MARMARA UNIVERSITY

FACULTY of ENGINEERING

DEPARTMENT of COMPUTER ENGINEERING

CSE4197 – Analysis and Design Document

"Concept and Implementation of a Turkish Chatbot Service"

Group Members

150116826 – Erencan Erkaya

170214013 – Mehmet Can Yüney

Supervised by

Assoc. Prof. Dr. Murat Can Ganiz

1. Introduction

Today in Turkey, university student candidates are forced to resort to various ways in order to obtain information about the department and the university of their choice. They can either collect information about the universities from the universities' website or try to collect information from forum sites. Although forums are easy to access and use, unfortunately, the speed in which the candidates' questions are answered is proportional to their ability at finding a relevant title for their question since responders see the title first. On the other hand, information sites like Wikipedia or universities' websites require no waiting time for obtaining data in comparison to the forums and are easy to access. But in these sites, information is usually found in the form of raw data, while individuals need only specific data such as the date of an event. The effort required to extract the information makes the candidates struggle and leave many of their questions unanswered. In order to fulfill the lack of satisfying information, a system is required for providing desired information and also retrieving the information as fast as possible.

A chatbot is artificial intelligent base software that can chat with a user in natural language. Even though artificial intelligent term became popular in 20th centuries, chatbots' history begins with the Turing Test in 1950 [18]. The growing demand on faster systems increased the requirement of chatbots in every field. Considering the technological point of view, chatbots have had resemblance to question answering systems that offer fast and accurate information platforms for the users. Thus, they provide a relatively reliable and casually rapid conversation as they extract the desired information.

The present project concerns a Turkish chatbot/question answering system that will help answer questions of university student candidates rapidly and precisely during the university selection period in Turkey. Hence this project is expected to manage the issues resulted from delay while obtaining an answer for a question and lack of satisfying information. Also, it is expected to offer a contribution to Turkish NLP systems.

1.1. Problem Description and Motivation

With the aid of technology, we live our lives at a higher pace than ever before. In every field, people have a preference for faster services. With the emergence of computers and the internet, people started to access the knowledge they're looking for as fast as possible through the World Wide Web. They sometimes search for this information from general information sites and from forum sites that they can use to obtain answers from other people.

Although forum sites are very useful places for providing the required information, getting a quick answer or even a concise answer heavily depends on the noticeability of the subject title that the user chooses. That reason sometimes causes some people to be unable to find the things that they were looking for and therefore employ different methods, as explained below.

A second method is visiting general information sites, which provide people with information but also bring up the possibility of acquiring false information because not all of the articles are written by authorities. The job of verifying the correctness of the information within falls to the users. Another issue of these sites is that not all of the information presented on them is simple and easily understood. Although the desired data is only a place or date of an event, the response usually comes in the form of a paragraph in these kinds of sites. Due to this, the speed of information flow is slowed, which is in contrast to the desires of the average user in regards to speed.

In order to resolve the issues between these two approaches, we suggest the chatbot and question answering systems. A chatbot is a service that simulates how a human would communicate as a conversational partner. With machine learning algorithms, chatbots can provide a continuous conversation with the user and guide the user for the context of the conversation. These systems are designed for obtaining the fastest and the most satisfying answers in response to users' questions. Owing to chatbot and question answering systems that are based on machine learning (ML), natural language processing (NLP) and information retrieval (IR), three emergent research fields of 20th and 21st centuries, users are able to reach all kinds of data at high speed. These systems are designed for providing the fastest and the most accurate answer to people's questions. The aforementioned systems have several advantages as mentioned below:

- High-speed access to information
- Concise and satisfying answers
- Minimum effort on the part of the user

These systems are generally implemented in morphologically poor languages (e.g. languages that do not depend on affixes much, such as English). However, Turkish is a highly difficult language with regard to NLP due to its affixes and word sense ambiguity (a single word have multiple meanings) born from a rich morphological system. Therefore, Turkish is a remarkably difficult language for implementing chatbot and question answering systems.

In this project, we are planning to implement a chatbot and question answering system in order to

fulfill the need for rapid and precise information in university selection periods. We mainly aim to help university candidates and hope to contribute to the advancement of Turkish NLP systems as well.
The reason we think our project is worthwhile, as two current university students who have had their whole education in Turkey, we believe that this proposed system will help the candidates during university selection periods. Moreover, because of being aware of the significance of accessing accurate data and the drawbacks of lacking information, we also believe that such a system provides a platform for the candidates to obtain desired information without any delay. Furthermore, we want the candidates not to have any worries related to inaccurate data.

1.2. Scope of the Project

1.2.1. Dataset

Since there is no existing state-of-the-art dialog dataset in Turkish, obtaining the data for our domain (university) is one of the crucial part for our project. We tried to obtain data out of famous forum sites however these sites contain a lot of offensive, immature and subjective information such as "Department X is better than department Y". These kinds of statements are not correct for everyone and confuse the candidates. So we will collect data out of counseling twitter pages and counseling forum sites because those sites contain most accurate answers even though the question is specific for each person. Our chatbot will be domain specific so out of domain questions are not in our scope because of that, we did not collect the questions/answers about non-related topics.

1.2.2. The answering mechanism

In our system, there can only be two intent, query or information.
After extracting the intent and excising irrelevant sentences out of the input, the system will return an answer to the given query. Answers will be created in two ways.
The first way involves a rule-based algorithm. Frequently asked questions or inflexible questions will be coded as rule-based. In other words, if such a question is detected, our system will return prepared answers.
The second way involves using a machine learning algorithm. For complex questions, in other words, questions that do not make sense via a rule-based algorithm answers will be generated out of the collected dataset. Our system will judge the similarity between dataset and the given question

and it will return the best-matched answer for on the given question.
Unlike conversational chatbots, passing the Turing test is not in the scope of the project even though our system will chat with the user. We are focusing on the fulfillment of the users' desire for information as fast as possible, so our system does not need to behave like a human or pretend to be one.

1.2.3. User interface

In order to test our system, we will create a web-application so that users can use the system with ease. With this user interface, we will collect the conversation between users and the system in order to improve the system with the collected data. We will also collect the usage of the links in order to calculate our success given below. We will ask the users to grade the system from 1 to 5 and for a brief comment, right after the conversation. Hence, we will find out the weaknesses of the system, so that we can improve these parts based on user feedback. We will also use this feedback for sentiment analysis and eventually evaluate user satisfaction through conversation with the dialog system itself alone.
For the whole system, our success objectives are listed below:

- 7 of the 10 responses returned during the conversation should be accurate
- The percentage of users who clicks on links should be above 60 percent
- The average star that is given by the users, should be 3.5 out of 5

1.2.4. Constraints

- ❖ The collected dataset can contain unrelated and uncalled-for information such as political or racist discourses. Therefore collected dataset must be filtered.
- ❖ Universities can have more than one name, such as short names. For example, Middle-East Technical University is called as ODTU, METU or even "orta-doğu". Therefore named entity recognition (NER) is a challenging topic for our project.
- ❖ Subjective questions such as "Is X department in Y university better than Z university", is hard to return a correct answer because there can be various answers for each person. Therefore the dataset must be collected carefully in order to not miss guide the candidate.
- ❖ The system requires the user's permission in order to obtain the conversation between the user and the system.

1.2.5. Assumptions

- Users will pay attention to punctuations due to sentence parsing
- Users will have a computer that can connect to the internet
- Users will provide a question which is related to the domain (universities)
- Each question must include the university or department name in order to understand the content of each sentence. In other words, there will be no usage of “it” for referring to a university or a department

1.3. Definitions, acronyms, abbreviations

All abbreviations are given in Table 1.

Table 1 – Abbreviations

Abbreviation	***Word in meaning***	***Abbreviation***	***Word in meaning***
ML	Machine Learning	MLM	Masked Language Model
NLP	Natural Language Processing	SQuAD	Stanford Question Answering Dataset
IR	Information Retrieval	SSL	Secure Sockets Layer
FAQs	Frequently Asked Questions	SQL	Structured Query Language
NER	Named Entity Recognition	ORM	Object-Relational Mapping
AIML	Artificial Intelligence Markup Language	UML	Unified Modeling Language
XML	Extensible Markup Language	ER	Entity-Relationship
CRF	Conditional Random Field	ÖSYM	Ölçme, Seçme ve Yerleştirme Merkezi
TL	Transfer Learning	UI	User Interface
LM	Language Model	MVC	Model - View - Controller
GloVe	Global Vectors for Word Representation	API	Application Programming Interface
LSTM	Long Short-Term Memory	FQAs	Frequently Asked Questions
biLM	Deep Bidirectional Language Model	e.g.	for example

2. Related Works

2.1. History of the chatbots

The history of chatbots dates back to the mid-20th century. The first known chatbot ever by the name of ELIZA was developed in 1966. ELIZA was designed to emulate a Rogerian psychologist [1]. The answering system of ELIZA mostly depends on basic pattern matching. With this method, ELIZA mostly returns to users their own sentences in the form of questions. Its conversational skills are designed to make the user feel more comfortable by asking a question about the user him/herself [2]. ELIZA needs structured sentences in order to return something meaningful, however even with the structured sentence sometimes it cannot return an answer related to the question or sentence. When this happens, ELIZA returns an unrelated question to continue the conversation. Even though this method might work on chatbots that try to communicate with the users on various topics, in our system, this method would not satisfy the users' desire. Because changing the subject or in other words leaving user's question unanswered, will leave the user uninformed, which is strictly against our aim.

After ELIZA, the next big name for chatbot history is Artificial Linguistic Internet Computer Entity (A.L.I.C.E). ALICE was developed in 1995 by Richard Wallace [3]. Unlike ELIZA, ALICE was able to use natural language processing which increased the meaningfulness of the conversation. ALICE was open source and on the backend, it was using AIML (artificial intelligence markup language). Also, AIML is open source to ease development of chatbots. AIML is very similar to XML [4]. It allows extracting the desired information from a sentence using regular expressions.

Due to its morphology, extracting the information out of a sentence in Turkish is not as easy as it is in English. For example, we can easily extract the emotion out of “I am bored.” in English, but the sentence like “Sıkıldım.” is not easy to process. However, with the aid of AIML, Topçu, Şen and Amasyalı [5], developed pattern matching based chatbot for Turkish.

With the help of ML and NLP, especially deep learning, chatbots become more flexible when selecting an answer for given questions. In the work of Mujeeb, Javed, Arshad [6], authors proposed a method which involves decision-trees and rule-based algorithms for a diagnostic chatbot for achluophobia and autism. The system that they proposed, achieved over 80% diagnostic accuracy. In the work Mrini, Laperrouza and Dillenbourg [7], authors introduced a chatbot that uses forum sites data. They obtained the data out of Wrong Planet forum and labeled the %1 of this data with the help of Amazon Mechanical Turk. They trained a model with the labeled dataset to predict the usefulness of replies in the remaining unlabeled threads by using sent2vec as a representation model [10]. The cited project's chatbot using word2vec [8,9] cosine similarity for selecting the most similar question. After selecting the most similar question, chatbot selects the most useful answer with word2vec matching.

Both works show that ML and NLP algorithms have great aid on chatbots, however making these chatbots requires a lot of data but in our project, data is limited and also hard to obtain.

2.2. Chatbot services

2.2.1. Rasa

Rasa introduced a pair of tools, Rasa NLU, Rasa Core. The main purpose of this open source project is to make machine learning based dialog algorithms and natural language understanding tools both accessible and easy-to-use for non-specialist software developers. Rasa only requires data to create a chatbot. However, there is no support for Turkish.

In Bocklish, Faulkner, Nick and Alan [19], authors test the Rasa Core with bAbI dialogue dataset [20]. They propose that this is a basic slot-filling task where the system has to fill several slots before being able to answer the given question about a restaurant. For this task, the available slots are location, the number of people, cuisine and price range. "bAbI is an interesting dataset due to the inherent non-linearity of the problem - there are multiple ways to get the same information so there is not a single 'correct' action in every case. [19]" so due to that reason; precision, recall or accuracy are not the most appropriate metrics for evaluating the success of this project. So as in Figure 1, the authors provided a probability table in order to see the success of the system while asking the questions about slots which system does not know about.

2.3. Turkish chatbot services

2.3.1. Dahi.ai chatbot

Dahi.ai is a chatbot creation platform that has developed by YapayTech. Compared to the known chatbot service, Wit.ai which is a platform based solely on meaningfulness, whereas the Dahi.ai platform is separated from its competitors by integration options and easy-to-use means of interpretation, dialogue system, third-party service, and messaging applications, and the ability to create bots from start to finish. Dahi.ai can be integrated with Facebook Messenger.

2.3.1. Sestek chatbot

Unlike Dahi-ai Sestek provides ready to use chatbot for the customer’s specific field. Sestek Chatbot is advertised as an effective, easy-to-use live chat software solution for providing high quality and intelligent service to customers.

2.3.1. hızlıYol

hızlıYol Technology creates workflows with company-specific scenarios and provides analysis and reporting of data collected with chatbots. hızlıYol Technology continues to develop solutions for HR, Sales, Construction, Logistics, Plant and Facility Management with ITU (Istanbul Teknik Üniversitesi) Core support.

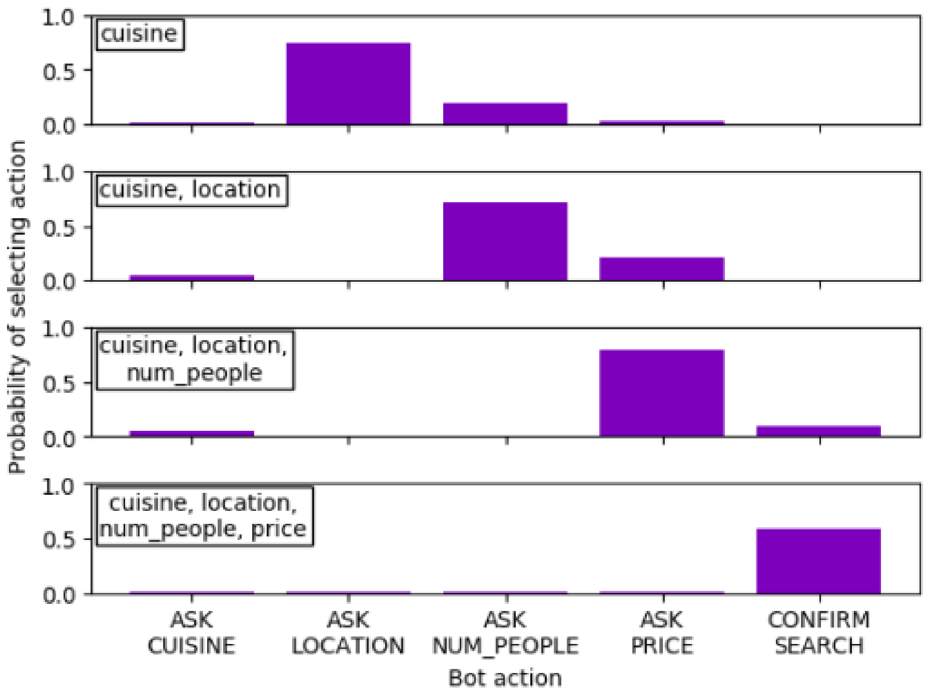


Figure 1 – Probabilities of choosing actions for bAbI dataset (Section 2.2.1) [19]

2.4. Related works for sub-components of chatbots

Parsing a sentence is a challenging task in Turkish due to the language’s complex morphology. In the work of Güngördü and Oflazer [11], authors suggested lexical-functional grammar formalism for parsing Turkish. This method inspired Tomita’s parser that was developed in Carnegie Mellon University Center. This method can parse 82% of the sentences directly. In the work of Eryiğit, Nivre and Oflazer [12], authors present data-driven dependency parsing of Turkish. They achieved the highest accuracy with the classifier-based approach with the use of inflectional groups, lexicalization and morphological information.

Named Entity Recognition (NER) is one of the most important steps for chatbots and question answering systems. In the work of Şeker and Eryiğit [13], authors introduced a Turkish NER model using a conditional random field (CRF). Proposed CRF was trained with morphological and lexical features. They achieved the highest results (95% in MUC and 92% in CoNLL metric) in the literature for Turkish NER.

2.4.1. Transfer learning in NLP

In computer vision, transfer learning (TL) is a highly used method thanks to the ImageNet [21] dataset. However in 2018, transfer learning started to be used in NLP and achieved some state of the art results on various NLP tasks. In basic terms, transfer learning based on three steps:

- Train a model with a large-scale dataset (language model)
- Use that pre-trained model to conduct learning with the domain-specific dataset (fine tuning)
- Use the model on required NLP task

There are three main TL methods on NLP which are ULMFiT, ELMo, BERT. These methods require a lot of computational power while training a model with a large-scale dataset.

2.4.1.1. ULMFiT

In Howard and Ruder [22], authors suggested an efficient TL method that can be applied in any NLP tasks and they named it as ULMFiT which is short for Universal Language Model Fine-Tuning. Existing NLP methods requires task-specific modifications and training from scratch. However, ULMFiT uses a pre-trained model which is trained with large-scale dataset so it does not require large corpus for domain specific training and authors also release their pre-trained model which is trained on Wiki-103 [23].

ULMFiT achieved state of the art results in following NLP tasks:

- Sentiment Analysis
- Question Classification
- Topic Classification

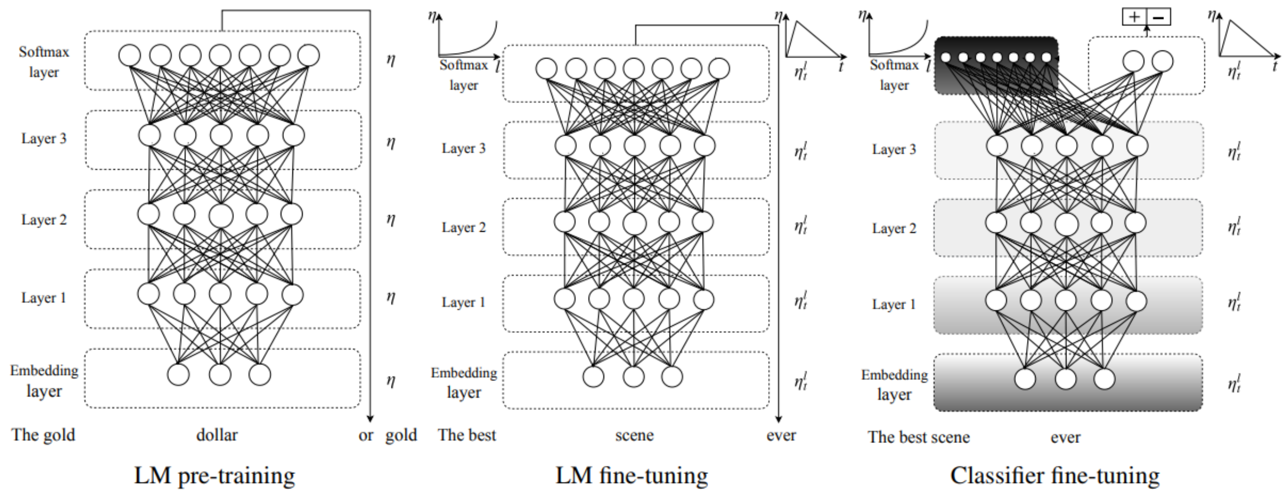


Figure 2 – Three stages of ULMFiT [22]

2.4.1.2. ELMo

In Peters, Neumann, Iyyer, Gardner, Clark, Lee and Zettlemoyer [24], authors introduced ELMo which is a method that uses deep bidirectional language model (biLM) which is pre-trained with a large corpus. Classic word embedding methods such as GloVe and word2vec, produce a vector for each word without looking the meaning of the word in a specific sentence. Instead of using fix embedding for each word, ELMo looks at the entire sentence and produces a specific vector for that word in that sentence so in other words, one word can have multiple embedding depending on the sentence. Authors also suggest that stacked LSTM layers improve the performance over using the top LSTM layer. They also show that ELMo significantly improves the state of the art in many NLP tasks like question answering, textual entailment, and sentiment analysis.

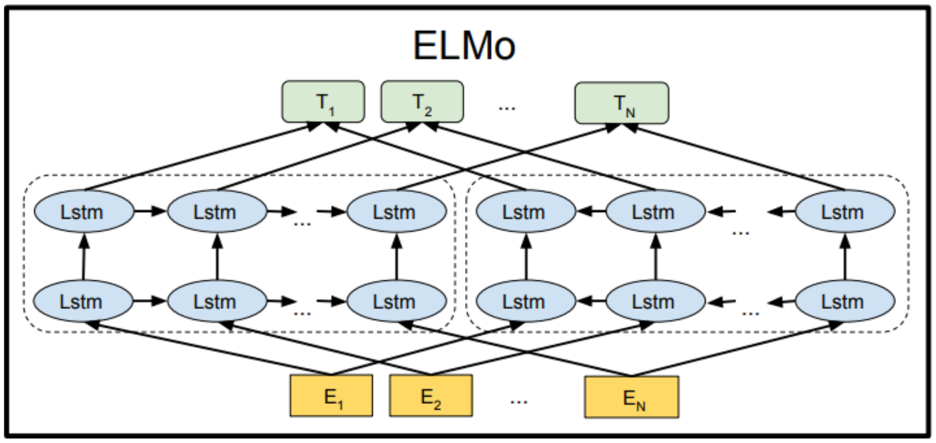


Figure 3 – ELMo pre-training model architecture by independently left-to-right and right-to-left LSTM [25]

2.4.1.3. BERT

BERT (Bidirectional Encoder Representations from Transformers) was introduced in the work of Devlin, Chang, Lee, Toutanova [25]. Unlike ELMo, BERT is based on deep bidirectional representations by jointly conditioning on both left and right context in all layers. So basically, the pre-trained BERT models can be fine-tuned with just one additional output layer. BERT uses new pre-training objective: the masked language model (MLM). MLM randomly masks some of the tokens from the given input sentence and tries to predict the original vocabulary id of the masked word. MLM allows fusing the left and the right context which allows to pre-train deep bidirectional Transformer unlike concatenation of independently trained left-to-right and right-to-left language models. In [25], the results show that BERT achieved state of the art results in eleven NLP tasks including SQuAD v1.1 [26].

SQuAD (The Stanford Question Answering Dataset) contains 100k question/answer pairs and questions posed by crowd workers on a set of Wikipedia articles. In the work of Rajpurkar, Zhang, Lopyrev and Liang, logistic regression achieved 51.0 F1 score. However, human performance is 91.2(F1 score) so authors claim that this dataset presents a good challenge for future research. For understating the success of BERT, this dataset represents a good example because BERT achieved 93.2 F1 score which outperforms the human by 2.0.

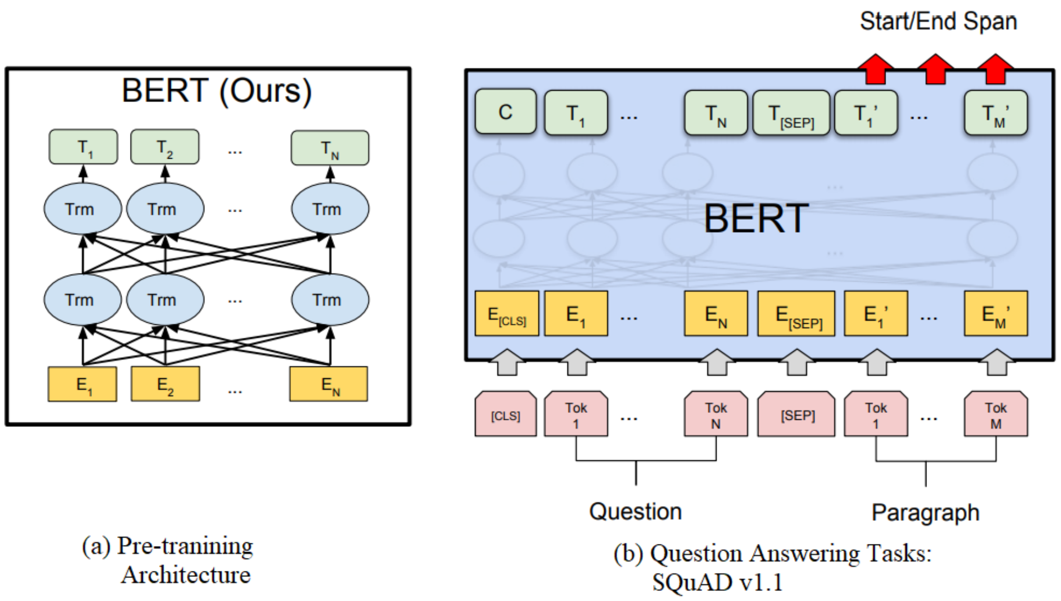

BERT (Ours)
T_1 T_2 ... T_N
Trm
E_1 E_2 ... E_N
Start/End Span
C T_1 ... T_N $T_{[SEP]}$ T_1' ... T_M'
BERT
$E_{[CLS]}$ E_1 ... E_N $E_{[SEP]}$ E_1' ... E_M'
[CLS] Tok 1 ... Tok N [SEP] Tok 1 ... Tok M
Question
Paragraph
(a) Pre-traning Architecture
(b) Question Answering Tasks: SQuAD v1.1

Figure 4 – a) BERT pre-training model architecture by a bidirectional Transformer

b) BERT for providing result with only one additional output

layer for sequence-level task [25]

3. Project Requirements

3.1. Functional Requirements

i. There will not be a login system for users.

ii. Users shall be able to:
   - Chat with the system
   - List the frequently asked questions
   - End the conversation
   - Rate and the system at the end of the conversation

3.2. Nonfunctional Requirements

3.2.1. Security Requirements

- ✓ The connection must be encrypted with SSL.
- ✓ In order to eliminate the SQL injection risks, we will use object-relational mapping (ORM) tool.

3.2.2. Performance Requirements

- ✓ There must be no extra delay while answering a question because our system is not a conversational chatbot (Section 1.2.2)
- ✓ If response time exceeds one minute, the process must be terminated and the user must be warned.

3.2.3. Usability Requirements

- ✓ The interface must be responsive.
- ✓ There must be a browser cookie in the web service in order to receive permission to obtain the user's data and whole conversation between the system and the user.

3.2.4. Maintainability Requirements

- ✓ If there is a problem while returning an answer to the given question, the error must be printed to the user interface with a proper natural language and also to the log of the conversation so the developer can see and evaluate the error. Consequently, this brings a rapid solution to the issue.

4. System Design

4.1. System Design

4.1.1. UML Case Diagrams

Following diagram shows the main case diagram of our system. There will be no login service to our system however if the user wants to share his/her information with the system via chatting, the system will keep track of these information and store them however only the developer will be able to see these information.

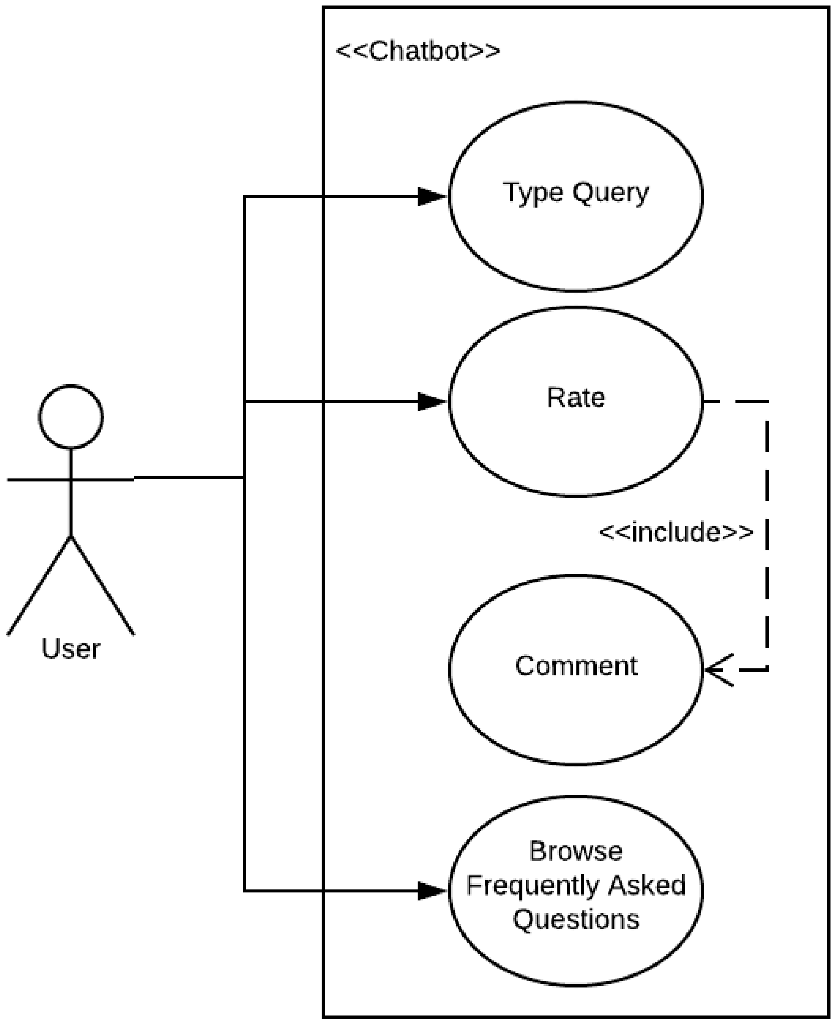
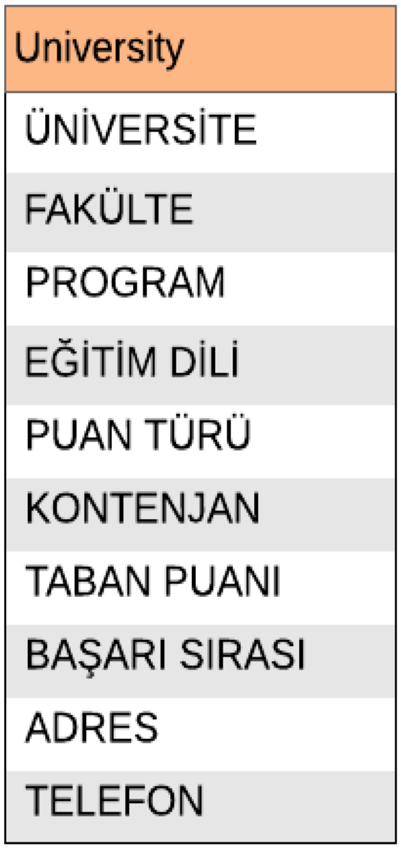


Figure 5 – UML Case Diagram – Chatbot on Website

4.1.2. Database ER Diagrams

In our system, there will be three databases.

First one is responsible for universities. It includes information such as name, location, departments. We collected this data out of universities website and also from ÖSYM's website. After collection, we used Elasticsearch for speed up the search, because this dataset will be used for each simple question that is defined in Section 5.2.



University
ÜNİVERSİTE
FAKÜLTE
PROGRAM
EĞİTİM DİLİ
PUAN TÜRÜ
KONTENJAN
TABAN PUANI
BAŞARI SIRASI
ADRES
TELEFON

Figure 6 – Entities for University Database

Second database is responsible for question-answer pairs. As mentioned in Section 1.2.1, we will collect our data out of famous counseling twitter pages and forum sites in order to increase the information inconsistency. There will be many-to-many relationships between question-answer pairs. The system will also track the number of asked per each question.

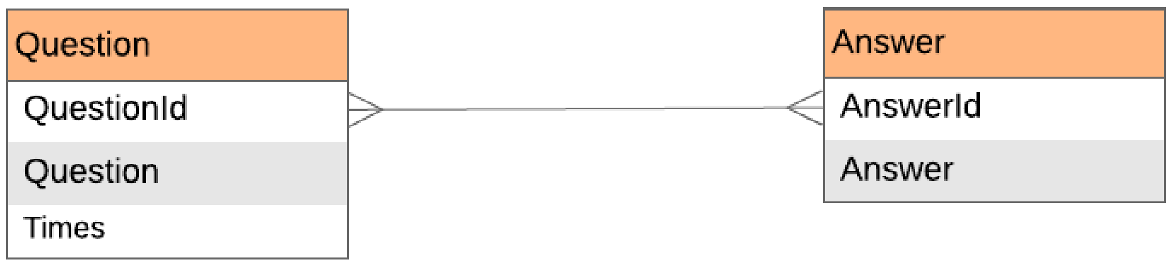


Figure 7 – ER Diagram for Question-Answer Database

The last database will be responsible for logging. At the end of each conversation, the system will store the whole conversation script and also rate and optionally comment which are taken from the user. For this purpose, we will use MongoDB which is a no-SQL database.

4.1.3. User Interface

Our website will be built on MVC (Model – View – Controller). Users' interactions will be controlled by MVC. MVC architecture of our system is given in Figure 8.

- View: Handles the display operations
- Model: Handles the data manipulations.
- Controller: Handles the user's actions.

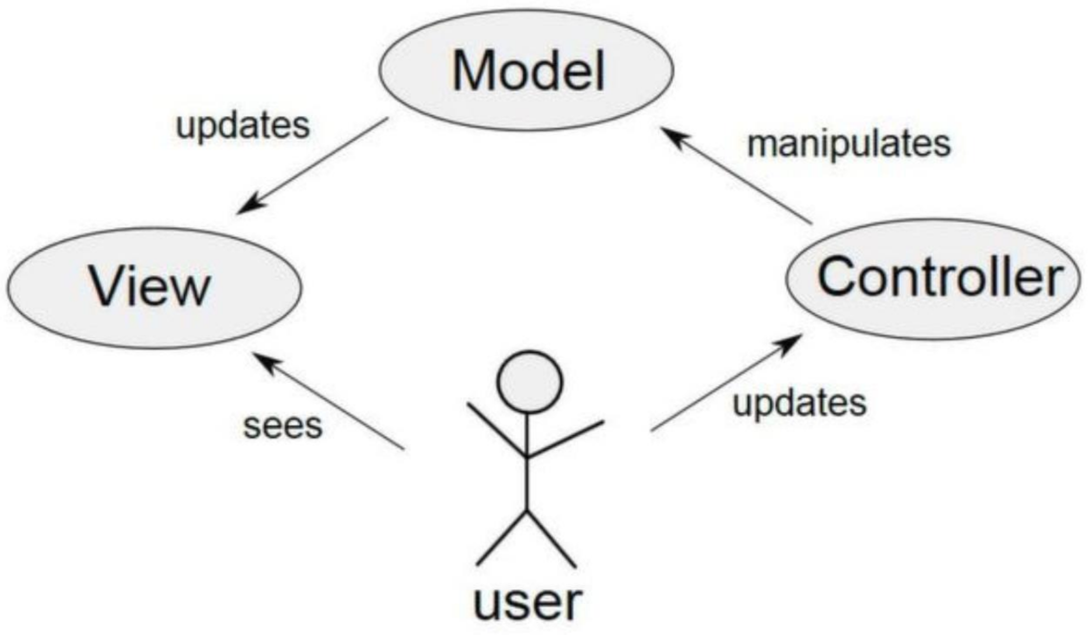
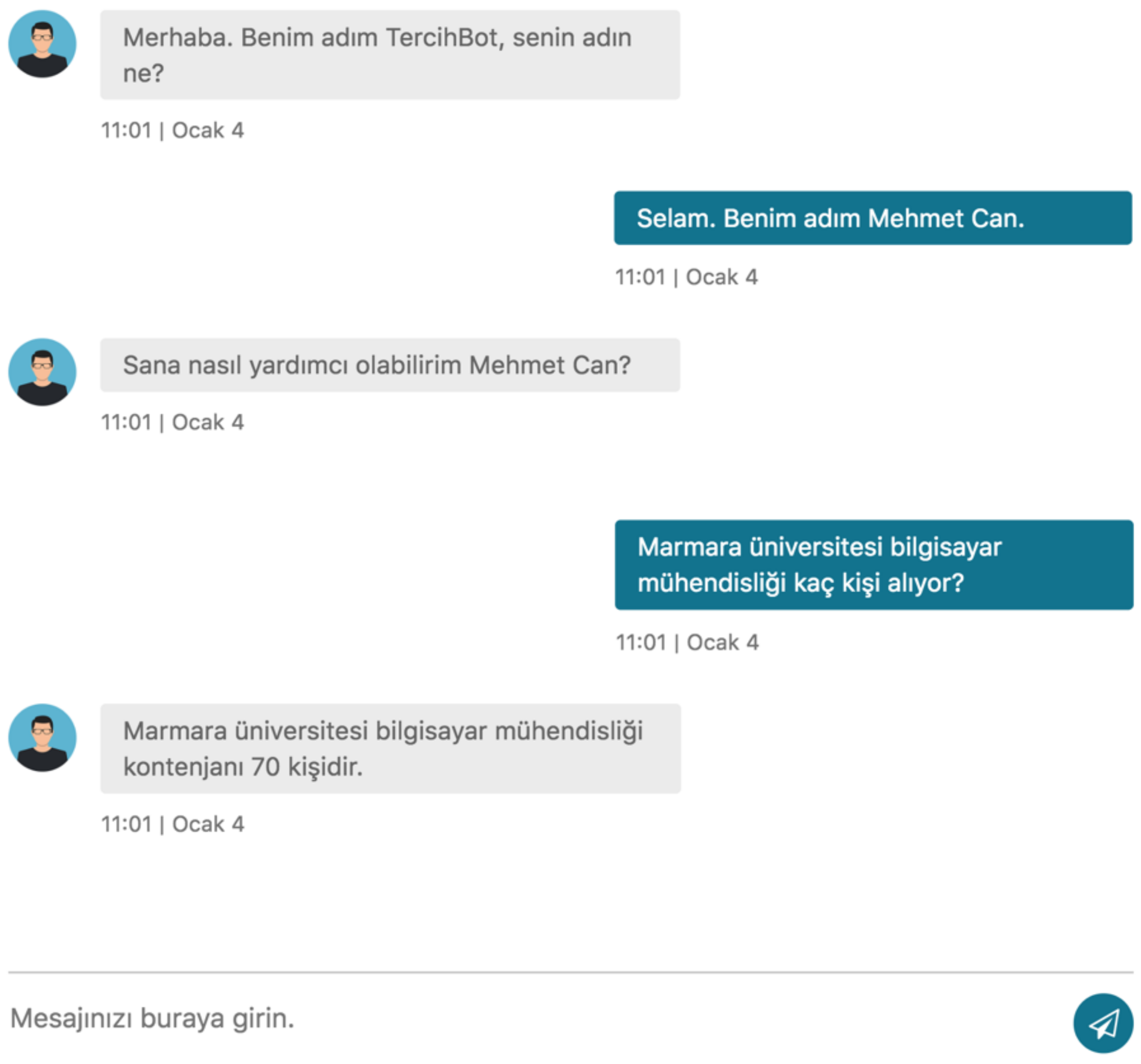


Figure 8 – Model View Controller architecture

When the user starts the system, he/she will see only one chat screen. Therefore, all operations will be done on this page. There will be three buttons for operations that are listed below.

- Submit
- End conversation (or the user can end the conversation via natural language)
- List frequently asked questions

An example of chat and UI is given in Figure 9.



Merhaba. Benim adım TercihBot, senin adın ne?

11:01 | Ocak 4

Selam. Benim adım Mehmet Can.

11:01 | Ocak 4

Sana nasıl yardımcı olabilirim Mehmet Can?

11:01 | Ocak 4

Marmara üniversitesi bilgisayar mühendisliği kaç kişi alıyor?

11:01 | Ocak 4

Marmara üniversitesi bilgisayar mühendisliği kontenjanı 70 kişidir.

11:01 | Ocak 4

Mesajınızı buraya girin.

Figure 9 – Example of chat with UI

4.1.4. Test Plan

The success of the system will be evaluated in 2 steps.

- ✓ Testing the API
- ✓ Testing the UI

4.1.4.1. Testing the API

Our chatbot API will contain both simple and complex question scenarios, so testing for these algorithms will be separated from each other.

For simple questions, we will implement unit tests in order to find out whether algorithms identify the question or not or whether the answering mechanism works correctly or not. However basic answer generation part requires ML algorithm for NER, so NER algorithm will be tested with F1 and accuracy metrics which are described further below.

Test case for basic question identification:

Test Case Name:	Basic Question
Requirements:	The system must identify the question and it's scenario (prepared question-answer tuple)
Procedure:	Asking a university related basic question such as quota or location
Expected Result:	The system identifies the scenario

Test case for basic answer generation:

Test Case Name:	Basic Answering Mechanism
Requirements:	The system must identify the entities and search database for required information
Procedure:	Asking a university related basic question such as quota or location
Expected Result:	For given question, the system returns the correct information in a proper format from database

However for complex questions, the answer is not strict and can be various. In order to measure the success of these scenarios, accuracy and F1 score metrics will be used on our trained models. Our data will be split into training (80%) and test (20%) sets. In other words, as 80% of the dataset is used for training the algorithms, the remaining 20% will be used for testing the model.

The accuracy of a machine learning model is the proportion of correct predictions. (Formula 1)

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions}$$

Formula 1 – Accuracy Formula

Even though the accuracy metric is enough for evaluating the prediction of the model, it is not a good predictor of success in an imbalanced dataset. Precision and recall are two metrics used to evaluate classifiers on imbalanced datasets. Precision (Formula 2) and recall (Formula 3) are found by using the given information at Table 1.

In this table:

- **True Positive:** Samples that are positive and predicted by the trained model as positive
- **True Negative:** Samples that are negative and predicted by the trained model as negative
- **False Positive:** Samples that are negative but predicted by the trained model as positive
- **False Negative:** Samples that are positive but predicted by the trained model as negative

Table 2 – Confusion Matrix

number of elements	Actual Positive	Actual Negative		
Predicted Positive	True Positive	False Positive	a	a + b = number of elements
Predicted Negative	False Negative	True Negative	b	
	x	y		
	x + y = number of elements			

$$Precision = \frac{\#\{True\ Positives\}}{\#\{True\ Positives\} + \#\{False\ Positives\}}$$

Formula 2 – Precision

$$Recall = \frac{\#\{True\ Positives\}}{\#\{True\ Positives\} + \#\{False\ Negatives\}}$$

Formula 3 – Recall

Both metrics are incomplete by themselves. Thus, in this phase, the F1 score (Formula – 4), which finds a balance between precision and recall, is used. In the manner, F1 score and accuracy will be checked for each algorithm that we used for retrieving an answer for complex questions. If F1 and accuracy values have a success rate over 70%, we will bring the system into service and ask the volunteers to vote the system. Then, we will update the data or make some arrangements on the algorithms by evaluating the rate of use and the comments.

4.1.4.2. Testing the UI

Test case #1:

Test Case Name:	Submit Button
Requirements:	The system must transmit the given input to the API and print the output to the chat screen
Procedure:	User presses the submit button
Expected Result:	The system prints the output to chat screen

Test case #2:

Test Case Name:	List Frequently Asked Questions
Requirements:	The system determines top 10 asked question
Procedure:	User requests frequently asked question list
Expected Result:	The system shows the frequently asked questions

Test Case #3:

Test Case Name:	Log the Conversation
Requirements:	The system shall ask the user for rating the system and brief comment about it
Procedure:	End the conversation
Expected Result:	The system stores the whole script of conversation

5. Software Architecture

5.1. Data Flow

Data flow for each database (Section 4.1.2) is given in Figure 10.

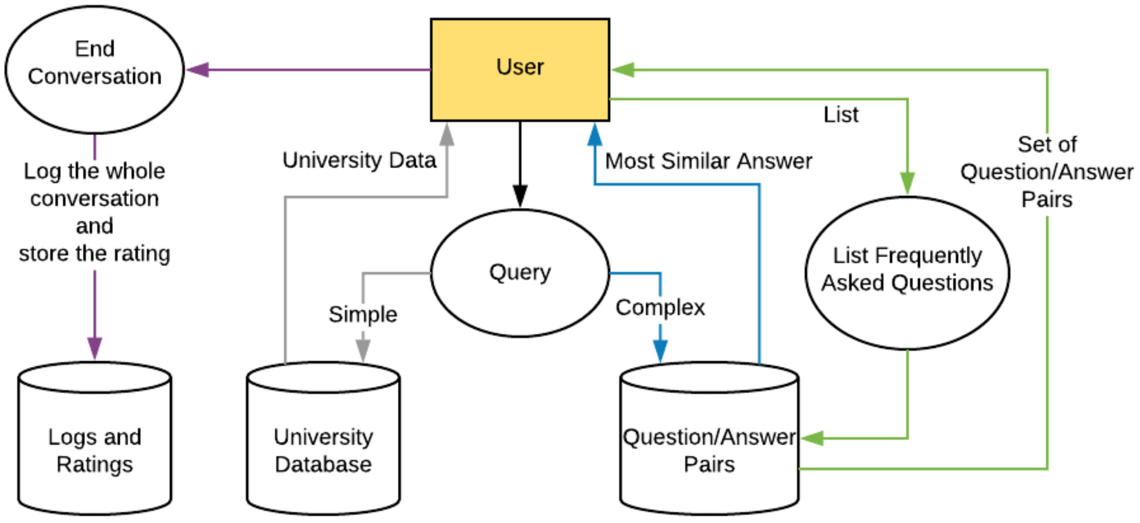


Figure 10 – Data Flow

5.2. Control Flow

The main control flow of our system is given in Figure 11.

When the user feeds the system with input text, the system will detect the abusive content and filter this content out of input. For this purpose, we will find a dataset of abusive words and we will filter these words out by searching the given input.

The user can feed the system with both information and queries. For this reason, the system will parse the sentences out of the input and will generate an answer for each sentence. For parsing sentences, we are planning to implement a rule-based algorithm that will identify the end of sentences through punctuation which is why the user needs to be careful with punctuation. For example:

Given input: —Merhaba, benim adım Can. Marmara Üniversitesi nerede?

the system will parse the all input text into:

Sentence one: —Merhaba, benim adım Can.

Sentence two: —Marmara Üniversitesi nerede?

However, this method is not our final decision. We are planning to look for a state-of-the-art NLP algorithm for this purpose.

With the pre-processed input, the system needs to detect whether the given input is whether a query or an information. Therefore, we have two different control flows after this state.

- ❖ Information (Section 5.1.1)
- ❖ Query (Section 5.1.2)

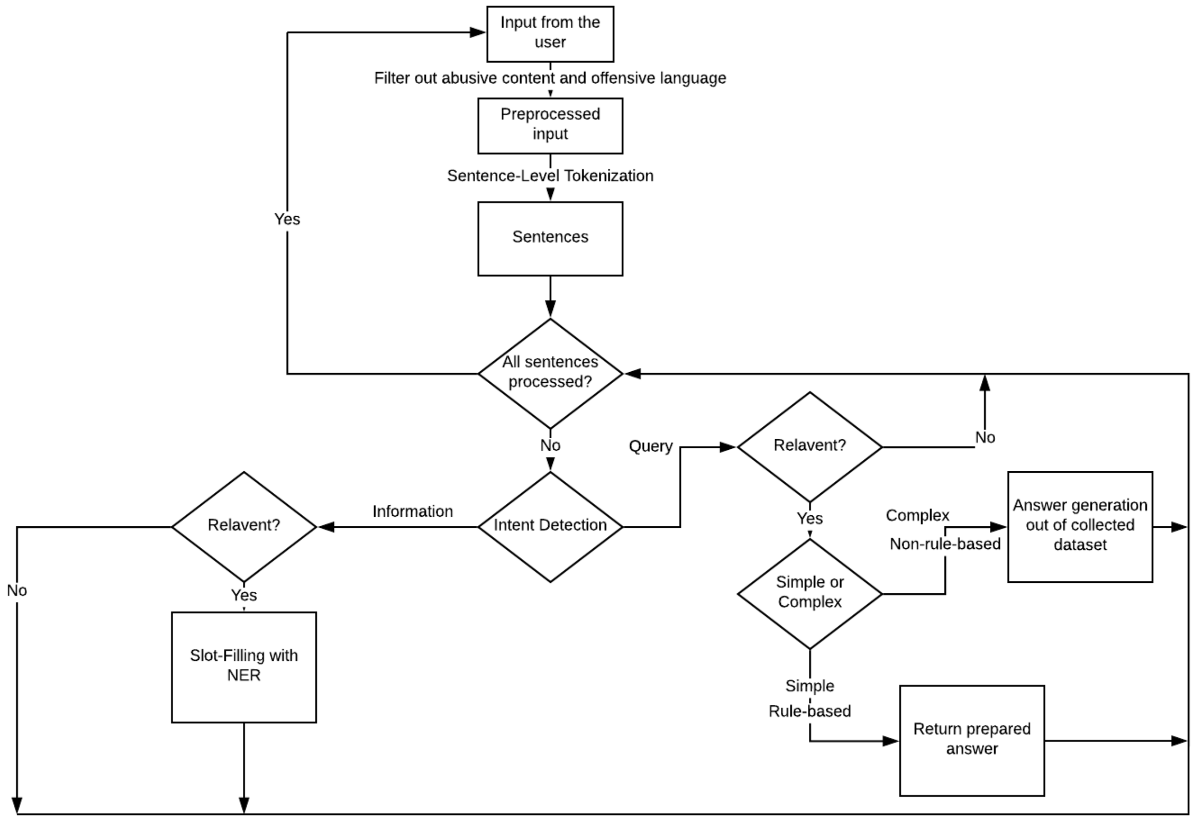


Figure 11 – Main control flow of the system

5.2.1. Information

If the detected intent is information, the system will fill the slots of specific fields such as the name of the user and the age of the user. After parsing the named entities, we will store that information. With this method, the system will remember the user's information during the whole conversation. Control flow of this process is given in Figure 12.

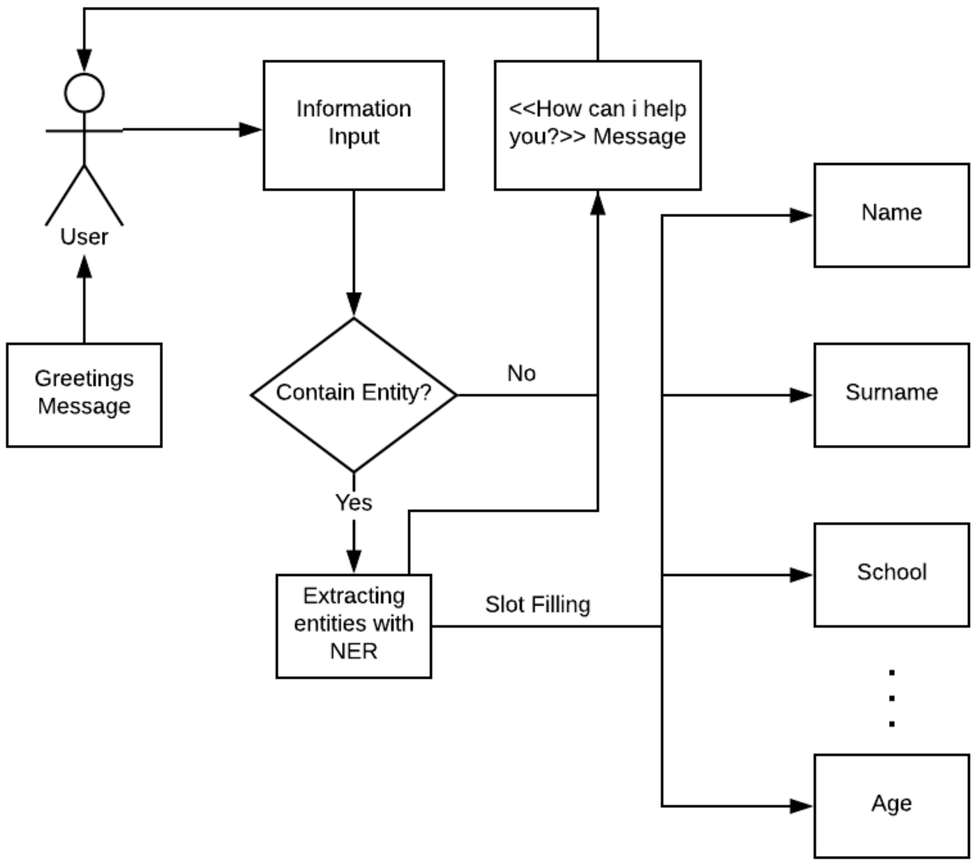


Figure 12 – Control Flow of Slot-Filling and Start of the Conversation

5.2.2. Query

If the detected intent is a query, the system will return the answer with two methods.

- ❖ Rule-Based Answers
- ❖ Complex Answers

For rule-based algorithms, the system starts with extracting the named entities such as the name of the university and name of the department. We will use NER for this extraction. After extraction of the entity, we will detect the question. The system will return previously prepared information (from universities websites) as a result for the user's question.

If the system cannot produce an answer via rule-based algorithms, the system will select the most similar answer for the input by looking at the database.

Control flow of query process is given in Figure 13.

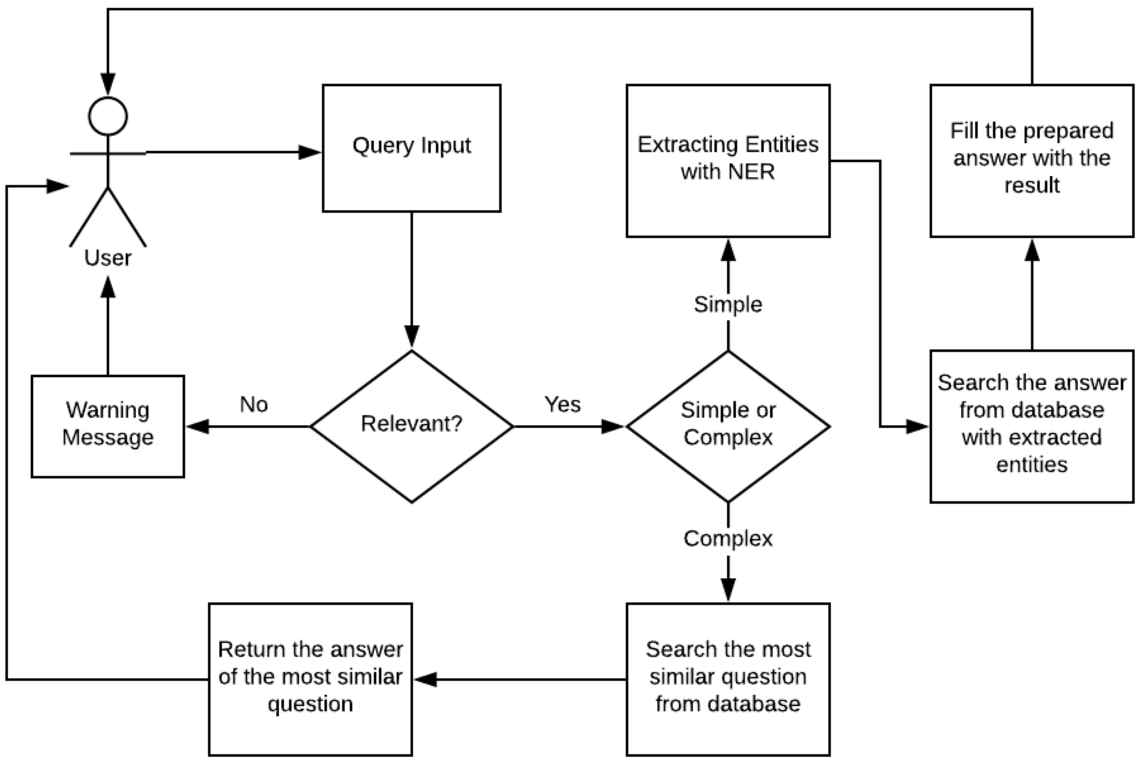

User
Query Input
Extracting Entities with NER
Fill the prepared answer with the result
Simple
Warning Message
No
Relevant?
Yes
Simple or Complex
Search the answer from database with extracted entities
Complex
Return the answer of the most similar question
Search the most similar question from database

Figure 13 – Control Flow of Query

5.2.3. Listing Frequently Asked Questions (FAQs)

When the user sends a request for listing FAQs, the system will find the top 10 asked question by connecting to the database.

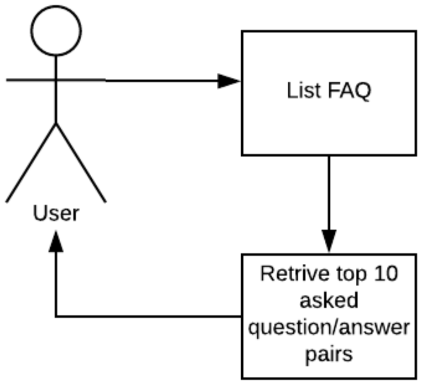


Figure 14 – Control Flow of Listing FAQs

5.2.4. Rating and Comment System

When the user ends the conversation, the system will ask for a brief comment and rating for the whole conversation.

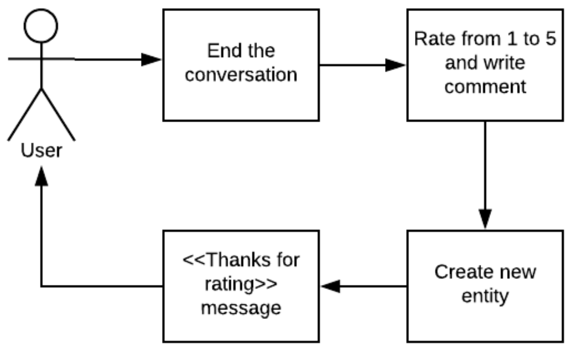

User
End the conversation
Rate from 1 to 5 and write comment
Create new entity
<<Thanks for rating>> message

Figure 15 – Control Flow of Rating/Comment System

5.3. Modular Design

The whole system layout (Figure 16) contains two layers:

- UI Layer (Front–End)
- Back End

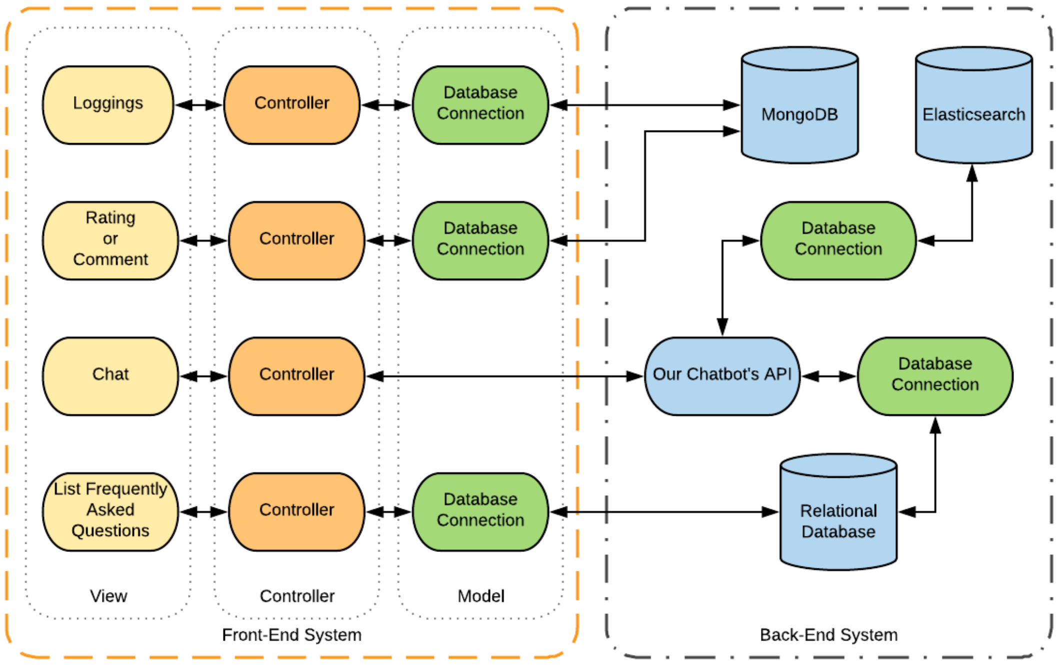


Figure 16 – Modular Design

Users will interact with our chatbot and receive rating/command via UI layer (Front–End). Interactions with chatbot will be done with communication of API and MVC’s controller. However, ratings/commenting and store operation for conversation’s script will be done directly by MVC’s model.
API operations will be done by the back-end.

6. Task Accomplished

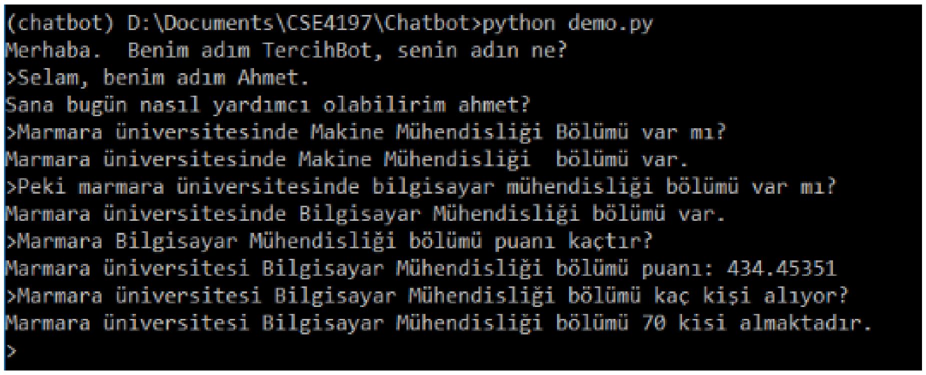
6.1. Current state of the project

In the current state of the project, these tasks were accomplished;

- ❖ Data about universities were collected and database (Elasticsearch) is created with this data.
- ❖ Data for complex questions were collected from forum sites however the data was inaccurate and immature so we will not use this data.
- ❖ Basic greetings scenarios were implemented by using the NLTK’s chatbot module which is similar to AIML. An example is given below:

```
 r'Merhaba, Benim adım (.*)',
(
    "Tanıştığıma memnum oldum %1, sana nasıl yardımcı olabilirim?",
    "Sana bugün nasıl yardımcı olabilirim %1?",
    "Seni burada görmek ne güzel, bugün sana nasıl yardımcı olabilirim %1?",
)
```

- ❖ Basic university questions were created. Example is given below:
  Question : X üniversitesindeki Y bölümünün kontenjanı kaçtır?
  Answer: X üniversitesindeki Y bölümü kotenjanı Z kişi almaktadır.
- ❖ Primitive NER were implemented via using fuzzy search that is provided by fuzzywuzzy (Python module)
- ❖ Demo created by using the aforementioned methods. (Figure 17)



```
(chatbot) D:\Documents\CSE4197\Chatbot>python demo.py
Merhaba.  Benim adım TercihBot, senin adın ne?
>Selam, benim adım Ahmet.
Sana bugün nasıl yardımcı olabilirim ahmet?
>Marmara üniversitesinde Makine Mühendisliği Bölümü var mı?
Marmara üniversitesinde Makine Mühendisliği  bölümü var.
>Peki marmara üniversitesinde bilgisayar mühendisliği bölümü var mı?
Marmara üniversitesinde Bilgisayar Mühendisliği bölümü var.
>Marmara Bilgisayar Mühendisliği bölümü puanı kaçtır?
Marmara üniversitesi Bilgisayar Mühendisliği bölümü puanı: 434.45351
>Marmara üniversitesi Bilgisayar Mühendisliği bölümü kaç kişi alıyor?
Marmara üniversitesi Bilgisayar Mühendisliği bölümü 70 kisi almaktadır.
>
```

Figure 17 – The screenshot of the demo

6.2. Task Logs

During this semester, we had meetings with our advisor on every two weeks. In these meetings, we reported our progress and discussed algorithms that we can investigate. Besides, on a weekly basis, we gathered for 20 minutes after our advisor's class hours and discussed the data that we can collect, the basic methods and the algorithms that we can use. Besides these meetings, we also gathered as a group when we can.

02.08.2018 – Thursday / 18:30-20:30: We attempted to meetup activity about "Chatbot Development Process in Business" at sahibinden.com's headquarter. Emre Yazıcı talked about Chatbots and techniques that are being used in the industry. It was a great experience for us to develop our own chatbot.

05.10.2018 – Friday / 8:30-9:30: We organized a meeting for selecting the specific topic of our Chatbot with our advisor Assoc. Prof. Dr. Murat Can Ganiz and Aydın Gerek who is a postdoctoral researcher at Marmara University. We decided to build our chatbot for guidance during the university selection period in Turkey.

17.12.2018 – Monday / 13:30-15:30: We organized a meeting for investigating the new algorithms for chatbots with our advisor Assoc. Prof. Dr. Murat Can Ganiz and Aydın Gerek who is a postdoctoral researcher at Marmara University. We decided to investigate ULMFiT, ELMo and BERT.

6.3. Task plan with milestones

6.3.1. Phases

Phase 1: Collecting more data from Twitter

Phase 2: Research about advance deep learning algorithms

- ULMFiT
- ELMo
- BERT

Phase 3: Sentence level tokenization

Phase 4: Slot filling with NER

Phase 5: Detection of offensive language and random sentences

Phase 6: Building an API

Phase 7: Developing test platform and test cases

Phase 8: Preparing the project‘s report and poster

6.3.2. Timelines

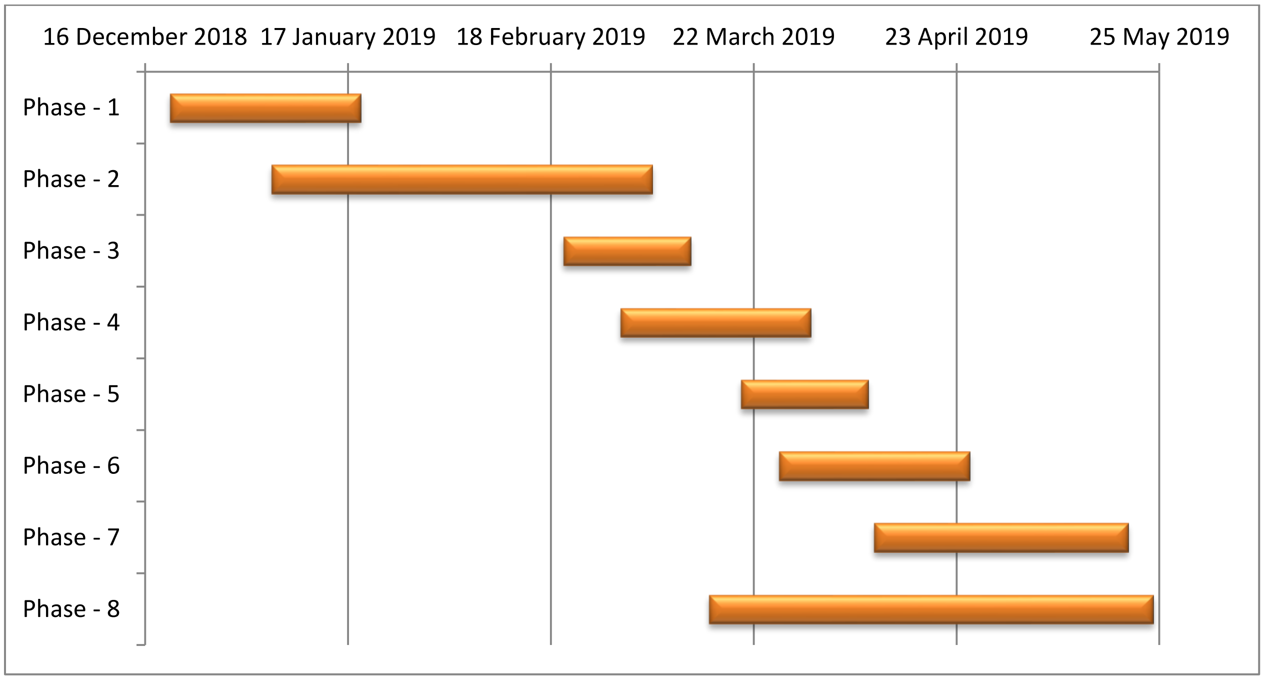


Figure 18 – Timelines

References

[1] BRADEŠKO, Luka; MLADENIĆ, Dunja. A survey of chatbot systems through a loebner prize competition. In: Proceedings of Slovenian Language Technologies Society Eighth Conference of Language Technologies. 2012. p. 34-37.

[2] WEIZENBAUM, Joseph. ELIZA—a computer program for the study of natural language communication between man and machine. Communications of the ACM, 1966, 9.1: 36-45.

[3] ABUSHAWAR, Bayan; ATWELL, Eric. ALICE chatbot: trials and outputs. Computación y Sistemas, 2015, 19.4: 625-632.

[4] SHAWAR, Bayan Abu; ATWELL, Eric. A comparison between Alice and Elizabeth chatbot systems. University of Leeds, School of Computing research report 2002.19, 2002.

[5] TOPÇU, S.; ŞEN, C.; AMASYALI, M. F. Türkçe Sohbet Robotu. Akıllı Sistemler ve Uygulamaları, ASYU, 2012.

[6] MUJEEB, Sana; JAVED, Muhammad Hafeez; ARSHAD, Tayyaba. Aquabot: A Diagnostic Chatbot for Achluophobia and Autism. INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS, 2017, 8.9: 209-216.

[7] MRINI, Khalil; LAPERROUZA, Marc; DILLENBOURG, Pierre. Building a Question-Answering Chatbot using Forum Data in the Semantic Space. 2018.

[8] MIKOLOV, Tomas, et al. Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. 2013. p. 3111-3119.

[8] GOLDBERG, Yoav; LEVY, Omer. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722, 2014.

[10] LIU, Haixia. Sentiment analysis of citations using word2vec. arXiv preprint arXiv:1704.00177, 2017.

[11] GÜNGÖRDÜ, Zealal; OFLAZER, Kemal. Parsing Turkish using the lexical functional grammar formalism. Machine Translation, 1995, 10.4: 293-319.

[12] ERYIĞIT, Gülşen; NIVRE, Joakim; OFLAZER, Kemal. Dependency parsing of Turkish. Computational Linguistics, 2008, 34.3: 357-389.

[13] ŞEKER, Gökhan Akın; ERYIĞIT, Gülşen. Initial explorations on using CRFs for Turkish named entity recognition. Proceedings of COLING 2012, 2012, 2459-2474.

[14] Donanımhaber.com, forum page for universities Available:
https://forum.donanimhaber.com/forumid_618/tt.htm (Date of Access: 10/10/2018)

[15] TURNEY, Peter D.; PANTEL, Patrick. From frequency to meaning: Vector space models of semantics. Journal of artificial intelligence research, 2010, 37: 141-188.

[16] Bojanowski, Piotr; Grave, Edouard; Joulin, Armand; Mikolov, Tomas. fastText. 2016. Available: https://research.fb.com/fasttext/ (Date of Access: 20/10/2018)

[17] OFLAZER, Kemal. Two-level description of Turkish morphology. Literary and linguistic computing, 1994, 9.2: 137-148.

[18] TURING, Alan M. Computing machinery and intelligence. In: Parsing the Turing Test. Springer, Dordrecht, 2009. p. 23-65.

[19] BOCKLISCH, Tom, et al. Rasa: Open source language understanding and dialogue management. arXiv preprint arXiv:1712.05181, 2017.

[20] BORDES, Antoine; BOUREAU, Y.-Lan; WESTON, Jason. Learning end-to-end goal-oriented dialog. arXiv preprint arXiv:1605.07683, 2016.

[21] DENG, Jia, DONG, Wei, SOCHER, R., LI, Li-Jia, LI, Kai and FEI-FEI, Li. ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition. 2009.

[22] HOWARD, Jeremy; RUDER, Sebastian. Universal language model fine-tuning for text classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2018. p. 328-339.

[23] MERITY, Stephen, et al. Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843, 2016.

[24] PETERS, Matthew E., et al. Deep contextualized word representations. arXiv preprint arXiv:1802.05365, 2018.

[25] DEVLIN, Jacob, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[26] RAJPURKAR, Pranav, et al. Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250, 2016.