# CMPE 492 Final Report
# A Turkish Chatbot For Administrative Inquiries in Bogazici University

Advisor: Tunga Güngör

Elifnaz Utkan, Ozan Kınasakal

October 2018

## Contents

# 1 Introduction and Motivation

Chatbot, also known as chatterbot, interactive agent, Artificial Conversational Entity, or just bot, is an artificial intelligence software or a computer program that simulates a conversation with users in natural language.[1]

Chatbot technology has improved with the help of natural languages processing and machine learning techniques. The techniques enable to understand the intent of the user better and to provide more accurate and relevant response, so it can emulate a human dialogue better.

Mobile messaging applications have more than 2 billions users worldwide now and as shown in Figure 1 more and more users are using messaging apps in comparison with social networks.[2][3] In addition, they have higher usage rates and retention than most mobile applications. Especially after big companies such as Facebook and Microsoft provided resources that allow the integration of chatbots into their messaging platforms, many chatbots are being developed for these mobile messaging applications and also for other messaging platforms such as Slack, Kik and WeChat.

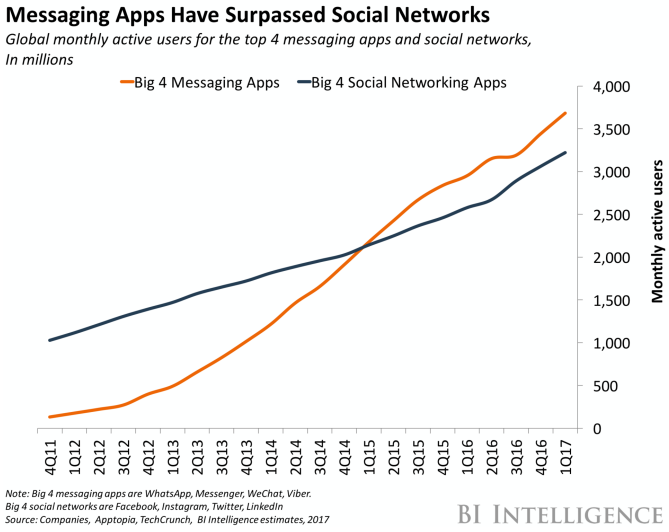



Figure 1: Active users for top 4 social networks and messaging apps. Source: https://www.businessinsider.com/the-messaging-app-report-2015-11

Chatbots can serve as virtual assistants to help users to perform specific tasks such as searching the web, setting up appointments, answering the most frequently asked questions. Mostly people find the websites hard to navigate and complain about not getting answers to simple questions quickly.[4] By solving these problems, chatbots improve online experiences for users.

Chatbot has become more popular in business groups right now as they can reduce human efforts and customer service cost by automating it and handle multiple users at a time. Many companies are starting to build a chatbot for their own needs. According to Chatbot Survey 2017, customer service function will benefit the most from chatbots followed by sales/marketing and order processing.[5]

In our project we built a chatbot for educational environment to answer FAQs related to rules and regulation regarding Computer Engineering department in Bogazici University. We observed that the same questions are asked by students again and again. People tend to ask these questions on Whatsapp or Facebook groups rather than trying to find the information on the website of the department. At the end we linked our chatbot with Telegram so that students can use it easily.

# 2 State of the Art

In order to create a chatbot first we have to understand the different approaches and methods that are used previously. Therefore we had to make quite a bit of research. We started with reading "A Corpus Based Approach to Generalising a Chatbot System" by Abu Shawar and "Designing Bots: Creating Conversational Experiences" by Amir Shevat. We will introduce some of the most important chatbots throughout the history which were mentioned in these books and also in various academic papers.

## 2.1 Chatbot History

### 2.1.1 ELIZA

Eliza was designed to imitate a psychotherapist. It was developed by Joseph Weizenbaum in 1966 in Massachusetts Institue of Technology. Eliza was not using any machine learning algorithms and it was not considered as very intelligent. However it was pretty successful at impressing the people with it's intelligence, a

conversation with Eliza can be seen in Figure 2. Eliza was looking for the keywords in the user's input, then it was applying simple transformations such as replacing the word "you" in the input sentence with "I" in the answer[6]. Then it was creating basic and general responses.

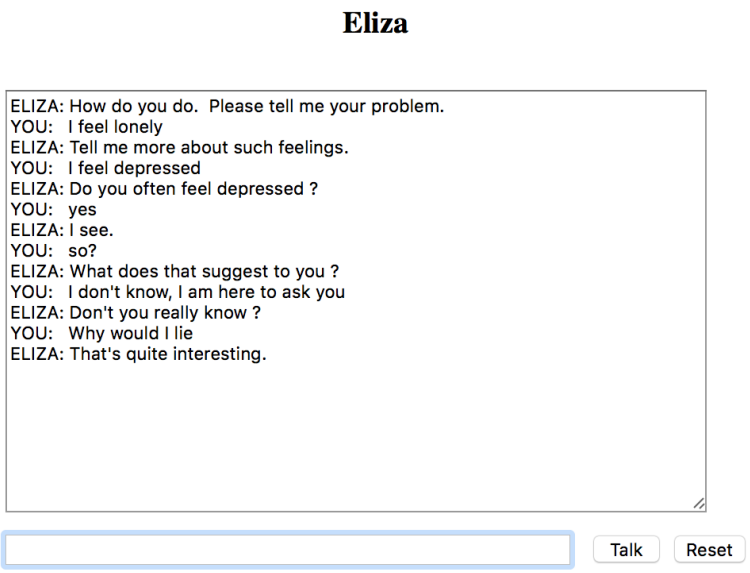



Figure 2: A conversation we had with Eliza.(https://www.masswerk.at/elizabot/)

### 2.1.2 PARRY

Parry was another early chatbot example which was built by Kenneth Colby in 1972. It was designed to simulate a paranoid patient. Parry was using similar techniques to Eliza. It was also as effective since the doctor's who were given the transcripts of conversations with Parry, could not distinguish them with the real conversations conducted with patients. The input text was portioned into patterns and compared for similarities with the semantic patterns stored in the database.[7]

### 2.1.3 ALICE

Artificial Linguistic Internet Computer Entity(ALICE), developed by Richard Wallace, is a chatbot using pattern matching algorithms which are specified using AIML(Artificial Intelligence Markup Language). Before applying the pattern

matching algorithms the input sentence is normalized and an input path is produced from each sentence. It was easy to configure ALICE as your special chatbot using the AIML tags. ALICE has won the Loebner Prize[8] three times.

```
<category>
<pattern>MY NAME IS *</pattern>
<template>Nice to meet you, <set name="name"><star /></set></template>
</category>
```

Figure 3: An example of AIML code

There are other bots that are designed with using AIML as well. A contemprory example is another multi-award winning chatbot Mitsuku[9].

### 2.2 FAQ Chatbots

One of the related works we found while we were doing research was FAQchat. The FAQchat system was produced by using the FAQ in the School of Computing at the University of Leeds to retrain the ALICE system. The working principle of this system is explained as follows: "In retrieving information FAQchat will try to give the results using most significant words as keywords, and try to find the longest pattern to match without using any linguistic tools, or analyzing the meaning."[10] Results of this chatbot project were satisfying. Students who used the system preferred FAQChat over Google because it was able to give direct answers and number of the links it provided was less than Google's.

## 3 Methods

### 3.1 Flow

Once a user writes a question that sentence has to be passed through different stages as seen in Figure 4. The expected scenario for the flow is:

1. A client writes to Chatbuddy using Telegram.(Telegram username: @bogazicichatbot)

2. Telegram server receives the message and deliver it to Chatbuddy's web-hook.

3. Java process running on Chatbuddy's Server receives the message. JSON formatted message is parsed and extracted text is passed on to methods for NLP.

4. Text is convert into lower case and all punctuation are removed. Then it is corrected using the deasciifier and the stem of each word in the text is extracted from the morphological analysis of the sentence.

5. A POST request is made to the Python process running in the same server. JSON formatted array of stems is passed in the body of the request.

6. Array of stems is turned into a 0-1 array over the unique words in corpus using Bag of Words model.

7. Result of BoW array is given into the DNN model and prediction is made.

8. Predicted tag and it's probability is returned in JSON format as a response to the POST request.

9. End-user response is retrieved for the predicted tag from our initial data. Then POST request is made to Telegram server's sendMesssage endpoint with our response in the body.

10. Telegram servers receive our request and deliver the message to the client.

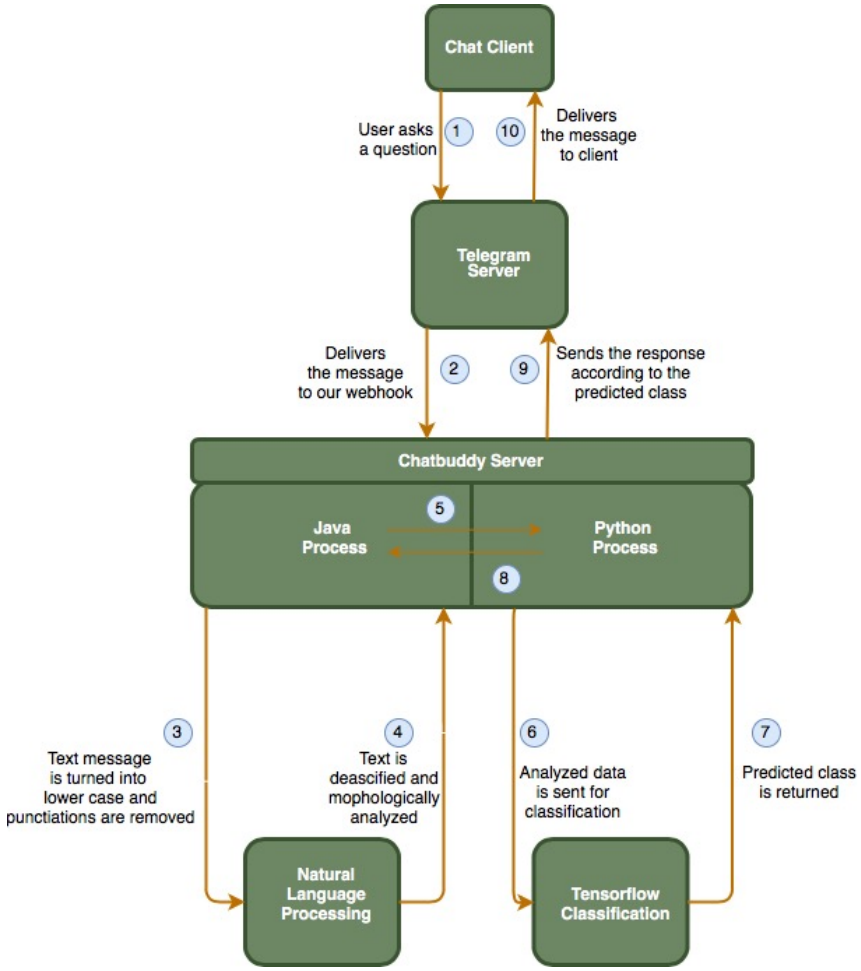



Figure 4: Flow of our system.

## 3.2 Architecture and Design

### 3.2.1 Main Processes

In our system there are two processes(Java and Python) running on the same machine but on different ports(Java runs on 8443, Python runs on 5000). NLP operations and communication with Telegram API is handled in Java process whereas classification, predictions and training is done in Python process. These two pro-

cesses communicate with each other using REST API calls. We used "Jetty" and "Flask" libraries for serving the programs.

There are two endpoints for Python code: "/train" and "/classify" . A GET request is made to the former endpoint manually in order to train the model when the processes are started for the first time(It can be done with either using Postman or directly from the browser). Of course it can be triggered later to train the model again(maybe if some new data is added). Then the model will be trained and saved into the same folder which is to be retrieved later for the classification.

A POST request is made to the latter endpoint from the Java process, each time a new message is received by our server. The array of stems are sent in the request body. Then this array is turned into a 0-1 array compared to unique words in our corpus using Bag of Words model. It is given to to the model as in order to predict the correct tag which is then returned in response with its prediction probability.

For the Java part we created an endpoint called "/Telegram" and used it as a webhook to receive new updates from Telegram API. Whenever the endpoint is triggered it will parse the JSON formatted request body and extract the "text" field of Message object and "id" field of Chat object. We need the chat ID value to be used later when sending a message back to user. After the text passes through NLP and classification parts a predicted tag is returned from the Python process. Then using this tag "/sendMessage" endpoint of Telegram API is called with the initial chat ID of the conversation and our response for the predicted tag in the requst body. Telegram delivers our message to the chat client.

### 3.2.2 Telegram API

In our midterm progress report we mentioned to integrate our chatbot into one of the popular messaging applications(Messenger, Telegram, Slack etc.). We decided to use Telegram since it did not require any waiting time or reviewing process for our bot. As we make the connection with Telegram we would directly be able to use it on production.

Telegram has a well documented API[11]. We had to set up a webhook for Telegram to deliver us the updates(new messages). So that whenever someone writes to our chatbot via Telegram, our endpoint will be triggered and Java process will handle the request. To do that we deployed our code to a server using

AWS EC2 service. We configured the environment and run the two processes on server. Telegram wants a "https" endpoint to set a webhook. So we created a self-signed certificate and used reverse proxy to show a valid endpoint to Telegram. We used "nginx" for the reverse proxy so that our application listens the requests from telegram on 443 port with a valid https endpoint. It delivers all the requests to 8443 port which is the http endpoint for our Java process.

### 3.2.3 NLP

For the morphological analysis and deascifying operations in NLP engine, first we tried using ITU Turkish Natural Language Processing Pipeline[12]. But there were some problems about it. First of all it was costly to send a request each time we get a new message from a user and we had to wait for the result in order to pass the text. The other reason was it was not stable. Sometimes we experienced that ITU server did not response. Therefore we decided to use Zemberek-NLP[13] library, so we could consistently get the expected results of our program.

Pre-processing steps for our data:

1. Punctuations are removed from the given input sentence
2. Sentence is turned into lower case letters
3. Sentence is deascifiied using Zemberek-Normalization package.
4. Morphological analysis is applied using Zemberek-Morphology package.
5. From a single analysis, stems are taken in an array for each word.

## 3.3 Data Collection

First we examined the rules and regulations both in our departments website and in school's general website. Based on these information we determined around fifteen categories as our classes such as "repeat", "senior", "graduation", "add/drop", "registration" etc. Now we had to find appropriate questions regarding these categories. To find enough data to build our chatbot, we used different methods. We started with analyzing FAQs on the website of the department. But there were only training related questions. Another source of data was WhatsApp groups that were created by computer engineering students. There is a group for each year where students communicate with each other and get information. People are asking very

similar or even the same questions regarding the regulations in these groups. We extracted the group chat histories and collected questions related to each category. We collected more rules and regulation related questions from Facebook groups that are used by many students from different departments. In addition, collecting data by talking to students was another method we used.

We increased the data size by producing new sentences from the collected ones. We used these strategies for increasing the data size:

- Using synonyms for some words
- Variations of multi-word expressions
- Asking the same question in a more polite way
- Creating imperative form of the sentence

After dumping this data we shaped it so that for each tag(intent) there were patterns(previously asked questions) and responses which consist of related articles of the rules and regulations.


```
{
  "tag": "5xx",
  "patterns": [
    "500 kodlu dersleri lisans öğrencileri alabiliyor mu?",
    "500 lu dersleri alabilmek icin senior mi olmak lazim?",
    "Lisans öğrenciyim. 500 kodlu bir ders alabilir miyim?",
    "500 kodlu dersleri alma koşulu ne?",
    "3. sınıf birinin 500 kodlu ders alması için consent atması
      yeterli mi yoksa dilekçe mi yazılması gerekiyor?"
    ...
  ],
  "responses": [
    "5xx kodlu dersleri sadece son sınıf olan ve GNO>=2.50 olan
      öğrenciler alabilir.Özel durumlarda ilgili yönetim kurulu
      karar almaya yetkilidir."
  ]
}
```


Figure 5: A small sample from our data.

In Figure 5 a sample data with the tag name "5xx" is shown. Normally there are around twenty questions for each tag. The responses are single line answers for each tag.

## 3.4 Models

To answer the questions correctly, we needed to find which class the asked question belongs to. So it can be modeled as a classification problem. We used to main models in order to solve this classification problem.

### 3.4.1 Naive Bayes

First we implemented the system using the Naive Bayes model. The system we developed is word-based, meaning that the classification is based on the frequencies of the words.

In the training part the word probabilities for each tag by dividing the number of times the words appears in sentences of the tag by total number of words in sentences of the tag and the tag probabilities are calculated.

We used Laplace Smoothing method in order to overcome the problem caused by multiplying with zero probability values.

In the testing part for each sentence *s* in the test set, the predicted tag *t* is determined by using Eqn.(1)

$$p(t|s) = \frac{p(s|t) * p(t)}{p(s)} \cong p(s|t) * p(t) = (\prod_{i=1}^{n} p(w_i|t)) * p(t) \qquad (1)$$

where *n* denotes the number of words in the sentence and $w_i$ denotes the *i*th word. We use the independence assumption such as each word $w_i$ is independent of other words in that sentence. For a sentence we calculated the probabilities of belonging to the tag *t* for each tag and the one with the highest probability is chosen as a predicted result.

### 3.4.2 Deep Neural Network

The second model is the Deep Neural Network model. We used TensorFlow, an open source software library used for numerical computations, to implement the model.[14]

In this model we used an additional method during the preprocessing of the data to prepare it as the TensorFlow input. We created bag of words array for each sentence which contain 0s or 1s depending on whether the corpus word appears in the sentence or not.

After doing several experiments by changing the features of the model, the trained model with one hidden layer with 32 nodes for 50 epochs gave the best testing results.

We used softmax activation to find the probabilities of the predicted results. As it is shown in Figure 6, in our test results more than 98.6% of correct predictions were above the 0.25 probability value. This value is determined as threshold to eliminate irrelevant questions. Since user may ask any question, belonged to one class or not, we used the threshold for answering only to tag related questions.

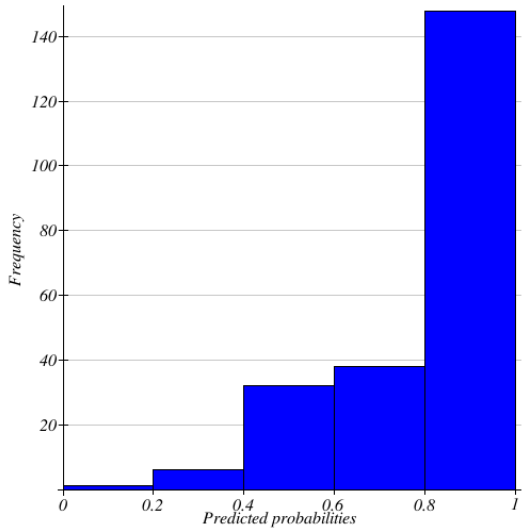



Figure 6: Frequencies of correctly predicted probabilities.

# 4 Results

We used an automatic evaluation as an evaluation metric. We trained our model using 90% of the dataset as the training data and we tested remaining 10% of the dataset. In the automatic evaluation we calculated accuracies and 10-fold cross-validation to evaluate the models. The results are shown in the Table 1.

Table 1: Summary of classification results

| Classification accuracy, by classifier | |
|---|---|
| Naïve Bayes | 75.94% |
| Deep neural network | 87.89% |

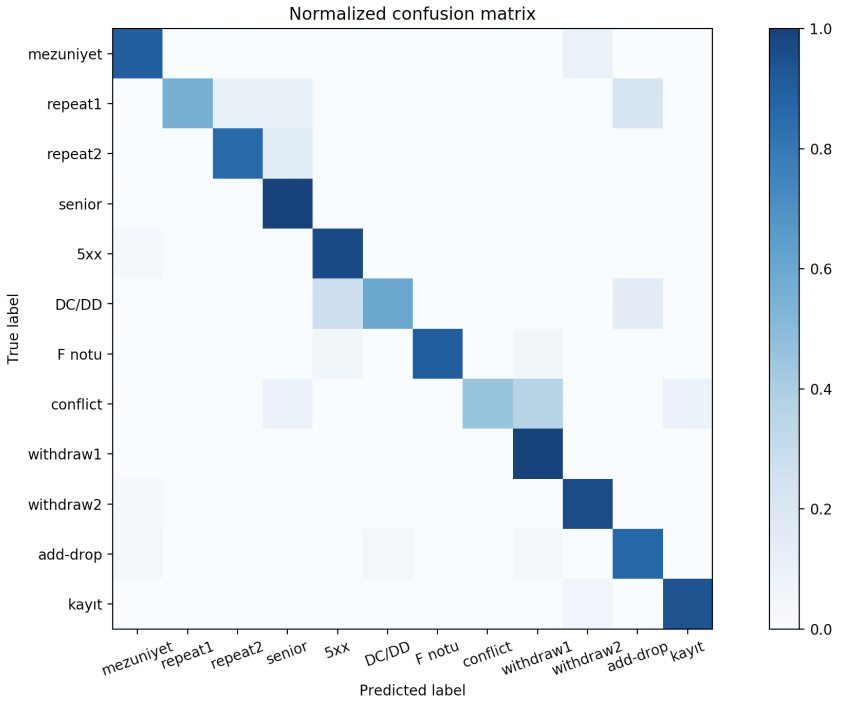
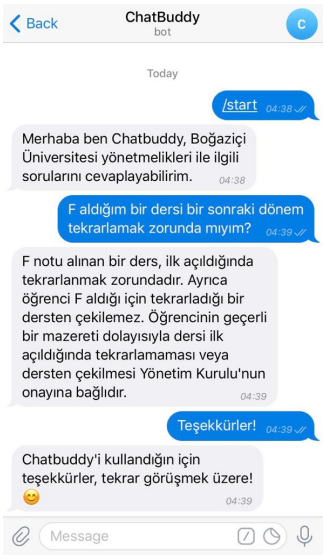


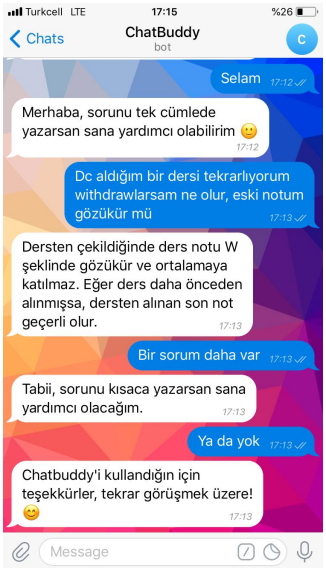

Figure 7: Confusion Matrix for the DNN model.

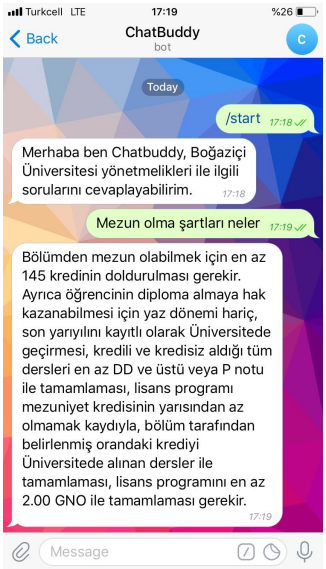
We also created a confusion matrix for our model which is shown in Figure 7.

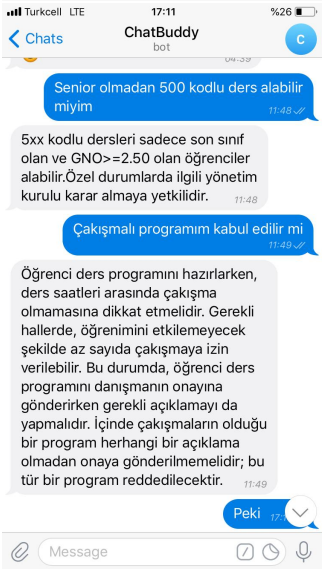
The evaluation results are only based on the collected data. The classes such as greeting, ending which are added to make the program user-friendly and they enables to create more fluent conversations with users are added after integrating our chatbot into Telegram. These additional classes are excluded, when we evaluated the system.

## 4.1 Sample Conversations



# 5 Conclusion and Discussion

We implemented a chatbot using Naive Bayes and DNN models. By examining the evaluation results, we observed that the DNN model gives the best results. Therefore we decided to integrate our chatbot into the popular messaging application Telegram by using this model.

There was no pre-collected data for this topic. We collected all data we used in this project. Therefore we had a relatively small dataset compared to a regular dataset used in the classification problem.

# 6 Future Work

Since we integrated our chatbot into a messaging application, now we are able to collect new sentences which can be added to our dataset. We are planning to add new questions to the database, if user is satisfied with the answer. There could be a voting system at the end of each conversation to get a feedback from users.

We observed that when people play with our chatbot they usually tend to ask many different questions about their specific classes, schedules etc. So the bot could be improved for answering some user specific questions as well.(by using conditional design for some parts of the bot) Also number of tags can be increased in this way. There are many more tags that we don't have in our dataset such as "erasmus", "internships", "graduate".

An improved version of this system may be used in Bogazici University Student Affairs Office. With their insight and thoughts on this topic bot can be modified to answer other questions, eventually it may reduce the traffic of incoming questions to humans and improve the performance.

# 7 References

# References


[1] SCHLICHT, Matt. The Complete Beginner's Guide To Chatbots [online]. 2016

[2] Statista. Number of mobile phone messaging app users worldwide from 2016 to 2021 (in billions). https://www.statista.com/statistics/483255/number-of-mobile-messaging-users-worldwide/

[3] The Messaging App Report, Business Insider. https://chatbotsmagazine.com/the-complete-beginner-s- guide-to-chatbots-8280b7b906ca

[4] The 2018 State of Chatbots Report: How Chatbots Are Reshaping Online Experiences. https://mk0drift0ho9g7wbfexi.kinstacdn.com/wp-content/uploads/2018/01/2018-state-of-chatbots-report.pdf

[5] Global Chatbot Trends Report – 2017. http://mindbowser.com/chatbot-market-survey-2017/

[6] Shah, H. , Warwick, K. , Vallverdú, J. and Wu, D. (2016) Can Machines Talk? Comparison of Eliza with Modern Dialogue Systems. Computers in Human Behavior, volume 58 : 278–295

[7] Shawar, Bayan Abu. Corpus Based Approach to Generalizing a Chatbot System. La Lambert Academic Publ, 2011.

[8] Loebner Prize - AISB. https://www.aisb.org.uk/events/loebner-prize

[9] Mitsuku, a four-time winner of the Loebner Prize Turing Test. https://www.pandorabots.com/mitsuku/

[10] Abu Shawar, B., Atwell, E., and Roberts, A. (2005). FAQChat as an information retrieval system. In Vetulani, Z., editor, Human Language Technologies as a Challenge. Proceedings of the 2nd Language and Technology Conference, Wydawnictwo Poznanskie, Poznan, Poland, pages 274–278.

[11] Telegram Bot API https://core.telegram.org/bots/api

[12] ITU Turkish Natural Language Processing Pipeline. http://tools.nlp.itu.edu.tr

[13] Zemberek-NLP. https://github.com/ahmetaa/zemberek-nlp

[14] TensorFlow. https://github.com/tensorflow/tensorflow