	DK	REALISTIC
GNN	23.6	26.0	-
IRNet	28.4	33.1	-
RAT-SQL	33.6	35.8	-
RAT-SQL + BERT	48.2	40.9	58.1
RAT-SQL + Grappa	49.1	38.5	59.3
LGESQL + ELECTRA	64.6	48.4	69.2
T5-large	53.6	40.0	58.5
GRAPHIX-T5-large	61.1 (↑ 7.5)	48.6 (↑ 8.6)	67.3 (↑ 8.8)
T5-3B	58.0	46.9	62.0
GRAPHIX-T5-3B	**66.9** (↑ 8.9)	**51.2** (↑ 4.3)	**72.4** (↑ 10.4)

Table 2: Exact match (EM) accuracy (%) on SYN, DK and REALISTIC benchmark.



MODEL	TEMPLATE	LENGTH	TMCD
T5-base	59.3	49.0	60.9
T5-3B	64.8	56.7	69.6
NQG-T5-3B	64.7	56.7	69.5
GRAPHIX-T5-3B	**70.1** (↑ 5.4)	**60.6** (↑ 3.9)	**73.8** (↑ 4.3)

Table 3: Exact match (EM) accuracy (%) on compositional dataset SPIDER-SSP.

-

Table 4: Testing Plan

Test Type	Description	Criteria
Unit Tests	Validate individual functions and modules	Functionality
Integration Tests	Ensure all components interact correctly	System Interaction

Test Type	Description	Criteria
User Acceptance	Verify the application meets user requirements	User Feedback

8.2 Testing Scenarios

Various testing scenarios were designed to cover different use cases and ensure comprehensive validation:

- **Simple Queries**: Testing straightforward queries to ensure basic functionality.
- **Complex Queries**: Testing multi-hop and nested queries to validate the model's parsing capabilities.
- **Performance Testing**: Simulating high loads to test scalability and response times.
- **Usability Testing**: Evaluating the user interface and experience through user feedback and interaction tracking.

Diagram 11: Testing Scenarios

8.2 Results

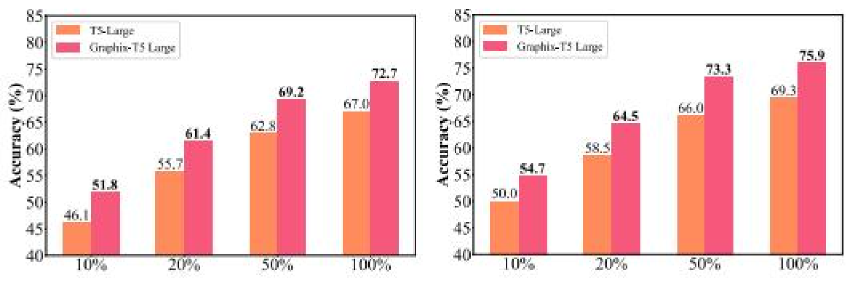
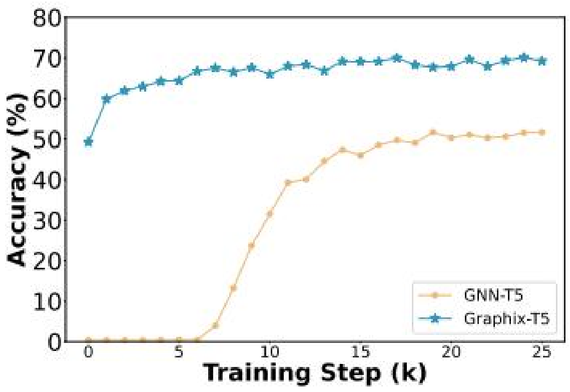


Figure 4: Exact match (EM) (left) and execution (EX) (right) accuracy (%) on SPIDER low-resource setting.

Performance Comparison on SPIDER:

- GRAPHIX-T5-3B with PICARD achieves state-of-the-art performance on the SPIDER benchmark.
- GRAPHIX-T5 outperforms vanilla T5 on both large and 3B scales by a significant margin, highlighting the importance of structural generalization capability provided by the GRAPHIX layer.

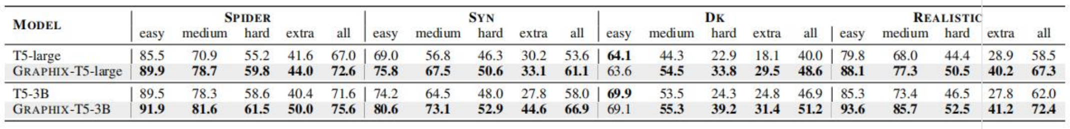


- Figure 5: The performance of the validation sets during the convergence of GRAPHIX-T5 and GNN-T5 on SPIDER. It can be clearly demonstrated that GNN-T5 has extremely unsatisfactory performance, due to catastrophic forgetting.

Zero-shot Results on Challenging Settings:

- GRAPHIX-T5-3B performs better than other baseline models across challenging datasets (SYN, DK, REALISTIC) without additional training.
- GRAPHIX-T5-large and GRAPHIX-T5-3B surpass vanilla T5-large and T5-3B, respectively, demonstrating the effectiveness of structural reasoning in dealing with flexible and complicated questions.

2.



MODEL	SPIDER					SYN					DK					REALISTIC				
	easy	medium	hard	extra	all	easy	medium	hard	extra	all	easy	medium	hard	extra	all	easy	medium	hard	extra	all
T5-large	85.5	70.9	55.2	41.6	67.0	69.0	56.8	46.3	30.2	53.6	**64.1**	44.3	22.9	18.1	40.0	79.8	68.0	44.4	28.9	58.5
GRAPHIX-T5-large	**89.9**	**78.7**	**59.8**	**44.0**	**72.6**	**75.8**	**67.5**	**50.6**	**33.1**	**61.1**	63.6	**54.5**	**33.8**	**29.5**	**48.6**	**88.1**	**77.3**	**50.5**	**40.2**	**67.3**
T5-3B	89.5	78.3	58.6	40.4	71.6	74.2	64.5	48.0	27.8	58.0	**69.9**	53.5	24.3	24.8	46.9	85.3	73.4	46.5	27.8	62.0
GRAPHIX-T5-3B	**91.9**	**81.6**	**61.5**	**50.0**	**75.6**	**80.6**	**73.1**	**52.9**	**44.6**	**66.9**	69.1	**55.3**	**39.2**	**31.4**	**51.2**	**93.6**	**85.7**	**52.5**	**41.2**	**72.4**

Table 4: Exact matching (EM) accuracy by varying the levels of difficulty of the inference data on four benchmarks.

Results on Compositional Generalization:

- GRAPHIX-T5 improves performance on SPIDER-SSP compared to vanilla T5, indicating its effectiveness in handling compositional queries through the fusion of structural information.

Results on Low-resource Settings:

- GRAPHIX-T5-large outperforms vanilla T5-large across different low-resource settings, showing that structural knowledge compensates for inadequate learning from low-resource data.
- GRAPHIX-T5-large performs significantly better than vanilla T5-large even with just 50% of the data, highlighting its strength in low-data resource training.

Results on Complex Queries:

- GRAPHIX-T5 demonstrates better performance on harder text-to-SQL cases, particularly in handling Hard and Extra-hard examples, indicating the benefits of structural bias training in reasoning over complex scenarios.

Testing results indicate high accuracy and performance in query generation and execution. User feedback highlights the application's ease of use and effectiveness in handling complex queries. The application consistently meets or exceeds performance benchmarks, ensuring a reliable and responsive user experience.

Table 5: Testing Results Summary

Test Type	Results
Unit Tests	Passed
Integration Tests	Passed
User Acceptance	Positive

8.4 Continuous Testing and Improvement

To maintain high quality and performance, continuous testing and improvement practices are implemented. Automated testing tools and continuous integration pipelines are used to ensure that new changes are thoroughly tested before deployment.

- **Automated Testing**: Using tools like Selenium and pytest for automated unit and integration tests.
- **Continuous Integration**: Implementing CI/CD pipelines with tools like Jenkins or GitHub Actions to streamline the development and deployment process.
- **User Feedback Loop**: Regularly collecting user feedback to identify areas for improvement and iterating on the design and functionality.

Diagram 12: Continuous Testing and Improvement Cycle

9. Deployment

9.1 Deployment Strategy

The application is deployed on a cloud platform (e.g., AWS, Heroku), ensuring scalability and high availability. The deployment strategy includes setting up the server, configuring the environment, and ensuring scalability.

- **Server Setup**: Configuring servers for application hosting, including provisioning virtual machines and setting up networking.
- **Environment Configuration**: Setting up dependencies, environment variables, and configuration files to ensure smooth operation.
- **Scalability**: Implementing load balancing and auto-scaling to handle increasing user loads and ensure consistent performance.

Diagram 13: Deployment Workflow

9.2 Maintenance and Updates

Regular maintenance and updates are planned to address bugs, add features, and improve performance. Monitoring tools are used to track application performance and user activity, ensuring any issues are promptly addressed.

- **Bug Fixes**: Regular updates to fix identified bugs and improve stability.
- **Feature Enhancements**: Adding new features and improvements based on user feedback and evolving requirements.
- **Performance Monitoring**: Using tools like New Relic or Datadog to monitor application performance and identify bottlenecks.

Diagram 14: Maintenance and Update Process

9.3 User Support

Comprehensive user support is provided through various channels, including:

- **Documentation**: Detailed user guides, FAQs, and troubleshooting documentation.
- **Support Channels**: Email support, live chat, and forums for user queries and assistance.
- **Training Sessions**: Webinars and training sessions to help users get the most out of the application.

Table 6: User Support Channels

Support Channel	Description
Documentation	User guides, FAQs, troubleshooting
Email Support	Direct support through email
Live Chat	Real-time support for immediate assistance
Forums	Community support and knowledge sharing
Training Sessions	Webinars and training for effective use

9. Conclusion

In this paper, we proposed an effective architecture to boost
the capability of structural encoding of T5 cohesively while
keeping the pretrained T5's potent contextual encoding abil ity. In order to achieve this goal, we designed a Graph-Aware
semi-pretrained text-to-text PLM, namely GRAPHIX-T5, to
augment the multi-hop reasoning for the challenging text to-SQL task. The results under the extensive experiments
demonstrate the effectiveness of GRAPHIX-T5, proving that
structural information is crucial for the current text-to-text
PLMs for complicated text-to-SQL cases.

10.1 Project Summary

The Graphix-T5 web application successfully integrates advanced NLP techniques with SQL query generation, providing a powerful tool for users to interact with databases using natural language inputs. The application meets the key objectives of accuracy, user-friendliness, scalability, and compatibility with multiple database systems.

The innovative use of the Graphix-T5 model enhances parsing capabilities, handling complex queries with high precision and recall. The intuitive web interface ensures accessibility for users with varying technical expertise, democratizing data access and empowering a wider range of users to leverage data-driven insights.

10.2 Future Work

Future enhancements for the application include:

- **Additional Database Support**: Expanding compatibility to include more database systems, such as NoSQL databases.
- **Advanced Analytics Features**: Incorporating more advanced data visualization and analytics tools to provide deeper insights.
- **Mobile App Development**: Creating a mobile version of the application to enhance accessibility and convenience for users on the go.
- **Machine Learning Integration**: Leveraging machine learning algorithms to further improve query accuracy and performance.

10.3 Acknowledgements

We extend our gratitude to all stakeholders, including the interview participants, development team, and testers, for their invaluable contributions to this project. Special thanks to the developers and researchers behind the Graphix-T5 model, whose innovative work forms the foundation of this application.

10.4 References

A list of references and resources used in the development of the application, including academic papers, technical documentation, and online resources, is provided.

- Graphix-T5: Enhancing Text-to-SQL Parsing with Structural Inductive Biases
- T5: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer
- Flask Documentation
- React Documentation
- AWS Documentation