PANL-SQL: Palestinian Arabic Natural Language to SQL Translator on University Database

Obada Tahayna
Dept. Computer Science
Birzeit University
Ramallah, Palestine
1191319@student.birzeit.edu

Ibrahim Duhaide
Dept. Computer Engineering
Birzeit University
Ramallah, Palestine
1190283@student.birzeit.edu

Omar Etkaidek
Dept. Computer Engineering
Birzeit University
Ramallah, Palestine
1192301@student.birzeit.edu

***Abstract*—This paper investigates the application of the AraT5 transformer model in the realm of Arabic Natural Language (NL) to SQL translation, focusing specifically on the Palestinian dialect within a university database context. We adopted a sequential fine-tuning approach that first involved leveraging the large English NL-SQL dataset WikiSQL. It was subsequently translated into Arabic using the Turjuman model, which provided an extensive corpus of NL-SQL pairs for initial model fine-tuning. The second phase of fine-tuning utilized our uniquely curated university dataset, imbued with the particularities of our database schema and the nuances of the Palestinian dialect. This dual-layered approach fostered a thorough understanding of SQL syntax and the semantic complexities of Arabic NL in our model. The model demonstrated promising proficiency in generating SQL queries from Arabic NL inputs but also underscored areas for enhancement, especially in processing complex queries and managing unknown labels. This study signifies an important advancement in NL to SQL systems, particularly for underrepresented languages such as Arabic, paving the way for ongoing optimization and refinement. The datasets, model, and fine-tuning codes are open-sourced and available on GitHub.[1]**

***Index Terms*—Natural Language Interface to Database (NLIDB), Dataset Creation, AraT5, Sequential Fine-tuning**

1. Introduction

In the age of data-driven decision making, the need for intuitive and user-friendly systems for data retrieval has become paramount. Traditionally, accessing structured relational databases requires knowledge of specific query languages like SQL (Structured Query Language) [1]. However, this presents a hurdle for ordinary users who may not have the necessary technical knowledge. Natural Language Interface to Database (NLIDB) systems have thus emerged as a solution to bridge this gap, providing an interface where users can input queries in a natural language and retrieve desired information [2].

1. gist.github.com/obada-jaras/72702434577fd9cc298043b862172b2a

In this work, we present PANL2SQL: a Palestinian Arabic NLIDB System for a University Database. This system is designed to translate Palestinian Arabic natural language queries into SQL statements, providing an intuitive information retrieval system specifically for students and staff of a university setting. Our aim is to facilitate easier access to structured data such as course details, faculty information, department specifics and more, all in the users' native language.

PANL2SQL is unique in its focus on Palestinian Arabic [3], a language variant that may differ significantly from Modern Standard Arabic (MSA) [4] in syntax, vocabulary, and morphology. The system leverages the AraT5 transformer [5], an Arabic variant of the T5 model [6], for translating natural language queries into SQL. This Arabic focused model allows the system to better capture the linguistic nuances of Palestinian Arabic and improve the translation accuracy.

This paper will detail the development, implementation, and evaluation of the PANL2SQL system. We will discuss the generation of our datasets, the sequential fine-tuning process [7] we applied on the AraT5 model, and the practical application of the system on our university database. Furthermore, we will discuss the performance of the system.

The key contribution of this work lies not only in its application of natural language processing techniques [8] to a specific dialect of Arabic, but also in its potential as a scalable model for similar tasks in other languages and dialects. Our system serves as a proof-of-concept for how language-specific NLIDB systems can significantly improve the accessibility and usability of structured data [9].

2. Methodology

Our approach involves three core stages: preparation of the dataset, fine-tuning of the AraT5 model, and evaluation of the system's performance. While this section offers an overview of the entire process, finer details will be discussed more extensively in the Implementation section.

2.1. Dataset Preparation

Dataset creation involved two main tasks: creating a general dataset and a specific dataset related to our university database.

2.1.1. General Dataset. To establish the general dataset, our initial step involved utilizing the WikiSQL dataset.With approximately 80,654 examples of English Natural Language (NL) to SQL query pairs, providing a vast set of examples [10]. However, our project required Arabic NL queries, prompting us to translate the natural language portion of each pair from English to Arabic.

To accomplish this, we translated each English NL query in the dataset into three distinct translations. This process multiplied the initial 80,654 pairs into approximately 241,962 Arabic NL-SQL pairs.

2.1.2. Specific Dataset. Our specific dataset, on the other hand, was tailored to our university database schema. We began by defining 70 general queries that could be asked about the database. For each of these queries, we wrote them in Modern Standard Arabic (MSA) along with the corresponding SQL statements. We ensured that the SQL queries were correct and matched the intended meaning of the NL queries. After generating these MSA-SQL pairs, we proceeded to translate them into Palestinian Arabic dialect using a web form completed by our university students. This process resulted in around 2,844 dialect-specific NL-SQL pairs.

2.2. Model Selection and Fine-tuning

Our model of choice for this project was the AraT5 transformer, this model was specifically developed for Arabic NLP tasks, making it ideal for our project. The fine-tuning process was a sequential fine-tuning using the HuggingFace transformers [11] library involving the following two steps:

2.2.1. General Fine-tuning. In the first step, we fine-tuned the AraT5 model on the general translated WikiSQL dataset. This helped the model learn the basic structure of SQL queries and establish a high-level understanding of the relationship between Arabic words and SQL.

2.2.2. Specific Fine-tuning. After general fine-tuning, we proceeded with a second round of fine-tuning, this time using our specific dataset. This was done to adapt the model to our specific use case, translating Palestinian Arabic NL queries into SQL for our university database. We used a greater learning rate in this step to emphasize the importance of the specific dataset in the model's learning process.

2.3. Evaluation

Once the fine-tuning process is complete, we proceed to evaluate the performance of our model. We use common NLP evaluation metrics BLEU [12], to assess the quality of the generated SQL queries. Also we perform manual inspections and use-case tests to observe how well the model performs in real-world scenarios. Through this processes, we ensure a thorough evaluation of the model's capabilities.

3. Implementation

3.1. Dataset Preparation

Creating the datasets required for our research involved two main steps: crafting a general dataset and designing a specific dataset, which was finely tuned to our university database. This process was necessary due to the absence of any pre-existing Arabic Natural Language (NL) to SQL datasets online.

3.1.1. General Dataset. The creation of the general dataset started with the WikiSQL dataset, an invaluable resource within the Natural Language Interface to Database (NLIDB) community. The WikiSQL dataset encompasses 80,654 examples of English NL to SQL pairs, spread across 24,241 tables extracted from Wikipedia [10].

Given the focus of our project on Arabic NL queries, the English NL queries of each pair in the WikiSQL dataset had to be translated to Arabic. To accomplish this, we leveraged Turjuman [13], a translation model established on the foundation of the AraT5 transformer. Turjuman is specifically designed to provide translations from various languages to Arabic.

For each English NL query, Turjuman was used to generate three separate translations. This approach increased the size of the original dataset significantly, as each NL-SQL pair was essentially tripled, culminating in 241,962 Arabic NL-SQL pairs. Importantly, these pairs contained 80,654 distinct SQL queries, preserving the diversity of the original WikiSQL dataset. Although the translations might not be perfect or very high quality, the main goal here was to train the model on how to structure SQL queries and provide it a high-level understanding of the correlation between Arabic words and SQL.

3.2. Specific Dataset

In contrast to the general dataset, we custom-built a specific dataset tailored to the requirements of our university database schema. This process, laboriously executed manually, allowed us to infuse the dataset with queries that were highly relevant to our specific database.

The inception of the specific dataset process involved the identification and formulation of 70 frequently asked queries pertaining to our university database. These queries, articulated in Modern Standard Arabic (MSA), formed the core of the dataset, each accompanied by its corresponding SQL statement.

To verify the accuracy and consistency of these SQL statements, we populated our database with dummy data.

This simulation exercise allowed us to run each SQL query to ensure that its output was consistent with the intentions encapsulated in its corresponding MSA question. We went a step further and included the SQL execution output with each MSA-SQL pair in the dataset, earmarking it for future evaluation exercises.

Subsequent to the creation and validation of MSA-SQL pairs, we embarked on a dialect translation exercise to enrich the dataset with regional context. Leveraging the familiarity of our university students with the Palestinian Arabic dialect, we designed a simple web form to facilitate this task. Each form, presenting five MSA queries at a time, was shared with a large pool of our students, specifically about 400 of them filled the form.

The students were tasked to provide their translations for each MSA question in the form, bringing in their unique perspective in the form of variant translations. The web form allowed them to generate as many variant translations as they could for each question. Upon completing the set of five questions, they had the option to request another set by clicking on the ”Show Another 5 Questions” button.

Notably, the new set of questions was not randomly generated but rather intelligently selected to balance the number of variants for each question. The questions with fewer variants were prioritized to ensure a relatively equal number of variants for each question. This systematic approach led to the generation of 2,844 variants for the initial 70 MSA questions, all produced by around 400 students. The result was a richly varied dataset, encapsulating the linguistic nuances of a large group of users.

Having completed the generation of simple queries and their dialect translations, we applied the same steps to create complex queries. We carefully constructed about 50 complex queries, aligning them to our database schema and filling them with dummy data. While these complex queries were not used for training the model or included in the translation form, they added a valuable layer of depth to our dataset, and they can be used in the future work.

3.3. Model Selection and Fine-tuning

Our study adopted a sequential approach to refine our selected model, starting with a broad, general-purpose fine-tuning, followed by a more specific and tailored fine-tuning. This two-pronged strategy was driven by the desire to create a model well-versed in generic Arabic NL to SQL translations and capable of handling the subtleties of our specific university database and the Palestinian Arabic dialect.

3.3.1. Model Selection. For this task, we elected to utilize the AraT5 transformer, a language model known for its robustness in handling Arabic language processing tasks. AraT5, essentially an Arabic adaptation of Google's T5 transformer, has demonstrated impressive efficiency in translation tasks, making it an excellent fit for our study's objectives [5].

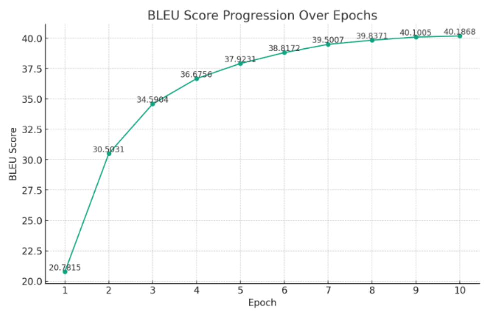


Figure 1. Progression of BLEU Score Over Ten Epochs During the General Fine-tuning Phase Using the WikiSQL Dataset

3.3.2. General Fine-tuning. Upon choosing the AraT5 transformer, we embarked on the initial phase of fine-tuning. During this stage, we adapted the model to our large translated WikiSQL dataset, which contained a plethora of Arabic NL - SQL pairs and general database scenarios.

Employing the sophisticated resources of the HuggingFace transformers library, we equipped our model with a learning rate of 1e-5, set the maximum target and source lengths to 512, and trained it for ten epochs. These steps empowered the model to grasp the fundamental structure of SQL queries and to establish a relationship between Arabic NL queries and SQL.

This training proved effective, with the model's BLEU score, a common metric for evaluating machine translation, steadily rising over the epochs. The score began at 20.7815 and, through successive iterations, improved to reach a peak of 40.1868. This metric revealed an encouraging performance, indicating a promising start to our model's fine-tuning journey.

3.3.3. Specific Fine-tuning. With the general fine-tuning completed, we proceeded to the second phase of our fine-tuning process. This step involved fine-tuning our pretrained AraT5 model on our specific university database dataset. This dataset contained Arabic NL - SQL pairs strictly related to our database, ensuring that the fine-tuning would calibrate the model to understand the peculiarities of our university database schema and the Palestinian Arabic dialect.

We assigned a higher learning rate for this phase, 8e-4, underlining the importance of this stage in the model's learning. This specific fine-tuning stage spanned 10 epochs, allowing ample time for the model to adapt to the intricacies of our specific dataset.

The translation and fine-tuning of the model were central tasks conducted via Google Colab PRO [2] and the HuggingFace transformers library [3]. With Google Colab PRO's high-RAM and A100 GPU, we automated the translation of 241,962 NL-SQL pairs from English to Arabic and carried out a two-step fine-tuning process on the model.

2. colab.research.google.com/signup
3. huggingface.co/docs/transformers

3.3.4. System Integration. With the fine-tuning of the model completed, we embarked on the final phase of our project - system integration. We aimed to incorporate the trained model into an interactive interface. The interface was implemented as a web page, with ReactJS [4] employed for the front-end development and Python Flask [5] for the back-end to seamlessly integrate the models, which were programmed in Python [6]. The database in use was MySQL [7], which was already in place.

The web page was designed with simplicity in mind, featuring only an input field where the user could input a Palestinian Dialect NL query. The system would then return the final answer retrieved from the database, along with the generated SQL statements.

In order to provide multiple potential outputs, we configured our seq2seq AraT5 model to generate three different SQL statements. This was achieved by setting the following attributes:

```
do_sample=True,
top_k=120,
top_p=0.95,
early_stopping=True,
num_return_sequences=3
```

Invalid SQL statements were not executed, but they were still presented to the user along with all other outputs. This provided a comprehensive range of possible interpretations for the input NL query, both valid and invalid.

4. Evaluation and Results

In this section, we delve into the performance of our model by exploring both evaluation metrics and practical results. The foundation of our assessment is the BLEU score, a common metric used in machine translation tasks.

4.1. Evaluation Metrics

We used the BLEU (Bilingual Evaluation Understudy) score as our primary metric for evaluating the performance of our model. The BLEU score is a popular metric in Natural Language Processing tasks, specifically in machine translation where it measures the quality of the translated text. The score ranges from 0 to 100, where a higher score indicates a better match with the reference translation [12].

However, it's worth mentioning that while the BLEU score is a good indicator for translation tasks, it may not be the perfect choice for evaluating translations to SQL. This is due to its inherent limitation: it relies on exact match n-gram comparisons, which might not fully capture the functional equivalence in SQL queries. Two differently worded SQL queries can often yield the same results, a nuance that the BLEU score might overlook [14].

4. www.react.dev
5. flask.palletsprojects.com
6. www.python.org
7. www.mysql.com

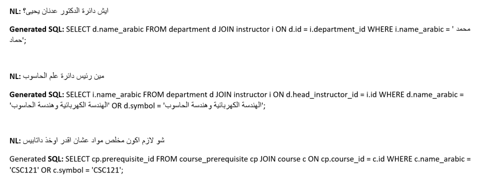
As for our general fine-tuning phase, the evaluation file showed that our model achieved a BLEU score of 40.1868 at the end of the 10th epoch. The checkpoint at this point was identified as the model's best performance.

In regards to the Turjuman model that we used for translation, according to their paper, it outperformed several baselines in 13 out of 20 test splits. It gained high BLEU scores, outperforming the S2SMT, mBART, and mT5 models with +8.07, +11.28, and +0.53 BLEU points on average [13].

4.2. Practical Results

Moving beyond metrics, we put our model to the test by generating some SQL statements from Arabic Palestinian Dialect NL and observed several key points:

1) **SQL Statement Structure:** The model generated well-structured SELECT statements in most cases, leading to a minimal amount of SQL execution errors.
2) **Handling Unknown Labels:** The model struggled with labels it was not trained on. For instance, if the model encounters an unfamiliar instructor's name, it attempts to replace it with the closest equivalent from its training data.



NL: ايش دائرة الدكتور عدنان يحيى؟

Generated SQL: SELECT d.name_arabic FROM department d JOIN instructor i ON d.id = i.department_id WHERE i.name_arabic = ' محمد حماد';

NL: مين رئيس دائرة علم الحاسوب

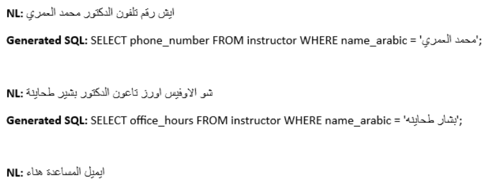
Generated SQL: SELECT i.name_arabic FROM department d JOIN instructor i ON d.head_instructor_id = i.id WHERE d.name_arabic = 'الهندسة الكهربائية وهندسة الحاسوب' OR d.symbol = 'الهندسة الكهربائية وهندسة الحاسوب';

NL: شو لازم اكون مخلص مواد عشان اقدر اوخذ داتابيس

Generated **SQL:** SELECT cp.prerequisite_id FROM course_prerequisite cp JOIN course c ON cp.course_id = c.id WHERE c.name_arabic = 'CSC121' OR c.symbol = 'CSC121';

Figure 2. Examples of the model handling unknown labels

3) **Simple Queries:** For very simple queries (those without JOIN operations), the model performed excellently.



NL: ايش رقم تلفون الدكتور محمد العمري

Generated SQL: SELECT phone_number FROM instructor WHERE name_arabic = 'محمد العمري';

NL: شو الاوفيس اورز تاعون الدكتور بشير طحاينة

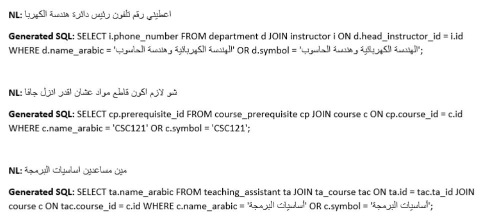
Generated SQL: SELECT office_hours FROM instructor WHERE name_arabic = 'بشار طحاينه';

NL: ايميل المساعدة هناء

Generated SQL: SELECT email FROM teaching_assistant WHERE name_arabic = 'هناء العمران';

Figure 3. Examples of the model translating simple queries

4) **Simple Queries with JOINs:** The model also performed well on simple queries involving JOIN operations, albeit with a few errors.



NL: اعطيني رقم تلفون رئيس دائرة هندسة الكهربا

Generated SQL: SELECT i.phone_number FROM department d JOIN instructor i ON d.head_instructor_id = i.id
WHERE d.name_arabic = 'الهندسة الكهربائية وهندسة الحاسوب' OR d.symbol = 'الهندسة الكهربائية وهندسة الحاسوب';

NL: شو لازم اكون قاطع مواد عشان اقدر انزل جافا

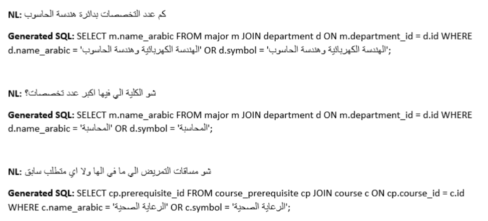
Generated SQL: SELECT cp.prerequisite_id FROM course_prerequisite cp JOIN course c ON cp.course_id = c.id
WHERE c.name_arabic = 'CSC121' OR c.symbol = 'CSC121';

NL: مين مساعدين اساسيات البرمجة

Generated SQL: SELECT ta.name_arabic FROM teaching_assistant ta JOIN ta_course tac ON ta.id = tac.ta_id JOIN
course c ON tac.course_id = c.id WHERE c.name_arabic = 'أساسيات البرمجة' OR c.symbol = 'أساسيات البرمجة';

Figure 4. Examples of the model translating simple queries with JOIN operations

5) **Complex Queries:** For complex queries (like that contains COUNT, GROUP BY, and other advanced SQL statements), even some that were included in the training data, the model's performance was not satisfactory. It occasionally generated outputs that were significantly different from the expected result.



NL: كم عدد التخصصات بدائرة هندسة الحاسوب

Generated SQL: SELECT m.name_arabic FROM major m JOIN department d ON m.department_id = d.id WHERE
d.name_arabic = 'الهندسة الكهربائية وهندسة الحاسوب' OR d.symbol = 'الهندسة الكهربائية وهندسة الحاسوب';

NL: شو الكلية الي فيها اكبر عدد تخصصات؟

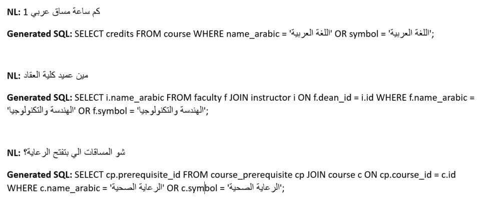
Generated SQL: SELECT m.name_arabic FROM major m JOIN department d ON m.department_id = d.id WHERE
d.name_arabic = 'المحاسبة' OR d.symbol = 'المحاسبة';

NL: شو مساقات التمريض الي ما في الها ولا اي متطلب سابق

Generated SQL: SELECT cp.prerequisite_id FROM course_prerequisite cp JOIN course c ON cp.course_id = c.id
WHERE c.name_arabic = 'الرعاية الصحية' OR c.symbol = 'الرعاية الصحية';

Figure 5. Examples of the model's performance on complex queries

6) **Label Variations:** A positive feature we noticed was the model's ability to manage variations in labels. As an illustration, if a user queries using a colloquial term for a specific college, the model successfully translates this into a SQL SELECT statement that searches for the formal name of the same college.



NL: كم ساعة مساق عربي 1

Generated SQL: SELECT credits FROM course WHERE name_arabic = 'اللغة العربية' OR symbol = 'اللغة العربية';

NL: مين عميد كلية العقاد

Generated SQL: SELECT i.name_arabic FROM faculty f JOIN instructor i ON f.dean_id = i.id WHERE f.name_arabic =
'الهندسة والتكنولوجيا' OR f.symbol = 'الهندسة والتكنولوجيا';

NL: شو المساقات الي بتفتح الرعاية؟

Generated SQL: SELECT cp.prerequisite_id FROM course_prerequisite cp JOIN course c ON cp.course_id = c.id
WHERE c.name_arabic = 'الرعاية الصحية' OR c.symbol = 'الرعاية الصحية';

Figure 6. Examples of the model managing label variations

It is important to note that the final fine-tuned model was trained only on the simple dataset and did not include the complex queries dataset. This could be a contributing factor to the limited performance on complex queries and provides an avenue for future improvement.

5. Further Work

While we have made significant strides with our model, there remains substantial room for enhancement and exploration. The following potential areas of further investigation and improvement could elevate our system's performance:

- **Preprocessing and Data Cleaning:** Improve the preprocessing and cleaning of our dataset. Enhanced cleaning and verification processes could ensure a more robust, reliable dataset for training and evaluation.
- **Model Experimentation:** Exploration of alternative large language models (LLMs) that may offer superior efficiency. Additionally, constructing a model with an Arabic Encoder and English Decoder [15] could potentially deliver better translation results.
- **POS and NER Integration:** Investigation into the application of Part-of-Speech (POS) tagging [16] and Named Entity Recognition (NER) [17] could aid the translation task. A rule-based or hybrid approach could potentially enhance the translation quality.
- **Intermediate Format:** Adopting an intermediate format that can be easily translated to SQL using defined rules. This could ensure the generation of valid and executable SQL statements in every instance [18].
- **SQL Value Management:** Improving the management of SQL values—specifically the values being queried—. Techniques such as POS tagging and NER could be employed to detect values from natural language queries.
- **Table and Field Identification:** Using POS and NER to identify table names and fields before inputting them into the model. This could potentially increase the accuracy of our SQL generation.
- **Complex Dataset Utilization:** Utilizing our more complex dataset for fine-tuning and evaluation, which could help in improving the model's performance on intricate queries.
- **Translation Approaches:** Exploring alternative translation methodologies, such as the Google Translate API [8], for translating the general dataset (WikiSQL) to Arabic, which could potentially offer higher-quality translations.
- **Hyperparameter Tuning:** Trying different hyperparameters could further optimize the performance of our model [19].

Each of these steps represents a potential improvement to our system. Through systematic exploration and refinement, we can continue to enhance our model's performance and effectiveness.

6. Conclusion

In conclusion, this research demonstrated the feasibility and promise of Arabic Natural Language to SQL translation, with particular emphasis on the Palestinian dialect and a university database context. Our study utilized the AraT5 transformer model, employing a two-step sequential fine-tuning process.

8. cloud.google.com/translate

The results of our research, while promising, highlighted an area that could be improved upon in future work. Specifically, it was found that the model struggled with instances where specific values were directly used during training. For example, if we trained the model on a query where a particular course name was given, the model tended to overfit to that specific course name.

A potential solution could be the use of more general labels during training. In this scenario, specific values such as course names or professor names would be replaced with placeholders during the training phase. The model would then be trained to translate NL queries to SQL, using these placeholders instead of the specific values. Once the model generates the SQL statement with the placeholder, it can be replaced with the original specific value from the user's query. This approach could enable the model to generalize better across different queries, avoiding overfitting to specific training instances.

Despite these challenges, our system demonstrated a strong ability to handle simple queries and manage label variations, attesting to the potential of such models in the field of NL to SQL translation.

This study serves as an encouraging stepping stone towards more efficient and effective Natural Language Processing models, particularly in the context of underrepresented languages such as Arabic. The findings pave the way for future research to improve upon these initial results and contribute to the broader development of NL to SQL translation systems.

References

[1] C. J. Date, *A Guide to the SQL Standard*. Addison-Wesley Longman Publishing Co., Inc., 1989.

[2] N. Nihalani, S. Silakari, and M. Motwani, "Natural language interface for database: a brief review," *International Journal of Computer Science Issues (IJCSI)*, vol. 8, no. 2, p. 600, 2011.

[3] A. Shoufan and S. Alameri, "Natural language processing for dialectical arabic: A survey," in *Proceedings of the second workshop on Arabic natural language processing*, pp. 36–48, 2015.

[4] I. Guellil, H. Saâdane, F. Azouaou, B. Gueni, and D. Nouvel, "Arabic natural language processing: An overview," *Journal of King Saud University-Computer and Information Sciences*, vol. 33, no. 5, pp. 497–507, 2021.

[5] E. M. B. Nagoudi, A. Elmadany, and M. Abdul-Mageed, "Arat5: Text-to-text transformers for arabic language generation," *arXiv preprint arXiv:2109.12068*, 2021.

[6] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[7] T. Tan, Z. Li, H. Liu, F. G. Zanjani, Q. Ouyang, Y. Tang, Z. Hu, and Q. Li, "Optimize transfer learning for lung diseases in bronchoscopy using a new concept: sequential fine-tuning," *IEEE journal of translational engineering in health and medicine*, vol. 6, pp. 1–8, 2018.

[8] K. Chowdhary and K. Chowdhary, "Natural language processing," *Fundamentals of artificial intelligence*, pp. 603–649, 2020.

[9] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch, "Natural language interfaces to databases–an introduction," *Natural language engineering*, vol. 1, no. 1, pp. 29–81, 1995.

[10] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *CoRR*, vol. abs/1709.00103, 2017.

[11] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[12] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

[13] E. M. B. Nagoudi, A. Elmadany, and M. Abdul-Mageed, "Turjuman: A public toolkit for neural arabic machine translation," in *Proceedings of the 5th Workshop on Open-Source Arabic Corpora and Processing Tools (OSACT5)*, (Marseille, France), European Language Resource Association, June 2022.

[14] S. Gala, "Translating english to sql," *Created Dec*, vol. 8, 2005.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[16] A. Voutilainen, *Part-of-speech tagging*, vol. 219. The Oxford handbook of computational linguistics, 2003.

[17] B. Mohit, "Named entity recognition," in *Natural language processing of semitic languages*, pp. 221–245, Springer, 2014.

[18] U. Brunner and K. Stockinger, "Valuenet: A natural language-to-sql system that learns from database information," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 2177–2182, IEEE, 2021.

[19] R. Bardenet, M. Brendel, B. Kégl, and M. Sebag, "Collaborative hyperparameter tuning," in *International conference on machine learning*, pp. 199–207, PMLR, 2013.