# TURKISH QUESTION GENERATION MODEL

**Senior Design Project II**

**Alp Gokcek** **Erdal Sidal Dogan**

**2021**

**MEF UNIVERSITY**
**FACULTY OF ENGINEERING**

**DEPARTMENT OF COMPUTER ENGINEERING**

# TURKISH QUESTION GENERATION MODEL

**Senior Design Project II**

**Alp Gokcek** **Erdal Sidal Dogan**

**Advisor: Asst. Prof. Seniz Demir**

**2021**

**MEF UNIVERSITY**
**FACULTY OF ENGINEERING**

**DEPARTMENT OF COMPUTER ENGINEERING**

| | |
|---|---|
| Project Title | : TURKISH QUESTION GENERATION MODEL |
| Student(s) Name | : Alp Gokcek, Erdal Sidal Dogan |
| Date | : 05/06/2021 |

I hereby state that the design project prepared by Alp Gokcek, Erdal Sidal Dogan has been completed under my supervision. I accept this work as a "Senior Design Project".

05/06/2021
Asst. Prof. Seniz Demir

I hereby state that I have examined this senior design project by Alp Gokcek, Erdal Sidal Dogan. This work is acceptable as a "Senior Design Project".

05/06/2021
Prof. Muhittin Gokmen
Head of the Department of
Computer Engineering

## ACADEMIC HONESTY PLEDGE

In keeping with MEF University Student Code of Conduct, I pledge that this work is my own and that I have not received inappropriate assistance in its preparation. I further declare that all resources are explicitly cited.

| NAME | DATE | SIGNATURE |
|---|---|---|
| Alp Gokcek | 05/06/2021 | |
| Erdal Sidal Dogan | 05/06/2021 | |

# ABSTRACT

TURKISH QUESTION GENERATION MODEL

Alp Gokcek Erdal Sidal Dogan

MEF UNIVERSITY
Faculty of Engineering
Department of Computer Engineering

Advisor: Asst. Prof. Seniz Demir

JUNE, 2021

Question Generation is a Deep Learning model that given a paragraph, passage or an entity in Turkish, produces the possible questions that can be answered solely by the given content to the system. Question Generation (QG) Systems are capable of generating various logical questions from the given text input. QG Systems are prevalent in several computer applications such as chatbots, automated grading systems etc.

In this project, our aim is to train a Deep Learning (DL) model for question generation and assess its performance according to evaluation metrics. We find this task worth tackling since there are not many examples of comprehensive studies on the topic and the outcomes of this project will empower the development of more capable-than-ever Question Answering Systems for the Turkish Language. QG models require extensively large training data, thus, in the last semester we have developed the initial dataset for the development of the QG model. This dataset is the largest Turkish Question-Answer dataset currently available. In this project, our aim is to train different language models for generation of questions and assess their performances as well as improving the diversity of our dataset.

***Keywords:*** Artificial Intelligence, Natural Language Processing, Deep Learning, Question Generation

# ÖZET

TÜRKÇE SORU OLUŞTURMA MODELİ

Alp Gokcek Erdal Sidal Dogan

MEF ÜNİVERSİTESİ
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Tez Danışmanı: Dr. Şeniz Demir

HAZIRAN, 2021

Soru Üretme (SÜ) Modeli, Türkçe bir paragraf, pasaj veya varlıktan, sadece verilen içerikle cevaplanabilecek olası soruları sisteme üreten bir Derin Öğrenme (DÖ) modelidir. SÜ Sistemler, verilen metin girdisinden çeşitli mantıksal sorular üretebilir. SÜ Sistemler, sohbet robotları, otomatik derecelendirme sistemleri vb. gibi çeşitli bilgisayar uygulamalarında yaygındır.

Bu projede amacımız, soru oluşturma için bir DÖ modelleri eğitmek ve performansını değerlendirmektir. Konuyla ilgili çok fazla kapsamlı çalışma örneği bulunmadığından ve bu projenin çıktıları Türk Dili için her zamankinden daha yetenekli Soru Cevaplama Sistemlerinin geliştirilmesini güçlendireceğinden, bu görevi çözmeye değer buluyoruz.

SÜ modelleri çok büyük eğitim verileri gerektirir, bu nedenle son dönemde SÜ modelinin geliştirilmesi için ilk veri kümesini geliştirdik. Bu veri kümesi, şu anda mevcut olan en büyük Türkçe Soru-Cevap veri kümesidir. Bu projede amacımız, soruların oluşturulması ve performanslarının değerlendirilmesi ve veri setimizin çeşitliliğinin iyileştirilmesi için farklı dil modelleri yetiştirmektir.

***Anahtar Kelimeler:*** Yapay Zeka, Doğal Dil İşleme, Derin Öğrenme, Soru Üretme

# Table of Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | | |
|---|---|---|
| **DL** | Deep Learning | v, 30 |
| **DÖ** | Derin Öğrenme | vi |
| **QG** | Question Generation | v, 7, 20 |
| **SÜ** | Soru Üretme | vi |

# 1. Introduction

Artificial Intelligence is one of the promising research fields in computer science. Scientists and researchers are developing novel or more efficient ways to teach computers how to achieve humanly-kinds of tasks.

Over time, developments in this area yield to a whole new class of possibilities and problems requiring brand new methodologies to reach the solution. With the emergence of Machine Learning, computers became to be able to parse data, learn from the data and apply their findings from it in a way that they are told to.

In recent years, researchers came up with a more niche way of handling Machine Learning (ML) tasks, creating a structure that imitates the Neurons in our brains. This new subclass of ML is named Deep Learning (DL). Compared to ML, it is capable of handling a much more complicated task due to its multi-layered structure.

Developments in this area enabled researches from numerous fields to move their research a step further. Today, Deep Learning is ubiquitous in computer applications. It is utilized for Computer Vision, Natural Language Processing, Biotechnology and much more. In this project, methods and techniques that are results of the studies conducted in the research area of Natural Language Processing are used heavily along with various Machine Learning and Deep Learning models. Consequently, the Turkish Question Generation is invigorated by the state-of-art Deep Learning and Language Models.

Turkish Question Generation Model aims to generate natural language questions from a given content, such as paragraphs, where the questions can be answered solely by the content. As the authors, we find this task worth tackling since there are not many examples of comprehensive studies on the topic and the outcomes of this project will empower the development of more-capable-than-ever Question Answering Systems for the Turkish Language.

## 1.1. Motivation

As humans, we are continually learning, adapting to changes or make decisions on an everyday basis. Even if we do not realize, we usually are dependent on other humans and their knowledge and expertise even for most trivial things we do in our daily lives.

For instance, a student feels in need for an instructor or others than can address his/her questions about a topic, an individual may be in a hurry to find out whether he/she can perform SWIFT transaction from his bank, or simply a person can ask about the weather forecast to the voice assistant on a smartphone. All of these examples are having a common point that they are all based on Question Answering (QA) Systems.

As we have discussed, QA Systems are getting more and more space in our lives and presents a wide range of opportunities. With these systems, we are able to get answers to our question at no time, without any human interaction at all. This feature enables humans to reach to even the most isolated information in the context a matter of time upon request.

Unfortunately, given that almost every human spoken language has a specific set of rules, grammar and vocabulary, QA systems need to be developed for each language separately. Development of such systems are not the easiest thing to do, first of all, even if the developer(s) has all the competency in the technical skills required, there is a need for clear, labelled, reliable and excessively large dataset. The dataset must contain questions and possible answers to that question. Creating such dataset manually is almost unattainable; therefore, there is a need for a system that can automate this process. Question Generation systems are perfect for meeting such need. The output of the QG system can be used as the dataset for development.

On top of that, QG systems can be utilized in the education field also. Given a passage, the instructor may want to create various questions automatically without putting any effort. The same application can be used by a student for practising the learnings from a section.

## 1.2. Broad Impact

### 1.2.1. Global Impact of the Solution

Online courses are gaining getting more and more popular each and every year. While their high-quality content is available on the web, for those who cannot understand English or mainstream languages of a specific subject, it is harder to keep up with the recent developments in the area. Up to this point, MOOC platforms offer translation and subtitles for such users. The Question Generation systems, along with Question Answering systems, might create an artificial interactive environment between the instructor and the student. Consequently, helping the information to spread around the world.

### 1.2.2. Economic Impact of the Solution

This project enables us to move faster to the point where we can automate almost every Question-Answer process where the data is assumed to be available to a computer system; there is not an ambiguity amongst possible answers, and the answers are static and clearly defined or quantitative.

Initial assumption would be that such a system would decrease human dependency and reduce the number of jobs consequently. However, given that the same approach is applicable to almost every new technology/product emerges in the market, it is shown that they also create new opportunities.

### 1.2.3. Environmental Impact of the Solution

Since the project will be software-based, no significant effect on the environment will be observed. Of course, one can discuss keeping the servers on, utilizing their resources to the maximum level continually will increase the total electricity consumption.

### 1.2.4. Social Impacts of the Solution

This project has many social impacts, though not directly. With the solution we developed, strengthening the question-answering model carried out in connection with this project, we will aid the learners' knowledge acquisition process.

### 1.2.5. Legal Issues Related to the Project

Such as any system that is designed to present information to the user, Question Answering systems, which our project Question Generation system will yield to development of, requires the necessary information for answering the question. Under these circumstances, the source of this information is critically important. Consent of the creator must be obtained before developing such a system.

The system must also not give inaccurate information to the user, since they may depend on the system on a critical task whose fault cannot be tolerated.

# 2. Project Definition and Planning

Turkish Question Generation Model is a neural network-based model capable of generating questions from a given piece of text in the Turkish language.

## 2.1. Project Definition

The project's main feature will be accepting input and creating possible questions from that input while assigning probability scores to each generated question. Questions are delivered in easily parsable file format such as JSON, CSV etc.

A GUI[1] will be designed to demonstrate the capabilities and enable majority of people to use it without any technical knowledge. Also, it will be presented as CLI[2] and *Python Module*[3] servers for public use, if allowed.

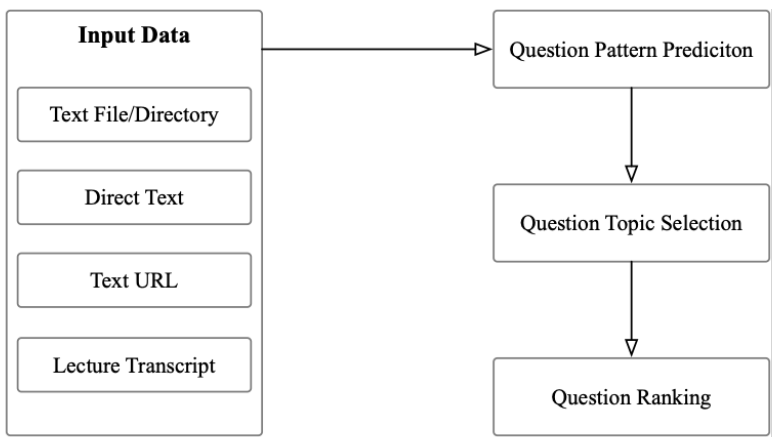



**Figure 1** – Question Generation system schema

[1]Graphical User Interface
[2]Command Line Interface
[3]An importable Python file that contains the functions for incorporating Turkish Question Generation for further research.

### 2.1.1. Dataset

There are lots of publicly available datasets for natural language processing tasks. However, for this project, our primary constraint was to build our system on top of a Turkish Dataset. Some other constraints we had was that the data must consist of correctly structured sentences in grammatical aspect, it must be labeled, come from a reliable source with the consent of the owner(s) and must be very large, given that larger dataset yields better models.

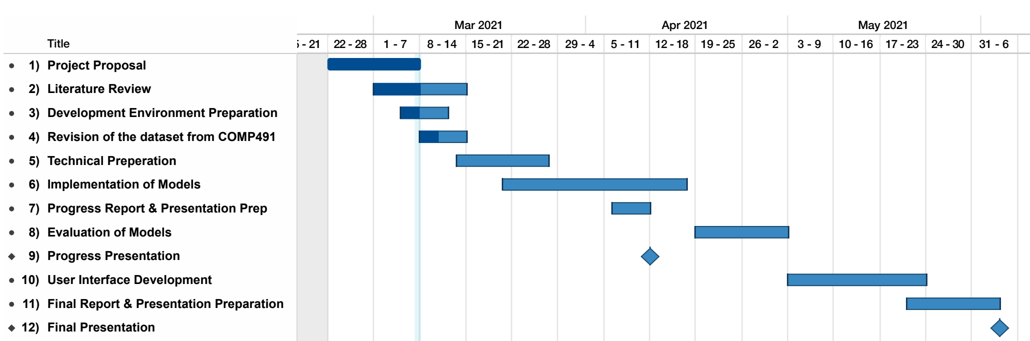
### 2.1.2. Programming Languages and Frameworks

This project is developed with *Python 3*. It has chosen because it is relatively simple and allows us to focus on the project rather than computational concerns, it has a vast community for troubleshooting, decreasing the chances of having an issue about the language.

Furthermore, Python is adapted by the AI researchers, and there are numerous resources, frameworks and libraries that are developed for Python in which we utilize heavily. We have not limited ourselves to a specific framework or library for the development of the models. Also, there are other libraries that are developed for scientific computing and have been used in this project such as *pandas, NumPy*, etc.

## 2.2. Project Planning

**Table 2** – Project plan for 14 weeks



### 2.2.1. Aim of the Project

Turkish Question Generation Model aims to fine-tune and train various state-of-art question generation models that are developed for other languages. Given a paragraph, passage or an entity in Turkish, it presents the possible questions that can be answered solely by the given content to the system.

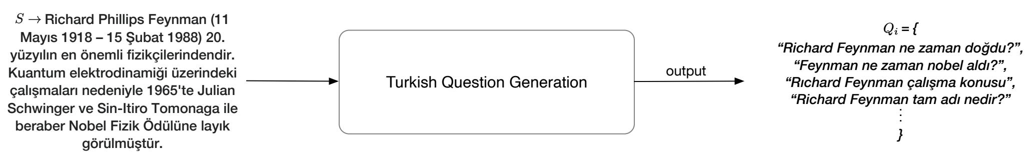



**Figure 2** – Question Generation Model diagram

QG models require extensively large training data. In this project, we have developed the datasets for the development of the QG model. Using our custom dataset, our aim is to compare performances of various models that we trained for Turkish Question Generation.

### 2.2.2. Project Coverage

In the previous semester, construction of a data set for the training of QG Model has been completed. Dataset consists of data which retrieved from the Turkish Wikipedia.

In this semester, we train several question generation models and fine tune if necessary for Turkish language with the dataset we had constructed. Later on, we aim to compare them regarding to common metrics such as BLEU & ROGUE [4] and present the results in detail.

The project's outcome will be a usable interface that users can pass paragraphs as input and retrieve the questions that has been generated automatically. Furthermore, a CLI[5] will be provided for other researchers to utilize in their workflows.

[4]See Section 2.2.4.
[5]Command Line Interface

### 2.2.3. Use Cases

Use case diagram of our project could be found in Figure 3. We have only one type of user at the moment. User will give a context paragraph to the system; then the system will return the questions generated from this context paragraph.

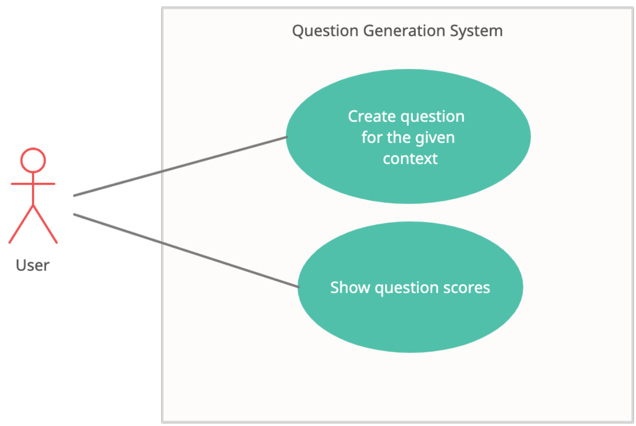



**Figure 3** – Use case diagram of our QG system

### 2.2.4. Success Criteria

A couple of metrics will evaluate the outcomes of the project. Given that the outputs will be natural language instances, existing methods and algorithms specifically designed to evaluate generated natural language examples will be utilized. There are multiple algorithms and methods available, which consider different aspects of the outcome. Some of them are:

- BLEU
- ROUGE
- METEOR

Each of the enumerated methods takes different aspects into account, while *BLEU* score is calculated with best match length is prominent, *ROGUE* is based on recall score. They all have their tradeoffs, and neither of them is merely a strong indicator of success.

### 2.2.5. Project Time and Resource Estimation

The project is estimated to take 16 weeks for two undergraduate students that are novices to the topic. While estimating, course load and other external factors are taken into account. The detailed timeline can be seen from Table 1 on the previous page.

Assuming that each member spends 8 hours a week working on this project's issues, it makes approximately 128 hours per person, 256 hours spent in total. Also, it has been mentioned that computing resources will be required for the project. If we use cloud computing providers (AWS for this particular example) rates for this estimation, we observe that the hourly price is  $2 for a server that we need to run. Considering that these are very intensive applications, it might be the cases that the server will be up and making computations for long hours, 20 hours a week. This estimation sums up to $600 in server running costs and 256 men hours. Notice that no access to books, online courses or any other learning material has not been included.

### 2.2.6. Solution Strategies and Applicable Methods

For the project, we decided to use off-the-shelf language models by fine-tuning them according to our data. We could have tried to achieve it without using such advanced models; however, there would be the risk of not completing the project in time or a flawed model could have been produced.

### 2.2.7. Risk Analysis

One of the significant risks that may occur is that the model gives questions that are not directly related to given inputs or contain grammatical/structural errors. There are not any risks.

### 2.2.8. Tools Needed

The project requires the development of Machine and Deep Learning Models from scratch. Due to wide selection of libraries for this purpose, the community behind it, and the de-facto standard of the AI research and scientific development, it has been decided that *Python Programming Language* will be used for development and implementation during this project.

Specifically, for the model development, an up-to-date, state-of-art library named *PyTorch* will be used extensively. Also, the development of such models and processing large amounts of data requires computing resources. Therefore, servers configured with GPU optimization in mind will be required for a faster development process.

# 3. Theoretical Background

## 3.1. Literature Survey

In this section, a survey of the literature from multiple sources could be found.

### 3.1.1. Question Answering

Question Answering (QA) automatically presents the answers to the question that users ask without any human interaction. In a QA model, it is expected that the system has access to the necessary information to answer the question.
To develop such a model, we need extensive data consisting of Questions and their Answers, a duple that we will denote by <Q, A>. Unfortunately, labelling this kind of data manually or creating answers from scratch for an Artificial Intelligence based solution is not the most efficient. They address these problems by;

1. Large scale high-quality dataset from Community-QA websites such as Yahoo, Quora is obtained since they provide large scale QA pairs generated by real users.

2. Two ways of accomplishing such task is implemented and compared. One is a retrieval-based method using Convolutional Neural Networks(CNN) and other is a generation-based method using Recurrent Neural Network (RNN).

3. Outcomes of the QG model is integrated with end-to-end QA task. It is evaluated on three state-of-art datasets, SQuAD, MS MARCO, and WikiQA. Results show that generated questions can improve QA quality on all these three datasets.

### 3.1.2. Question Generation

Research engineers from Microsoft Research proposed extracting question from a given piece of text. Duan et al. (2017) While they used NLP methodologies and Neural Networks, they did not include the semantics of words in their paper.

QG Engine is consist of four components:

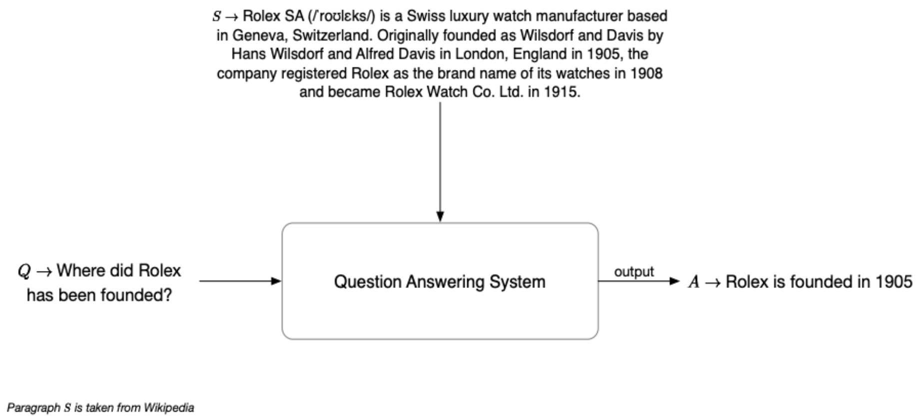



**Figure 4** – Sample Question Answering model

1. Question Pattern Mining
2. Question Pattern Prediction
3. Question Topic Selection
4. Question Ranking

### 3.1.3. BERT Language Model

We have encountered a language model called BERT, which is also known as Bidirectional Encoder Representation from Transformers developed by Google Vaswani et al. (2017). It uses a machine learning model called Transformers. We have researched other language models, and for the language translation problem, we have found two techniques which are LSTM and Transformers.

First of all, we have found out that LSTMs are slow to train, words are passed sequentially, and words are getting generated sequentially which can take significant time for the neural network to learn the language. Furthermore, LSTMs are not truly bidirectional; they learn left-to-right and right-to-left separately and concatenate afterwards. Thus, a need for Transformers arose. Transformers are faster because they can process words simultaneously and deeply bidirectional since they can learn in both directions.

On Figure 5, you can see the architecture of a transformer. It is formed by two components which are encoder and decoder.

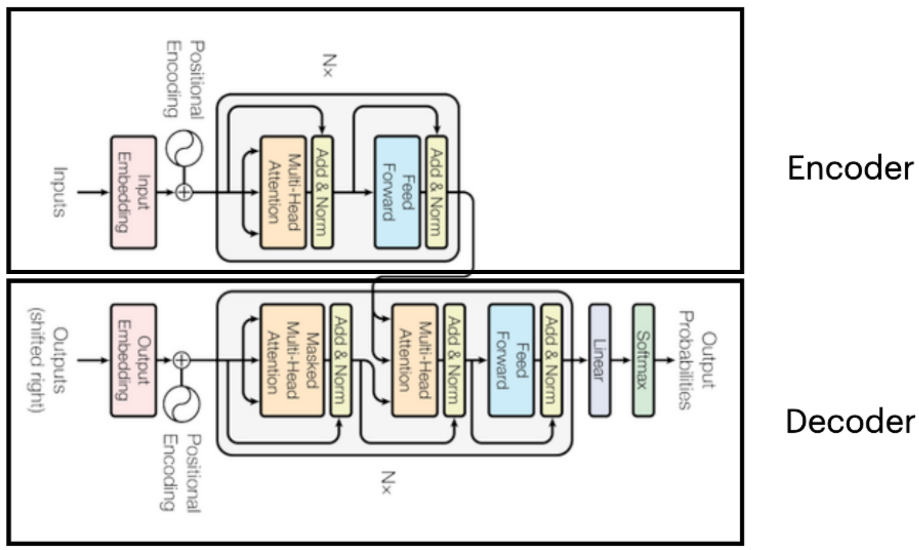



**Figure 5** – Architecture of a *Transformer*

As the BERT's name suggests, we will obtain the BERT language model if we put encoders on after another. The architecture of the BERT language model could be found in figure 6. It is the current state-of-the-art language model for NLP tasks. Chan and Fan (2019b)

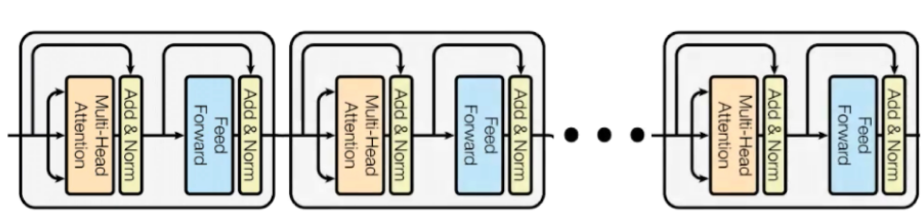



**Figure 6** – BERT language model architecture

We have found a paper that uses this language model to solve our problem, and they have made sequentially improved three models which are BERT-QG (BERT Question Generation), BERT-SQG (BERT Sequential Question Generation) and BERT-HLSQG (BERT Highlight Sequential Question Generation) models. QG is the simplest model that this paper introduces. It is the initial attempt to create a powerful Question Generation Model using BERT. As proposed by researchers, considering the previous decoded results significantly improve the quality of the model. However, in BERT-QG, token generation is performed without considering the previous states or decoded results. Due to this consideration, BERT-SQG is developed. BERT-SQG addressed the problem of ignoring the previous decodes in the BERT-QG model. However, researchers researched and concluded that BERT-SQG is not capable of producing quality questions in lengthy situations, and if an answer phase appears multiple times, it struggles to decide which one to decide. As a result, BERT-SQG is restructured, and BERT-HLSQG, a model outperforming BERG-SQG is obtained.

They have compared these models with the known to be the best question generation models which were NQG-RC and PLQG. NQG-RC is a seq2seq question model based on bidirectional LSTMs. PLQG is a seq2seq network which is capable of handling long text input. This model is known to be the state-of-the-art models for QG tasks. As shown in table 3, their models have outperformed the state-of-the-art models on every metric.

**Table 3** – Performance comparison of Question Generation models on different datasets

| | Model | BLEU 1 | BLEU 2 | BLEU 3 | BLEU 4 | METEOR | ROUGE-L |
|---|---|---|---|---|---|---|---|
| SQuAD 73K | NQG-RC | 43.09 | 25.96 | 17.50 | 12.28 | 16.62 | 39.75 |
| | PLQG | 43.47 | 28.23 | 20.40 | 15.32 | 19.29 | 43.91 |
| | BERT-QG | 34.17 | 15.52 | 8.36 | 4.47 | 14.78 | 37.60 |
| | BERT-SQG | 48.38 | **33.15** | 24.75 | 19.08 | 22.43 | 46.94 |
| | BERT-HLSQG | **48.29** | 33.12 | **24.78** | **19.14** | **22.89** | **47.07** |
| SQuAD 81K | PLQG | 44.51 | 29.07 | 21.06 | 15.82 | 19.67 | 44.24 |
| | BERT-QG | 34.18 | 15.51 | 8.57 | 4.97 | 14.57 | 37.65 |
| | BERT-SQG | 50.18 | 35.03 | 26.60 | 20.88 | 23.84 | 48.37 |
| | BERT-HLSQG | **50.71** | **35.44** | **26.95** | **21.20** | **24.02** | **48.68** |

### 3.2. Question Generation Model With BERT Language Model

As a result of this literature survey, we have decided to go with the BERT language model. BERT language model shows significant performance over various NLP tasks, such as classification, summarization, translation. It has a robust architecture behind it which is transformers. We have searched a pre-trained Turkish model of BERT, and we found a repository on GitHub for it. Starting from next week, we will start using this model as well.

As for the dataset, we have decided to go with Turkish Wikipedia dataset. Since the data of Wikipedia is close to textbook data, it should be beneficial to train our model with this dataset. Moreover, our advisor, Seniz Demir, had a Wikipedia Parser project. Thus, we will use the Turkish Wikipedia dataset.

# 4. Analysis and Modeling

Aim of this project is to develop a Turkish Question Generation Model using our custom dataset. Given a paragraph or a piece of text, our model creates the possible questions that can be answered with the information given in that paragraph.

However, such a model's development requires an extensively large and labelled dataset consisting of paragraphs, related questions, and answers to those questions. Unfortunately, such dataset is not available in the Turkish language. There are examples of it, but they neither contain large amounts of the information nor find the data reliable.

For the reasons described above, we composed our dataset in previous semester. Our dataset consists of paragraphs from Turkish Wikipedia. Using the Turkish Wikipedia's Person entities, we created the question patterns regarding to specification of the Person's in dataset. Later, these questions patterns are completed by substituting the person's name with the placeholders. As a result, we obtained formal question-answer pairs along with the paragraphs.

However, we must also confirm that the answer to the question is also found in the paragraph. For this reason, we use a fuzzy search algorithm in order to determine if the answer is in the paragraph or not. If so, we add the questions to our dataset, if not, we skip to the next question.

## 4.1. System Factors

Since our work is based on the data acquired from Wikipedia and its manipulation, source data is highly influential on the outcome. While creating the dataset, we always aim to create grammatically flawless questions while covering the variety of inputs that may come from the end-user

## 4.2. How System Works

Input data consists of  53,000 person entities from Turkish Wikipedia. Occupations of each person and their attributes are also presented. For instance, for a 'Soldier', given attributes are: 'Rank', 'Wars', 'Nationality', 'Commanders' etc. Given that most of these attributes are also mentioned in the corresponding person's description paragraph, when we create the questions that ask about these attributes, we will have the description paragraph and the questions together.

Initially, each question patterns for attributes are created manually containing a placeholder for the person's name. Later, by iterating over each person and substituting names for their attributes, we create example questions.

When we iterate over every person, we accumulate a dataset consisting of paragraphs that describe the person and questions about the person's attributes, which is also contained in the paragraph.

### 4.2.1. Modelling

The system can be inspected under two groups; Dataset Production and Development of the Question Generation Model.

Figure 6 shows the relationship between person entities, their attributes and the question patterns. Each person has a set of attributes, and each attribute has a set of question patterns.

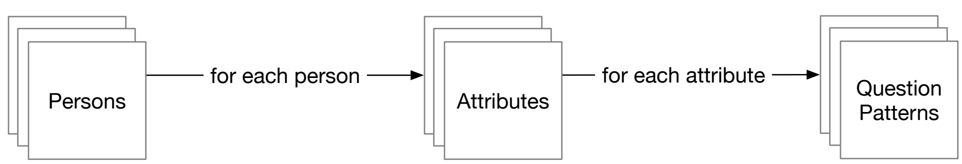



**Figure 7** – System Entities

Figure 7 shows the relationship between an attribute and the questions. For an attribute, there are multiple question patterns. When the questions are generated using the pattern, cumulatively these patterns create questions about the person's attribute.

Figure 8 shows the relationship between the person entity and how the output data for a single person is generated. Each person has a descriptive paragraph and set of attributes. These attributes yield to set of questions that the description paragraph can answer. When the paragraph and the questions are coupled together, we get a paragraph-questions pair for our

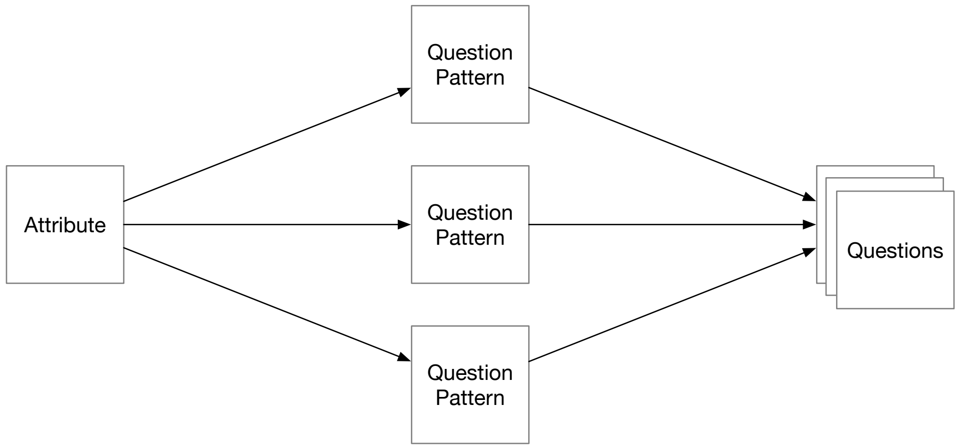



**Figure 8** – Relationship of an attribute to Questions

dataset.

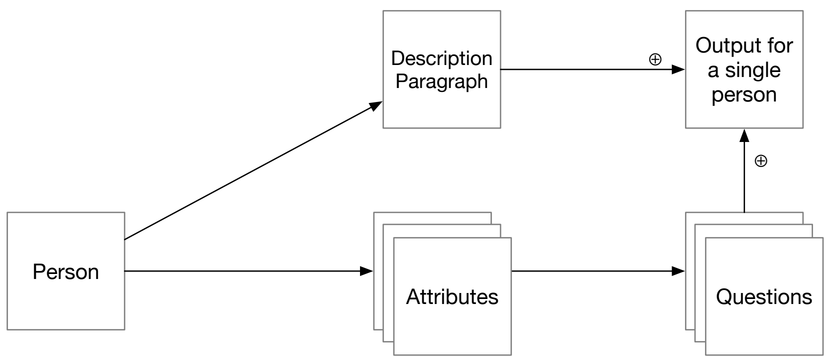



**Figure 9** – Relationship of an attribute to Questions

### 4.2.2. System Architecture

Explanation of system architecture.

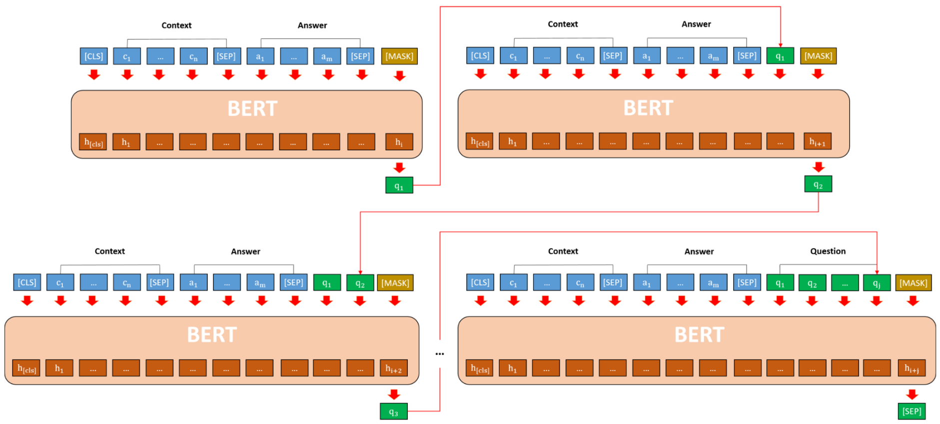



**Figure 10** – BERT-SQG Architecture Chan and Fan (2019a)

### 4.2.3. UML (Unified Modeling Language) Diagrams

UML Diagrams can be seen from Appendix C

# 5. Design, implementation and testing

During the design phase of our project, we had two different components that have to be completed. These are:

- A Turkish question answering dataset will be used to train the QG model.
- A QG model.
- User-Interface

This section will explain the design process we have followed during the design, implementation, and testing of these two components separately.

## 5.1. Design

### 5.1.1. Dataset

Before creating our dataset, we have researched if any of the datasets are suitable and sufficient. We have found one dataset that is suitable for our model. That dataset has around 2.2k questions, and the descriptive texts were about Turkish & Islamic Science History, which is not completely suitable and sufficient. Hence, we have decided to create our dataset. During the design phase of the dataset, we have used 2 Wikipedia datasets. The first data was semi-cleaned scrapped Turkish Wikipedia data. Data we had has the short descriptions and the table contents. An example of the table content could be found in Figure 11.

| 1. Moğol İmparatorluğu Hanı | |
|---|---|
| Hüküm süresi | 1206 – 18 Ağustos 1227 |
| Sonra gelen | Ögeday Han |
| Eş(leri) | Börte Ujin<br>Kulan<br>Yusui<br>Yusigen |
| Çocukları | Cuci<br>Çağatay<br>Ögeday<br>Tuluy |
| Tam ismi | |
| Temuçin | |
| Hanedan | Börçigin |
| Babası | Yesügey |
| Annesi | Höelin |
| Doğum | 1162<br>Dülün-Boldak, Moğolistan |
| Ölüm | 18 Ağustos 1227 (65 yaşında) |
| Dini | Göktengricilik |

**Figure 11** – Sample Wikipedia table content

However, this data was insufficient for our dataset since most of the table contents' data does not appear in the short description. Thus, we had to use long descriptions for each of them. After that, we also matched the whole Wikipedia Turkish dump we found from Kaggle with our first data. This data had long descriptions of every document that is currently available on Wikipedia. During the matching phase, we have decided to match each document by their id's.

After finishing the data part, we have started to analyze our data. We have grouped each person according to their occupation. After grouping, we have found out that we have twenty different occupations. Each occupation has its table attributes. Our main goal is to generate question patterns according to their occupations and the most used ten attributes for that specific occupation. Our analysis on the Wikipedia persons data could be found in Appendix A.

Finally, we have started to generate the question patterns according to the occupation and the attributes. During this process, we have decided to the question pattern styling.

### 5.1.2. Question Generation Model

As discussed in Section 4., we train various models for Question Generation task. This models are designed for other languages than Turkish. In this part of the project, our aim is train these models for Turkish while preserving their high performance scores.

Currently, we are working on two models based on BERT Language Model Vaswani et al. (2017).

- **Model 1 Rosasco (2020)** In this model, both the input embeddings and hidden size is selected as 512. Learning rate is 1e-4 and dropout 0.5.

- **Model 2 Zhang and Bansal (2019)** In this model, both the input embeddings size is 768 whereas the hidden size is selected as 600. Learning rate is 1e-5 and dropout 0.3.

### 5.2. Implementation

Similar to the design part, we have developed and implemented each component separately. We have implemented these components with the Python Programming Language.

#### 5.2.1. Dataset

While creating the dataset, we first created our classes according to the data's that we have. The classes have been created according to the data could be found in figure 19 and figure 20.

As we have mentioned in Section 5.1.1., we have already decided on the question patterns on the designing phase. In the implementation phase, we have generated around 1700 question patterns by hand. Sample question patterns that we have generated could be found in Figure 12. After generating

```
"Asker": {
   "rütbesi": [
            "{name} hangi rütbede görev yapmaktaydı?",
            "{name}'{_suffix1}n rütbesi nedir?”,
            ...
            "{name} hangi rütbeye sahiptir?"
   ],
  "bağlılığı": [
            "{name} hangi ordudaydı?”,
            "{name} hangi orduda görev aldı?”,
            ...
            "{name} hangi ülke ordusundaydı?”,
            "{name} hangi ülkenin ordusundaydı?”
   ],
  ...
  "savaşları": [
            "{name} nerede savaştı?",
            "{name} nerelerde savaştı?",
            "{name} hangi cephelerde savaştı?",
            ...
            "{name}'{_suffix1}n savaştığı cepheler hangileriydi?",
            "{name}'{_suffix1}n savaştığı cepheler hangileri?"
        ]
  ....
```

**Figure 12** – Sample Question Generation patterns

the question patterns, we needed to implement a fuzzy search mechanism while searching an attribute in the long description. We have researched and found out that in fuzzy search algorithms, the Lavensthein's distance formula. After that, we found a package that is the implementation of that formula called *fuzzywuzzy*. We pass the long description, and the answer that we want to generate a question from then the method returns us a ratio. We have decided to use 0.7 as the threshold for this ratio. If the ratio is higher than our threshold value, then we generate the questions. In Figure 13, the questions that have been generated for Ilker Basbug, who is the 26th Chief of the General Staff of Turkey. Wikipedia grouped him as a soldier, thus we have generated the possible questions using the data from the table.


```
{
  "description":"Mehmet İlker Başbuğ (29 Nisan 1943, Afyonkarahisar), Türk Silahlı Kuvvetlerinin 26. Genelkurmay Başkanı olan Türk orgeneral. ve yazar...",
  "data":[
    {
      "answer":"Orgeneral",
      "questions":[
        "İlker Başbuğ hangi rütbede görev yapmaktaydı?",
        "İlker Başbuğ'un rütbesi nedir?",
        "İlker Başbuğ'un rütbesi",
        "İlker Başbuğ'un rütbesi neydi?",
        "İlker Başbuğ rütbe",
        "İlker Başbuğ hangi rütbeye sahiptir?"
      ]
    },
    {
      "answer":"Afyonkarahisar, Türkiye",
      "questions":[
        "İlker Başbuğ nerede doğmuştur?",
        "İlker Başbuğ nerede doğdu",
        "İlker Başbuğ doğum yeri",
        "İlker Başbuğ doğum yeri neresi",
        "İlker Başbuğ doğum yeri neresidir",
        "İlker Başbuğ nereli",
        "İlker Başbuğ memleketi",
        "İlker Başbuğ memleketi nedir",
        "İlker Başbuğ doğduğu yer",
        "İlker Başbuğ doğduğu yer neresidir",
        "İlker Başbuğ doğduğu yer neresi",
        "İlker Başbuğ doğduğu il",
        "İlker Başbuğ hangi ilde doğdu",
        "İlker Başbuğ hangi ülkede doğdu",
        "İlker Başbuğ doğduğu ülke"
      ]
    },
    {
      "answer":"29 Nisan 1943",
      "questions":[
        "İlker Başbuğ doğum tarihi",
        "İlker Başbuğ doğum tarihi nedir",
        "İlker Başbuğ doğum tarihi ne zaman",
        "İlker Başbuğ doğum yılı",
        "İlker Başbuğ doğum yılı nedir",
        "İlker Başbuğ doğum yılı ne zaman",
        "İlker Başbuğ doğum günü ne zaman",
        "İlker Başbuğ doğum günü nedir",
        "İlker Başbuğ ne zaman doğmuştur?",
        "İlker Başbuğ ne zaman doğdu",
        "İlker Başbuğ doğum zamanı"
      ]
    },
    {
      "answer":"TSK Üstün Cesaret ve Feragat Madalyası, TSK Üstün Hizmet Madalyası, TSK Şeref Madalyası",
      "questions":[
        "İlker Başbuğ hangi madalyalara sahiptir?",
        "İlker Başbuğ hangi madalyaları almıştır?",
        "İlker Başbuğ aldığı madalyalar",
        "İlker Başbuğ sahip olduğu madalyalar nelerdir?",
        "İlker Başbuğ'a takdim edilen madalyalar nelerdir?",
        "İlker Başbuğ'a verilen madalyalar nelerdir?",
        "İlker Başbuğ'un kazandığı madalyalar nelerdir?",
        "İlker Başbuğ'un sahip olduğu madalyalar nelerdir?"
      ]
    }
  ]
}
```


**Figure 13** – Questions generated for Ilker Basbug (26th Chief of the General Staff of Turkey)

Another critical aspect of the implementation part was the speed. There were around fifty-three thousand people in the first data, and the whole Wikipedia data is around five hundred megabytes of data. Our computers were not sufficient for this processing so we have decided to use cloud services. We have used our previous knowledge on multiprocessing from Operating Systems and tweaked our code to work with multiple CPU's in parallel.

### 5.2.2. Question Generation Model

In the last semester, we constructed our custom dataset in a format that is similar to famous SQuAD Rajpurkar et al. (2016) dataset. Given that most of the question generation models are built on top of SQuAD dataset, we aim to incorporate our dataset as smoothly as possible.

We use the PyTorch Framework to train our QG Models based on BERT Language Model. The pretrained model uses 512 hidden dimensions, and 512 attention size. Dropout probability is set to 0.1 between layers. The Adam Optimizer is applied during the training process, with an cross entropy loss function and initial learning rate of 5e-5. The batch size for the update is set as 28. Our models used a single NVIDIA Tesla K80 GPUs on Microsoft Azure VMs[6] for 3 epochs training.

### 5.2.3. Tokenization

A tokenizer is in charge of preparing the inputs for a model. There are 3 different key features of the tokenizers. These are: Splitting strings in sub-word token strings, converting tokens strings to ids and back, and encoding/decoding Adding new tokens to the vocabulary in a way that is independent of the underlying structure which helps the model to learn unknown inputs easier

Managing special tokens (like mask, beginning-of-sentence, etc.): adding them, assigning them to attributes in the tokenizer for easy access and making sure they are not split during tokenization.

### 5.2.4. Delexicalization

Delexicalization is the process that replacing language-specific words with language-agnostic meaning. Delexicalization can be used for anonymization

[6]Standard NC6 Promo Instance

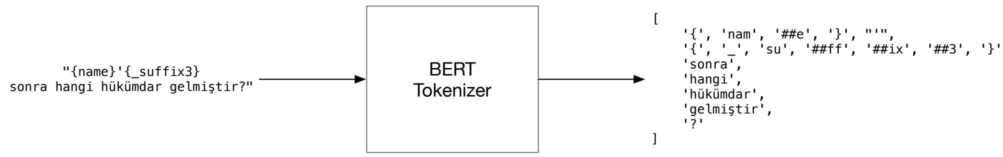



**Figure 14** – Tokenization example

(e.g., removing proper names) or to improve the performance of an NLP application by removing over-specific phrases (numericals, named entities, names, etc). After the generation of the questions, we are replacing the placeholders with the correct token. We have generated a delexicalized version of our dataset. We will mention the effect of the delexicalization on results in following sections.

| | |
|---|---|
| $x_1$ | Henry Rollins hangi dalda müzik yapmaktadır? |
| $x_2$ | {name} hangi dalda müzik yapmaktadır? |

**Figure 15** – Non-Delexicalized ($x_1$) vs Delexicalized ($x_2$) sentences)

### 5.2.5. Output Examples

Questions produced by our system, after placing the names in the placeholders as a result of delexicalization process, can be seen below against the ground truths.

**Ground Truth:** Erkin Koray aktif yılları nelerdir?
**Prediction:** Erkin Koray hangi yıllarda müzisyenlik yapmıştı?

**Ground Truth:** Alex De Souza'nın maçlardaki pozisyonu nedir?
**Prediction:** Alex De Souza hangi mevkiide oynamaktadır?

**Ground Truth:** Orhan Pamuk hangi ödülleri almıştır?
**Prediction:** Orhan Pamuk'a takdim edilen ödüller nelerdir?

## 5.3. Testing

BLEU or in longer version BiLingual Evaluation Understudy, is a metric for automatically evaluating machine-translated text. The BLEU score is a number between zero and one that measures the similarity of the machine-translated text to a set of high-quality reference translations. A value of 0 means that the machine-translated output has no overlap with the reference translation in other words it is a low-quality output while a value of 1 means there is perfect overlap with the reference translations which means a high-quality output. It has been shown that BLEU scores correlate well with human judgment of translation quality. Note that even human translators do not achieve a perfect score of 1.0. It is the N-gram precision and aims to find what percentage of machine n-grams can be found in the reference translation.

| BLEU Score | Interpretation |
|---|---|
| < 10 | Almost useless |
| 10 - 19 | Hard to get the gist |
| 20 - 29 | The gist is clear, but has significant grammatical errors |
| 30 - 40 | Understandable to good translations |
| 40 - 50 | High quality translations |
| 50 - 60 | Very high quality, adequate, and fluent translations |
| > 60 | Quality often better than human |

**Figure 16** – BLEU Score Table (scale 0-100)

**Table 4** – Evaluation results. Whole Paragraph, Delexicalization

| | | Validation | | Train | | |
|---|---|---|---|---|---|---|
| | BLEU | Sentence Loss | Word Loss | Sentence Loss | Word Loss | Train Loss |
| Epoch 1 | 0.512 | 25.223 | 1.632 | 14.323 | 0.921 | 1.156 |
| Epoch 2 | 0.523 | 24.647 | 1.553 | 8.051 | 0.527 | 0.629 |
| Epoch 3 | 0.527 | 26.744 | 1.763 | 9.135 | 0.606 | 0.548 |

Initially, we have tried to use our dataset without the delexicalization and the performance was not as expected, results are shared in Table 8. Therefore, we tried to apply delexicalization method to our dataset and the performance of our model have increased significantly, Tables 4, 5, 6, 7.

Initially our BLEU score was point three two five which was understandable to good translations. After the applying the method, we have scored

**Table 5** – Evaluation results. First Paragraph, Delexicalization

| | | Validation | | Train | | |
|---|---|---|---|---|---|---|
| | BLEU | Sentence Loss | Word Loss | Sentence Loss | Word Loss | Train Loss |
| Epoch 1 | 0.503 | 26.452 | 1.769 | 13.427 | 0.871 | 1.155 |
| Epoch 2 | 0.525 | 22.906 | 1.553 | 9.132 | 0.585 | 0.633 |
| Epoch 3 | 0.521 | 26.103 | 1.722 | 8.568 | 0.877 | 0.550 |

**Table 6** – Evaluation results. Last Paragraph, Delexicalization

| | | Validation | | Train | | |
|---|---|---|---|---|---|---|
| | BLEU | Sentence Loss | Word Loss | Sentence Loss | Word Loss | Train Loss |
| Epoch 1 | 0.502 | 31.758 | 2.046 | 13.701 | 0.882 | 1.135 |
| Epoch 2 | 0.501 | 25.602 | 1.608 | 9.698 | 0.631 | 0.634 |
| Epoch 3 | 0.524 | 22.875 | 1.506 | 8.593 | 0.568 | 0.572 |

point five two four which is very high-quality, adequate, and fluent translations. Also, our average sentence loss and word loss has been decreased significantly as well.

**Table 7** – Evaluation results. Middle Paragraph, Delexicalization

| | | Validation | | Train | | |
|---|---|---|---|---|---|---|
| | BLEU | Sentence Loss | Word Loss | Sentence Loss | Word Loss | Train Loss |
| Epoch 1 | 0.519 | 26.019 | 1.691 | 14.449 | 0.996 | 1.121 |
| Epoch 2 | 0.518 | 26.571 | 1.764 | 8.229 | 0.536 | 0.617 |
| Epoch 3 | 0.522 | 24.698 | 1.624 | 0.823 | 0.565 | 0.549 |

**Table 8** – Evaluation results. Whole Paragraph, Non-Delexicalization

| | | Validation | | Train | | |
|---|---|---|---|---|---|---|
| | BLEU | Sentence Loss | Word Loss | Sentence Loss | Word Loss | Train Loss |
| Epoch 1 | 0.327 | 67.563 | 5.753 | 52.255 | 4.521 | 4.771 |
| Epoch 2 | 0.314 | 72.211 | 6.161 | 33.405 | 2.909 | 3.001 |
| Epoch 3 | 0.325 | 77.203 | 6.570 | 23.486 | 2.016 | 2.164 |

# 6. Conclusion

In this project, we aimed to create a Turkish Question Generation Model. To develop such a model, we had to use a large dataset that would enable us to develop a capable and reliable model. Unfortunately, we realized that such a dataset does not exist in Turkish, and we had to create our own. For these purposes, we develop a system that would generate the paragraph and question pairs given Turkish Wikipedia's content.

Later, using this dataset, we ran our NN model. Initially, we encountered with the problem of person names not matching the output question of the model. To address this issue, we used a well-known method, delexicalization. Simply, we replaced the entity names with place holders before feeding them to the neural network. Consequently, outputs we obtained consisted of placeholders instead of names. Placeholders were replaced with the appropriate values in the last step. This change drastically improved the evaluation metrics.

Results were evaluation using BLEU and Cross Entropy Loss Function. Also, we attempt to train our NN with various parts of the dataset in order to find the best performing combination. Feeding the first, middle, and last paragraphs on separate occasions was one of the approaches. No significant difference in BLUE Score, Sentence, Word, and Train Loss values had been observed for different parts of the paragraph.

### 6.1. Life-Long Learning

At the beginning of the project, both of us were novels about Deep Learning and Natural Language Processing. Initially, we gathered information about the extent of these research areas from various academic survey papers, online discussion forums, and blog posts.

We also started to study the popular frameworks and libraries used for developing learning models during the initial stages. We started out with the *PyTorch* library developed for Python Language. We used online courses, textbooks and official documentation to inspect the application available on GitHub while studying the PyTorch framework.

Another topic we had to study was Language Models. Given that there are not many more reliable sources at this topic, we mostly used the official documentation of the academic papers.

Later on, we started to work on large amounts of data. At some point, our computing resources were not sufficient for such process; hence we used remote servers. We applied our knowledge from operating systems course for parallelism to utilize multi-CPU servers to their full capabilities.

Furthermore, developing a user interface which demonstrates the features of the final product enabled us to gain experience in web development.

### 6.2. Professional and Ethical Responsibilities of Engineers

We did not have a written ethical code of conduct before the beginning of this project. However, we knew each other over the years and knew each other's expectations during the project. Ethical responsibilities of engineers are not limited by the list presented below, yet, these can be examples of the ethical standards that we followed during the project.

- Honesty and authenticity
- Humility, respect and mutual understanding

- Consistency of our activity and expressions, clarity
- Cost-conscious (avoiding waste),
- Practicing in compliance with the mission and achieving the vision,

### 6.3. Contemporary Issues

Even though we may not realize it yet, we embraced Artificial Intelligence in our lives, especially over the last decade. Today, we use numerous applications from chatbots to classifiers to predictors day-to-day basis.

In this project, we developed a dataset that will enhance the development of AI applications in Turkish. We believe that the developers will appreciate the contributions of the outcomes of this project.

It is anticipated that in the upcoming years' AI will become even more ubiquitous. We strongly believe that the data will become more and more valuable with further progressions in Data Science.

Today, training of the usable DL models requires significant computing power that is not affordable by many, in the future quantum computing may enable us to have access to cheaper yet faster computing resources.

### 6.4. Team Work

Workload has been divided as equally as possible between the two members. The first task we aimed to accomplish was to create question patterns for the data. First task of the project, crafting a custom dataset has been started by Erdal, soon after Alp joined to speed up to the process.

Later, enhancements on the dataset has been carried out in collaborative fashion, just as the remaining of the tasks. There is one task, however, front-end implementation, carried out mostly by Alp.

Other tasks such as reports, meeting notes distributed evenly between Erdal & Alp.

Given that the project member-only consists of two people, we cannot talk about organizational inefficiencies or miscommunication issues.

# Appendix A: Attributes of *Person* Entities

attributes.txt

```
Total persons:  53115

Top Occupations and their attributes:
{'Asker': ['rütbesi', 1135,
           'doğumyeri', 1097,
           'doğumtarihi', 1092,
           'hizmetyılları', 1038,
           'bağlılığı', 940,
           'ölümyeri', 919,
           'savaşları', 915,
           'ölümtarihi', 887,
           'komutaettikleri', 689,
           'madalya', 578],
 'Basketbolcu': ['pozisyon', 1375,
                 'doğumtarihi', 1360,
                 'doğumyeri', 1356,
                 'takım1', 1303,
                 'takımyıl1', 1213,
                 'takım2', 1154,
                 'takımyıl2', 1103,
                 'takım', 1098,
                 'lig', 1031,
                 'takım3', 1013],
 'Bilim adamı': ['doğumyeri', 1411,
                 'dalı', 1388,
                 'doğumtarihi', 1258,
                 'milliyeti', 1011,
                 'çalıştığıyerler', 926,
                 'ölümyeri', 849,
                 'ölümtarihi', 838,
                 'ödüller', 775,
                 'önemlibaşarıları', 665,
                 'vatandaşlığı', 577],
 'Buz patencisi': ['ülke', 217,
                   'doğumtarihi', 214,
                   'koç', 207,
                   'combinedtotal', 192,
                   'combineddate', 191,
                   'fsscore', 185,
                   'fsdate', 185,
```

```
                  'koreograf', 145,
                  'spscore', 131,
                  'spdate', 130],
'Filozof': ['doğumtarihi', 281,
            'ölümtarihi', 241,
            'doğumyeri', 225,
            'çağ', 221,
            'bölge', 212,
            'ölümyeri', 192,
            'etkilendikleri', 192,
            'etkiledikleri', 178,
            'ilgialanları', 167,
            'okulgelenek', 150],
'Futbolcu': ['doğumyeri', 17819,
             'pozisyon', 17649,
             'doğumtarihi', 17559,
             'tamadı', 15754,
             'kulüp1', 14197,
             'kulüpyıl1', 14039,
             'kulüp2', 12783,
             'kulüpyıl2', 12725,
             'kulüp3', 11288,
             'kulüpyıl3', 11250],
'Güreşçi': ['doğumyeri', 152,
            'doğumtarihi', 150,
            'debut', 139,
            'başlık', 135,
            'ringadları', 131,
            'eğiten', 114,
            'doğumadı', 103,
            'eğitildiğiyer', 86,
            'yaşadığıyer', 44,
            'emekliliği', 34],
'Hakem': ['turnuva', 127,
          'görevi', 126,
          'doğumtarihi', 124,
          'yıl', 123,
          'doğumyeri', 114,
          'meslek', 73,
          'tamismi', 43,
          'ölümtarihi', 17,
          'ölümyeri', 7,
          'görevi2', 2],
'Kişi': ['doğumtarihi', 8861,
         'meslek', 8267,
```

```
         'doğumyeri', 7433,
         'ölümtarihi', 4109,
         'ölümyeri', 3675,
         'doğumadı', 2953,
         'aktifyılları', 2139,
         'başlık', 2051,
         'yer', 1775,
         'etkinyılları', 1711],
'Kraliyet': ['hükümsüresi', 1545,
             'sonragelen', 1425,
             'veraset', 1417,
             'ölümtarihi', 1412,
             'öncegelen', 1404,
             'babası', 1379,
             'hanedan', 1310,
             'doğumtarihi', 1285,
             'ölümyeri', 1075,
             'annesi', 1005],
'Makam sahibi': ['makam', 3694,
                 'doğumyeri', 3685,
                 'doğumtarihi', 3508,
                 'dönembaşı', 3486,
                 'öncegelen', 3160,
                 'dönemsonu', 2974,
                 'sonragelen', 2817,
                 'partisi', 2451,
                 'ölümtarihi', 1824,
                 'ölümyeri', 1802],
'Manken': ['yer', 734,
           'doğumtarihi', 733,
           'saçrengi', 715,
           'gözrengi', 713,
           'ulus', 645,
           'doğumadı', 234,
           'bgcolour', 209,
           'imagesize', 186,
           'birthname', 89,
           'boy', 69],
'Müzik sanatçısı': ['artalan', 5441,
                    'tarz', 5402,
                    'etkinyıllar', 4894,
                    'meslek', 3814,
                    'plakşirketi', 3442,
                    'doğumadı', 2874,
                    'köken', 2777,
```

```
                    'yer', 2134,
                    'doğum', 2069,
                    'çalgı', 1983],
'Oyuncu': ['yer', 2785,
           'doğumtarihi', 2584,
           'meslek', 2164,
           'etkinyılları', 2052,
           'doğumadı', 1389,
           'altyazı', 1133,
           'evlilik', 960,
           'ulus', 644,
           'ölümtarihi', 511,
           'ölümyeri', 494],
'Profesyonel güreşçi': ['doğumyeri', 96,
                        'başlık', 93,
                        'debut', 90,
                        'doğumadı', 90,
                        'doğumtarihi', 89,
                        'ringadları', 80,
                        'eğiten', 73,
                        'eş', 55,
                        'eğitildiğiyer', 53,
                        'çocuklar', 35],
'Sanatçı': ['doğumtarihi', 702,
            'alanı', 695,
            'ölümtarihi', 508,
            'ölümyeri', 475,
            'doğumyeri', 424,
            'milliyeti', 330,
            'doğumadı', 314,
            'yer', 294,
            'resimaltı', 278,
            'ünlüyapıtları', 212],
'Sporcu': ['doğumtarihi', 936,
           'doğumyeri', 814,
           'ülke', 800,
           'spor', 795,
           'yarışma', 513,
           'uyruk', 196,
           'kei', 179,
           'resimaltı', 178,
           'ağırlık', 176,
           'ölümtarihi', 170],
'Tenis sporcu': ['doğumyeri', 239,
                 'vatandaşlık', 235,
```

```
                 'enyükseksıralama', 229,
                 'oyunstili', 228,
                 'wimbledonsonuçları', 228,
                 'amerikaaçıksonuçları', 225,
                 'fransaaçıksonuçları', 224,
                 'avustralyaaçıksonuçları', 223,
                 'toplamkupa', 222,
                 'yaşadığıyer', 216],
'Voleybolcu': ['doğumtarihi', 112,
               'doğumyeri', 110,
               'pozisyon', 107,
               'milliyeti', 104,
               'kulüpyıl', 104,
               'kulüptakım', 103,
               'bulunduğukulüp', 101,
               'numarası', 99,
               'millitakım', 98,
               'milliyıl', 75],
'Yazar': ['doğumyeri', 1911,
          'meslek', 1897,
          'doğumtarihi', 1776,
          'milliyet', 1311,
          'ölümtarihi', 1079,
          'ölümyeri', 1043,
          'başlık', 751,
          'tür', 620,
          'dönem', 578,
          'ilkeser', 513]}
```

---

# Appendix B: UML Diagrams

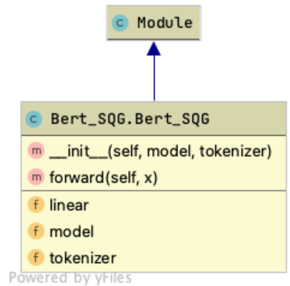



**Figure 17** – `BERT-SQG` Class

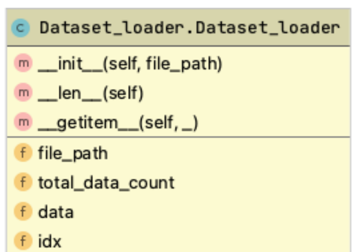



**Figure 18** – `Dataset_loader` Class

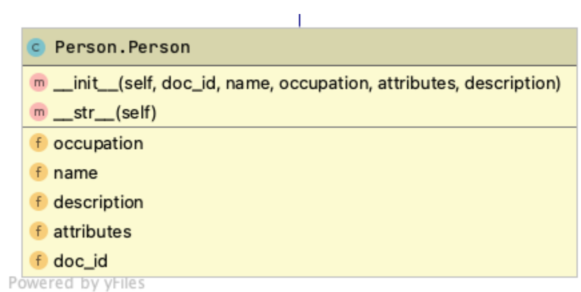



**Figure 19** – `Person` Class

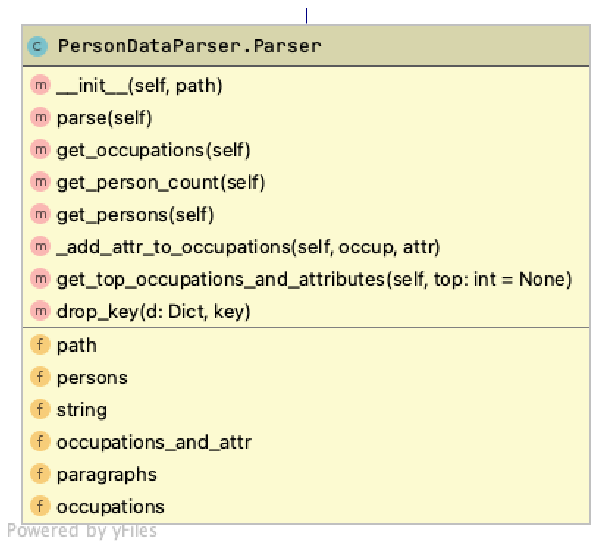



**Figure 20** – `PersonDataParser` Class

# References

Y.-H. Chan and Y.-C. Fan, "BERT for question generation," in *Proceedings of the 12th International Conference on Natural Language Generation.* Tokyo, Japan: Association for Computational Linguistics, Nov. 2019, pp. 173–177. [Online]. Available: https://www.aclweb.org/anthology/W19-8624

N. Duan, D. Tang, P. Chen, and M. Zhou, "Question generation for question answering," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.* Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 866–874. [Online]. Available: https://www.aclweb.org/anthology/D17-1090

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

Y.-H. Chan and Y.-C. Fan, "A recurrent BERT-based model for question generation," in *Proceedings of the 2nd Workshop on Machine Reading for Question Answering.* Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 154–162. [Online]. Available: https://www.aclweb.org/anthology/D19-5821

A. Rosasco, "Question Generation con BERT," Ph.D. dissertation, Università di Pisa, Human Language Technologies, May 2020. [Online]. Available: https://github.com/andrew-r96/BertQuestionGeneration/blob/master/HLT_Report.pdf

S. Zhang and M. Bansal, "Addressing semantic drift in question generation for semi-supervised question answering," 2019.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," 2016.