T.R.

GEBZE TECHNICAL UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

TURKISH TEXT TO SQL QUERY CONVERSION

GÜLNİHAL AKDEM

SUPERVISOR
YRD.DOÇ.DR. BURCU YILMAZ

GEBZE
2024

T.R.
GEBZE TECHNICAL UNIVERSITY
FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT

TURKISH TEXT TO SQL QUERY CONVERSION

GÜLNİHAL AKDEM

SUPERVISOR
YRD.DOÇ.DR. BURCU YILMAZ

2024
GEBZE

This study has been accepted as an Undergraduate Graduation Project in the Department of Computer Engineering on 16/03/2024 by the following jury.

JURY

Member
(Supervisor) : Yrd.Doç.Dr. BURCU YILMAZ

Member : Doç.Dr. HABİL KALKAN

ABSTRACT

Recently, natural language processing (NLP) has advanced significantly; however, converting natural language into structured queries remains a challenge.

The goal of this project is to generate SQL queries for Turkish data using deep learning methods.In the study of converting Turkish sentences to SQL queries, the representation of a Turkish sentence in its SQL query form is determined.

This project can be applied in various domains such as business intelligence, e-commerce, and finance to enable users to retrieve data from databases using natural language queries. By converting natural language to SQL, it enhances data accessibility and simplifies the process of generating complex queries for non-technical users.

In this study, a dataset was created using a company's database table. The generated dataset was used in the deep learning method.

ÖZET

Son zamanlarda, doğal dil işleme (NLP) önemli ölçüde ilerlemiştir; Ancak, doğal dili yapılandırılmış sorgulara dönüştürmek zor olmaya devam ediyor.

Projenin amacı derin öğrenme metodu kullanılarak Türkçe veriler için SQL Query karşılığını oluşturmaya dayanır. Türkçe cümleleyi SQL Query'ye çevirme çalışmasında Türkçe cümlenin SQL Query'sindeki gösterimi belirlenmiş olur.

Bu proje, iş zekası, e-ticaret ve finans gibi çeşitli alanlarda kullanıcıların doğal dil sorguları kullanarak veritabanlarından veri almasını sağlayarak uygulanabilir. Doğal dili SQL'ye dönüştürerek, veri erişilebilirliğini artırır ve karmaşık sorguların oluşturulma sürecini teknik olmayan kullanıcılar için basitleştirir.

Bu çalışmada bir şirket'in veritabanından tablosu kullanılarak veriseti oluşturuldu. Oluşturulan veriseti derin öğrenme yönteminde kullanıldı.

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol or Abbreviation	:	**Explanation**
ChequeNo (Çek numarası)	:	First column in Cheque table
persontype (Kişi tipi)	:	Second column in Cheque table
ChequeClearanceStatusCode (Tahsil durum kodu)	:	Third column in Cheque table
ChequeAmount(Çek miktarı/Çek tutarı)	:	Fourth column in Cheque table
ChequeMaturityDate(Çek vade tarihi)	:	Fifth column in Cheque table

CONTENTS

LIST OF FIGURES

1. DATA PREPARATION

First, a custom dataset containing Turkish text and SQL query sentence pairs needs to be created. When loading the dataset, it is also important to ensure that any empty rows are removed to maintain data integrity.

Subsequently, the dataset is shuffled and then divided into training and test sets to facilitate the evaluation of the model's performance during the training process.

```
# SPLIT TRAIN / TEST SETS
train_df, test_df = train_test_split(df3_Shuffled, test_size=0.2, random_state=42)
```

Figure 1.1: Splitting the dataset

1.1. Database

	ChequeNo	persontype	ChequeClearanceStatusCode	ChequeAmount	ChequeMaturityDate
1	4718549	1	TAH	65000	2021-04-26 00:00:00.000
2	6500127	1	TAH	12335,6	2019-08-26 00:00:00.000
3	2726489	1	TAH	7752	2018-02-12 00:00:00.000
4	8033271	1	TAH	25000	2021-06-02 00:00:00.000
5	9102528	1	TAH	15000	2020-09-21 00:00:00.000
6	5929611	1	TAH	28414,4	2020-03-12 00:00:00.000
7	24331	1	TAH	80700,95	2021-04-12 00:00:00.000
8	18926	1	TAH	74079	2019-04-01 00:00:00.000
9	119868	1	TAH	4000	2018-07-10 00:00:00.000
10	9119405	1	TAH	26000	2020-04-16 00:00:00.000

Figure 1.2: Cheque table in Bank database

In this project, a database named 'Bank' is used. It contains a table called 'Cheque,' which has the following attributes: ChequeNo, ChequeClearanceStatus-Code, ChequeAmount, PersonType, and ChequeMaturityDate. After generating the SQL query, it is executed, and the data are retrieved from this database. To establish the connection, SQLAlchemy and its engine is used.

It is also necessary to understand a few banking terminologies. For example, the PersonType attribute: if the value is 1, it denotes 'tüzel/ticari kişi' (legal/commercial

entity), and if the value is 0, it denotes ’gerçek kişi’ (individual). For the ChequeClearanceStatusCode attribute: if the value is ’TAH’, it means ’ödenmiş çek’ (paid cheque), and if the value is ’KR’, it means ’ödenmemiş’ and ’karşılıksız’ (unpaid and insufficient funds).

1.2. Dataset

question	sql
Çeklerin hepsini getir	SELECT * FROM Cheque
Çek numarası (\d+) ile (\d+) arasında ve miktarı (\d+) TL'den küçük çekleri getir	SELECT * FROM Cheque WHERE ChequeNo BETWEEN \1 AND \2 AND ChequeAmount < \3;
Çek numarası (\d+) ile (\d+) arasında ve miktarı (\d+) TL'a eşit çekleri göster	SELECT * FROM Cheque WHERE ChequeNo BETWEEN \1 AND \2 AND ChequeAmount = \3;
Çek numarası (\d+) ile (\d+) arasında ve tutarı (\d+) TL'den büyük çekleri getir	SELECT * FROM Cheque WHERE ChequeNo BETWEEN \1 AND \2 AND ChequeAmount > \3;
Gerçek kişilerin yazdığı çekleri listele	SELECT * FROM Cheque WHERE persontype = 0;
Tüzel kişilerin yazdığı çekleri göster	SELECT * FROM Cheque WHERE persontype = 1;
Ödenmiş çeklerin çek miktarlarını getir	SELECT ChequeAmount FROM Cheque WHERE ChequeClearanceStatusCode = 'TAH';
Çek vadesi (\d{4}-\d{2}-\d{2}) tarihine eşit ve tutarı (\d+) TL'a eşit çeklerin sayısını getir	SELECT COUNT(*) FROM Cheque WHERE ChequeMaturityDate = \1 AND ChequeAmount = \2;
Çek vadesi (\d{4}-\d{2}-\d{2}) tarihinden önce ve tutarı (\d+) TL'den büyük çeklerin sayısını göster	SELECT COUNT(*) FROM Cheque WHERE ChequeMaturityDate < \1 AND ChequeAmount > \2;
Çek numaralarını ve kişi tiplerini göster	SELECT ChequeNo, persontype FROM Cheque;

Figure 1.3: Dataset

This dataset is custom and was created by me. It contains various types of Turkish text along with their corresponding SQL queries. The dataset is used to fine-tuned the model and serves as a cornerstone for deep learning projects. Via creating Data Loader objects for both the training and tests sets to efficiently load batches of data during model training.

2. METHOD

We start by setting up the pre-trained BERT model, 'bert-base-multilingual-case', to prepare it for adapting to our translation task. Then, we establish an optimizer and a learning rate scheduler to guide the training process. We opt for AdamW as our optimizer, incorporating linear warm-up for better training stability.

Next, we fine-tune the BERT model through multiple training iterations, continually adjusting its parameters to improve its translation capabilities. Throughout this training loop, we use MLflow to keep track of the training progress. Specifically, we monitor and log the average training loss after each training epoch. Additionally, we create a visual representation of the training loss graph to gain insights into how the model learns over time.

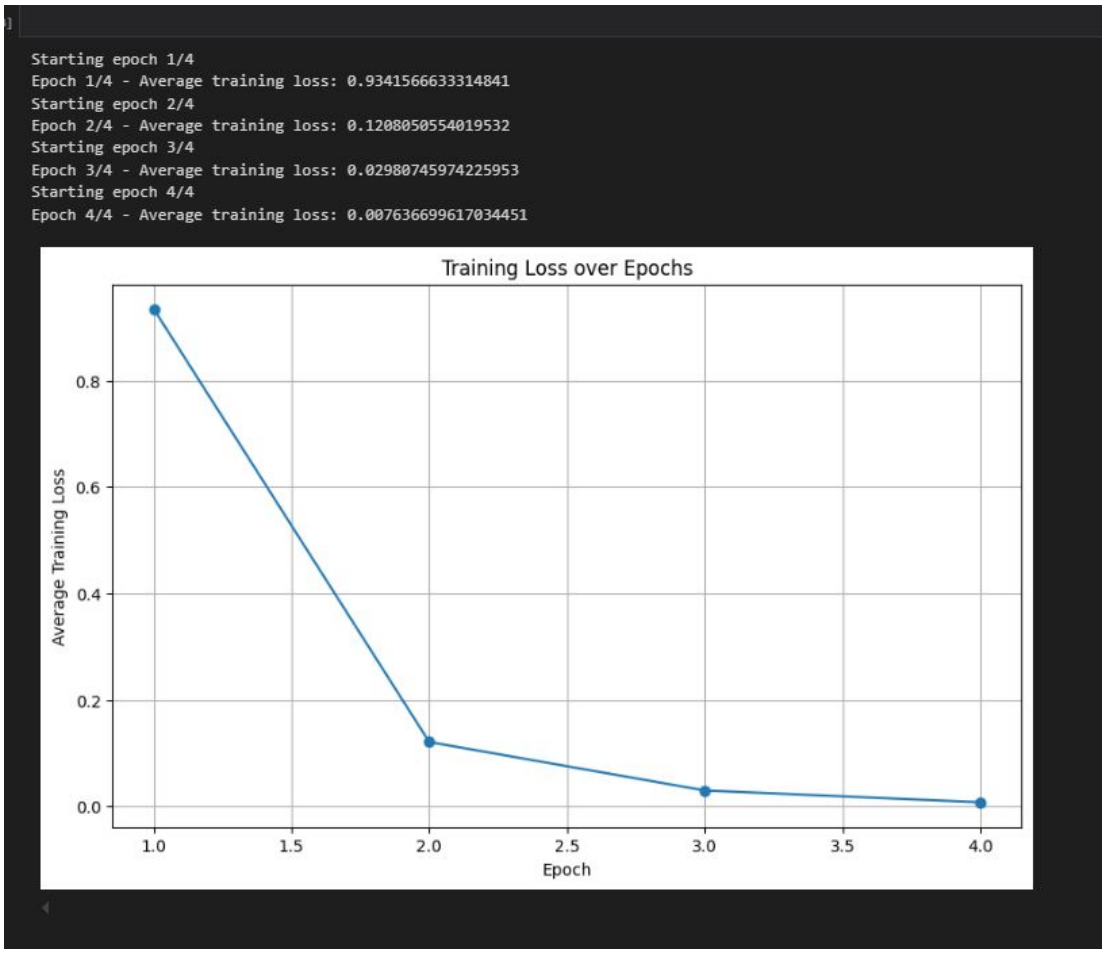

Figure 2.1: Training loss over Epochs

2.1. BLEU SCORE

After fine-tuning the model, it's essential to assess its performance on new data by calculating the BLEU score(machine translation). Bleu score measures the similarity between the predicted translation and the reference translation. A higher BLEU score indicates better translation performance. In this part, Evaluate the model on test data.

This project bears similarity to machine translation tasks, as it involves converting Turkish text into SQL queries. Therefore, the BLEU Score was utilized.

Figure 2.2: BLEU Score over Test set

2.2. Inference as a result of experience

In some cases, I initially began with 6 epochs, but found it unnecessary, hence updated it to 8. However, altering other parameters did not yield satisfactory results. Consequently, expanding the dataset resolved the issue. Nonetheless, upon increasing the dataset size, overfitting became apparent, as indicated by the training loss graph. Therefore, for the final epoch, it will be modified to 4.

After some experimentation, I found that starting with a learning rate of 2.5e-5 yielded good results. However, as I made changes to the number of epochs and dataset size, I tried adjusting the learning rate accordingly. Ultimately, after testing various values, I determined that a learning rate of 2.5e-4 was optimal for the final setup.

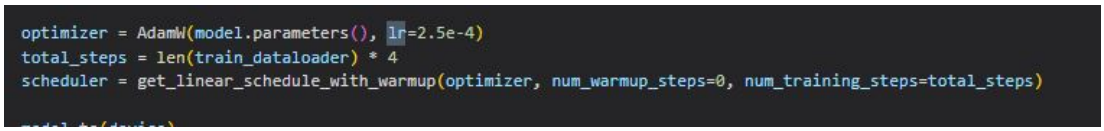

Figure 2.3: Learning rate

Another crucial parameter is the batch size. Initially, starting with a batch size of 16, I experimented with larger batch sizes but found that they did not yield satisfactory results. Consequently, I attempted decreasing the batch size, ultimately settling on 8, which led to an improvement in both the training loss and score.

```
# Create data loaders
batch_size = 8
train_dataset = TensorDataset(train_input_ids, train_attention_masks, train_target_ids)
train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_dataset = TensorDataset(test_input_ids, test_attention_masks, test_target_ids)
test_dataloader = DataLoader(test_dataset, batch_size=batch_size)
display(train_dataset)
```

Figure 2.4: Batch size

2.3. User Interface

In the local environment, an interface is created for user interaction using Tkinter. The interface prompts the user to input Turkish text, which is then processed to generate a result. Subsequently, the generated result is executed using the engine, and the SQL query result is retrieved from the Bank database.

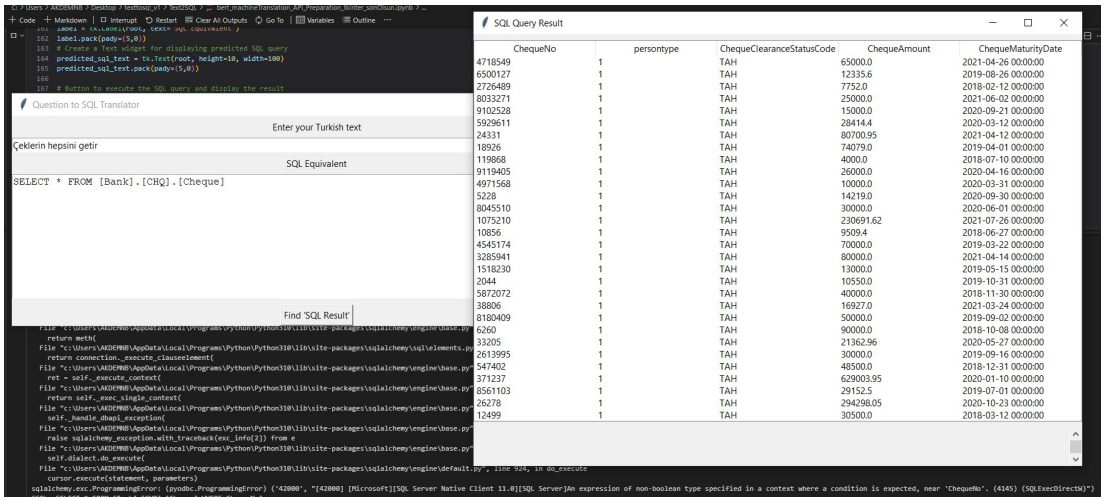

Figure 2.5: User Interface

3. CONCLUSIONS

This project initially started with experimentation using the T5 model, but it was observed that it was not efficient for Turkish text. Upon realizing that BERT was more suitable, experimentation shifted to BERT. A small trial was conducted with Helsinki to translate Turkish to English and then convert it into SQL grammar. However, due to the low results obtained, the focus shifted to BERT.

Subsequently, comparisons were made by increasing the epochs, but it was observed to indicate overfitting, hence the epochs were updated to 4. Batch size started at 16, then tried 32 and 24, but as they did not yield satisfactory results, it was reduced to 8. As the result was better than with a batch size of 16, the final update was made to 8.

Through trial and error with learning rates, the optimum was found to be 2.5e-4. The max seq length started at 128, then compared with 256 and 512. The best result was found at 256.

For the BLEU score calculation, the SmoothingFunction's method 4 was used to reduce the zero-sensitivity issue when certain n-grams did not occur in the reference translation.

BIBLIOGRAPHY

TRAILER VIDEO : https://www.youtube.com/watch?v=ngJgIUWX40I

1) https://huggingface.co/models?search=text

2) https://mccormickml.com/2019/07/22/BERT-fine-tuning/

3) https://www.scaler.com/topics/nlp/masked-language-model-explained/

4) https://www.bestgentools.ai/tools/ai2sql

5) https://blog.aiensured.com/untitled-7/

6) https://leimao.github.io/blog/BLEU-Score/