**Final Year Project**

**FYP Final Report**

**Interactive Chatbot for Admission Inquires in Roman Urdu**

**FYP Team**

**Okasha Khan**

**Usman Zafar**

**Zain-Ul-Abideen**

**Supervised by**

**Mr. Umair Arshad**

**Co-Supervised by**

**Dr. Mirza Omer Beg**

**FAST School of Computing**

**National University of Computer and Emerging Sciences**

**Islamabad, Pakistan**

**2021**

Table of Contents

[2. Introduction 4](#_Toc76552344)

[2.1. Problem Domain 4](#_Toc76552345)

[2.2. Research Problem Statement 5](#_Toc76552346)

[3. Literature Review 6](#_Toc76552347)

[3.1. Research Item # 1 6](#_Toc76552348)

[3.2. Research Item # 2 9](#_Toc76552349)

[3.3. Research Item # 3 12](#_Toc76552350)

[3.4. Research Item # 4 15](#_Toc76552351)

[3.5. Research Item # 5 17](#_Toc76552352)

[3.6. Comparison of Research Papers 20](#_Toc76552353)

[4. Proposed Approach 21](#_Toc76552354)

[4.1. Architecture of proposed solution 21](#_Toc76552355)

[4.2. Data Collection 23](#_Toc76552356)

[4.3. Data Cleaning 24](#_Toc76552357)

[4.4. Intent Generation 24](#_Toc76552358)

[4.5. Development of Web-based User Interface 24](#_Toc76552359)

[5. Implementation of proposed solution 25](#_Toc76552360)

[5.1. Data Collection (Scrapping and Manual Collection) 25](#_Toc76552361)

[5.2. Data Cleaning 27](#_Toc76552362)

[5.3. RASA 1.0 Installation and Migration to RASA 2.0 28](#_Toc76552363)

[5.4. Intent Generation and Classification 28](#_Toc76552364)

[5.5. Follow up Questions and Response Generation 29](#_Toc76552365)

[5.6. Development of Web based User Interface 32](#_Toc76552366)

[6. Results and Discussion 33](#_Toc76552367)

[7. Conclusion 36](#_Toc76552368)

[8. References 38](#_Toc76552369)

# Introduction

## Problem Domain

“One of the greatest desires of an organization in today’s world is to enhance its overall efficiency and customer satisfaction by spending as less resources as possible and getting as much output as possible in return. One of the reasons that organizations fail to achieve effectivity and customer satisfaction is information inadequacy. Information is one of the most essential resources in modern times, as it guides human thinking, planning and subsequent actions, which in turn generates consequences that are desired or not. Lack of needed information generates undesirable consequences and leads to more complaints and dissatisfaction from clients.

Major cause of information inadequacy is that information exists but cannot be found or delivered, or information transmission is delayed. To handle these problems, chatbot is one of the best solutions which can help both, organizations, and clients achieve their goals more effectively. A chatbot can automate customer support for similar queries, save human resources for qualitative tasks, accelerate operations, give better user interaction, and are easy to use, cost effective, and time efficient. A chatbot is available 24/7, and helps increase user satisfaction because of its ability of having low response time, being able to talk to multiple people simultaneously, supporting multiple languages and automating routine tasks.

Building a contextual, interactive, and intelligent chatbot that can handle dynamic queries and generate dynamic responses is not an easy task. It requires computer to understand human language. Computer must understand human language’s semantics, syntax, and pragmatics etc. Moreover, everyone has a different way of typing a message and not all the users can follow same format for writing queries.

A chatbot talking to a human being, which is a completely unique individual, raises problems like handling different usage of slangs, habit of misspelling certain words, usage of short forms and cool words etc. Natural Language Processing is not yet developed enough to be able to handle conversation in resource-poor local languages like Roman Urdu. Users are not consistent with their language and their goals. It is very difficult to predict everything a user can ask and every way a user can ask a question in. So, handling randomness of a human being is also a major problem for a chatbot developer. A user expects a chatbot to be as close to human intelligence as possible for a better experience. The most difficult task for a chatbot is to understand the intent of the user (what the user is looking for) and remember the context of the whole conversation.

All the limitations discussed above make it difficult to develop a chatbot that can facilitate its users effectively and efficiently specially in resource poor languages like Roman Urdu.”

## Research Problem Statement

Most of the existing chatbots have poor conversational understanding and they understand only a limited number of static queries and generate only a limited number of static responses. They only know what they are taught. Most of them don’t understand human context and it is a massive gap that irritates the users. Chatbot built using traditional approaches, are just hard-coded rule-based templates and rules to generate responses. Some chatbots have been developed for universities in the past but they have many flaws. Some of the flaws include chatbot being too rigid by rules for users to make inquiries, long latency between services, chatbot using frameworks that have some request limits, chatbots having no database to generate a variety of responses and depending on only APIs.

Most of the reliable chatbots developed till date are for resource-rich languages and rely heavily on already existing Natural Language Processing methods for English and some other limited number of resourceful languages. For a chatbot to be able to understand human language, it must understand human language’s semantics, syntax, and pragmatics etc. which is too difficult for a resource-poor language because there are too limited annotated resources available for these languages. Effectively incorporating these limited and unannotated resources to improve the performance of resource poor NLP is a difficult research problem.

Roman Urdu is a resource poor language due to which, there exists no chatbot in Roman Urdu that can understand dynamic queries of users and generate dynamic responses remembering the context of the conversation.

Our research problem is to build an AI based interactive, flexible, contextual, and intelligent chatbot that understands students’ queries written in Roman Urdu about FAST and admission process of FAST, and generate appropriate dynamic responses. Chatbot needs to:

* Understand and handle queries written in any format.
* Handle spell variations.
* Detect offensive language.
* Identify and handle out of scope queries.
* Ask follow up questions in case of incomplete information in user query.
* Extract required information from the query for processing and dynamic response generation, and
* Remember the context of the conversation.

To the best of our knowledge, our project is the first project that focuses on building an AI based interactive, contextual, and intelligent chatbot that handles students’ queries written in Roman Urdu.

# Literature Review

## Research Item # 1

**“Indonesian chatbot for university admission using a question answering system based on sequence-to-sequence modal”**

* + 1. Summary of the research item

In this research paper they have developed a chatbot which is based on a sequence-to-sequence model with attention mechanism. User asks a question about Telkom University SMB from chatbot. Chatbot searches for a similar question from existing data (previous conversation history) using sequence to sequence modal along with attention mechanism. The best answer to this question is searched and sent to the user. The input to this system is the question sentence about Telkom University SMB and the output is the answer.

Benefit of the attention mechanism is that, it does not lose information of the sentence (query asked by the question), particularly the information contained in the words that create a sentence. A learning rate of 0.001 and three different number of neurons (100 neurons, 200 neurons, and 300 neurons) were used to train a sequence-to-sequence modal, with attention mechanism and then without attention mechanism. A two layered LSTM is used as encoder and decoder. Input sentence is encoded into a context vector by encoder and then this context vector is decoded into target sentence (answer/response) by decoder.

In the model that uses sequence-2-sequence combined with attention mechanism, words, summarized into a context vector, are annotated by encoder using a bidirectional LSTM. As this model has forward as well as backward LSTM, it considers both previous and next words. Only one context vector is used by the model which is not combined with attention mechanism. The context vector depends on the annotated word (*h*1, ..., *ht*) mapped by the encoder from the input sentence.

A small conversation obtained from Telkom University Admission (SMB)’s WhatsApp In Bahasa Indonesia (Indonesian language), was used as a dataset. Input(query) sentences and target(response) sentences were in the train set and were fed to the sequence-to-sequence modal. Preprocessing techniques like lowercasing, punctuation removal, and tokenization were applied on these sentences. To get enough conversational data, typography-based and synonym-based augmentation was performed on the data. Data was then split and 2506 conversations were included in train set and 397 conversations were included in test set. Multiple questions and multiple answers were included in each conversation. Questions were input sentences and answers were target sentences.

Sequence-2-sequence model which was not combined with attention mechanism gave a BLUE score of 43.61, whereas, the sequence-2-sequence model which was combined with attention mechanism and used reverse sentences gave a BLUE score of 44.68.

* + 1. Critical analysis of the research item

They didn’t use rule-based methods, pattern matching, and Natural Language Processing approach, instead they used data driven modal that depends only upon data, which is a good strategy as Indonesian language is not resource rich language like English. On the down side, approach used here to build a chatbot requires data in the form of real conversations that has already taken place, and it is not a good data gathering strategy.

This chatbot can understand and respond to only those queries that are similar to any of the queries that already exist in training data (conversation history). A new query which is not same or similar to any query in existing data, cannot be answered. If response to a particular query that exists in conversation history changes, then there is no way to accommodate that change and chatbot will give an outdated response to the user which is no more a valid response. Moreover, a BLUE score of 44.68% is quite low for a conversational chatbot whose goal is to achieve effectivity and user satisfaction.

* + 1. Relationship to the proposed research work

Our project aims to develop a chatbot that can effectively handle students’ queries. Aim of the chatbot developed here is also same as that of our chatbot’s. This research work is relevant to our project also because we are developing a chatbot in a resource-poor language (Roman Urdu) and chatbot developed here is also in a resource-poor language (Indonesian), so research work where a chatbot is built with the same goal as ours, and same kind of language as ours (resource-poor language) is quite relevant to our project.

## Research Item # 2

**“Can Chatbots Help Reduce the Workload of Administrative Officers? -Implementing and Deploying FAQ Chatbot Service in a University”**

* + 1. Summary of the research item

This research work contains the evidences and analyzed how introducing a chatbot in an international college in Korea improved the administrative work patterns and decreased the staff’s workload, and whether students’ queries can actually be resolved by chatbots without human intervention or not. Previously, screen-based automatic response systems (ARS) showed higher satisfaction index (41.1%) than that of voice-based ARS (26%). So, researchers experimented with two offices, a chatbot was introduced in one office and the other office had no chatbot. Then they analyzed how introducing a chatbot improved the administrative work patterns and decreased the staff’s workload.

They used KakaoTalk’s (most popular chat application in Korea) API and deployed the chatbot to KakaoTalk. They used FAQ as data in order to simplify modelling of turn taking. Chatbot incorporated both contextual and menu-based chat types. User could either ask a direct question from the chatbot or traverse the items in the menu to get required information. To understand and remember the context of the conversation DialogFlow was used.

User could either search for required information from the menu or talk to chatbot directly by writing a query. Items in the menu contained academic notices or announcements, employment notices, faculty directories, and academic regulations including graduation requirements. Whereas, Chatbot could directly be asked about majors, graduation requirements, notices, and events inside the campus.

They conducted a pre chatbot experiment for seven days in two offices: office A and office B, and monitored the employees’ workload and number of email queries they received in these seven days. Then they deployed a chatbot in office A and gave its access to students of department A, whereas, staff and students of department B had no access to use this chatbot. They conducted post-chatbot experiment for six days on these two offices. The objective of this experiment was to analyze whether the workload of the employees, and number of email queries by students have been reduced after the introduction of the chatbot or not.

They measured staff’s perceived workloads using NASA-TLX (National Aeronautics and Space Administration Task Load Index) questionnaire, compared number of email inquiries before and after the introduction of chatbot, and the data of chatbot usage (the number of users and the usage frequency of each feature). After the introduction of chatbot, the number of email inquiries received in office A were 27.19% less than the number of queries they received before the introduction of chatbot. Whereas, the number of email inquiries office B received were 71.43% more than the number of queries they received before the introduction of chatbot.

When they analyzed the difference between NASA-TLX scores of workers, and number of email queries received in both of the offices before and after the introduction of chatbot in office A, they observed that there was a significant reduction of email queries sent by students in office A and NASA-TLX score of office A workers significantly improved. There was no improvement in office B. It shows that introduction of chatbot improved the administrative work patterns and decreased the staff’s workload and made it easier for the students to get the required information from the chatbot instantly instead of sending queries to staff and then waiting for the answer.

* + 1. Critical analysis of the research item

Since chatbot was provided through a popular chat application, it established a service environment that lessened users’ unfamiliarity. Although the number of queries through emails were significantly decreased after the introduction of chatbot but staff still received personal queries because chatbot only answered general queries and didn’t handle personal inquiries.

Handling of personal queries could have further decreased staff’s workload and increase students’ satisfaction level. Furthermore, there was a lack of accessibility of chatbot service to students of department A and the experiment was conducted over a very short period of time.

* + 1. Relationship to the proposed research work

Our project aims to develop a chatbot that can effectively handle students’ queries and decrease office workers’ workload. Aim of the chatbot developed here is also same as that of our chatbot’s. This research paper clearly shows that a chatbot in a university clearly automates the question answering process, increases students’ satisfaction, and decreases staff’s workload. We also learned from this research work that not handling students’ personal queries might decrease the chances of chatbot achieving its desired goals (student’s satisfaction and decreased workload for staff). So, we plan to handle students’ personal queries (such as aggregate calculation etc.).

## Research Item # 3

**“Automatic Keywords Extraction Based on Co-Occurrence and Semantic Relationships Between Words”**

* + 1. Summary of the research item

Automatic keywords extraction is a method that extracts words or phrases from a document which can express the main idea of the document. In this paper, researchers propose an unsupervised keyword extraction framework for individual documents.

They removed stop words, and performed length filtering (to improve quality and reduce number of candidate keywords) and regular matching using different methods in order to select candidate keywords. Semantic relationships (WordNet, Word Embedding, Normalized Google Distance), word co-occurrence, and combined methods in which semantic relationships and co-occurrence methods are combined in three different ways, were used to score words.

Methods for extracting keywords are performed in 3 steps:

* Selecting candidate Keywords
* Scoring the candidate keywords
* Selecting keywords based on the results

The main focus of the research paper is on second step. The method they used for keywords extraction are TF-IDF, TextRank, SingleRank, ExpandRank, RAKE, Semantic Relationships based Methods (WE, WordNet, NGD), and Combination Methods. The first combination method includes FCWN (first combination with WordNet), FCWE (first combination with word embedding), and FCNGD (first combination with NGD). The second combination method includes SCWN (second combination with WordNet), SCWE (second combination with word embedding), and SCNGD (second combination with NGD). The third combination methods include TCWN (third combination with WordNet), and TCWE.

Researchers used two datasets: First dataset is Hulth 2003(Inspec) containing 2000 abstracts with two methods of manually assigning keywords: controlled and uncontrolled. For model training, data is divided into 1000 for training and 500 for testing. Second dataset is DUC01 containing 308 new articles with manually assigned keywords. In INSPEC, 15 candidate keywords were assigned as keywords out of 6921 words or phrases, while in DUC01, 3080 words or phrases were extracted with dumping factor of 0.85 and window size of 6 and the top 10 candidate words were chosen as keywords.

Researches compared all the keywords extraction methods they used, by their Precision score, F1 score, and Recall score on two datasets named Inspec and DUC01. Looking at the experiment results, it was observed that methods which combined semantic relationships and word co-occurrence for keywords extraction, gave better results than the methods which only used either semantic relationship or word co-occurrence. It was also observed that keywords were extracted in a better way on individual documents when the method of using co-occurrence between words was applied instead of semantic relationship method.

When keyword extraction methods are compared, the third combination method TCNGD (third combination with NGD) performs best and gives 0.390 precision, 0.705 recall, and 0.502 F1 score on Inspec dataset, and 0.297 precision, 0.368 recall, and 0.329 F1 score on DUC01 dataset.

Below is the comparison of different keywords extraction methods.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Methods | Inspec | | | DUC01 | | |
| Precision | Recall | F1 | Precision | Recall | F1 |
| TextRank | 0.312 | 0.431 | 0.362 | 0.236 | 0.289 | 0.260 |
| SingleRank | 0.328 | 0.593 | 0.422 | 0.247 | 0.303 | 0.272 |
| ExpandRank | 0.383 | 0.692 | 0.493 | 0.288 | 0.354 | 0.317 |
| RAKE | 0.337 | 0.415 | 0.372 | 0.253 | 0.314 | 0.280 |
| CO | 0.374 | 0.677 | 0.482 | 0.268 | 0.332 | 0.296 |
| WordNet | 0.368 | 0.664 | 0.473 | 0.241 | 0.299 | 0.267 |
| WE | 0.373 | 0.674 | 0.481 | 0.248 | 0.307 | 0.275 |
| NGD | 0.375 | 0.677 | 0.482 | 0.253 | 0.313 | 0.279 |
| TCNGD | **0.390** | **0.705** | **0.502** | **0.297** | **0.368** | **0.329** |

Analyzing all these methods, it was found that the combination of semantic relations and co-occurrence improved keywords extraction.

* + 1. Critical analysis of the research item

The precision, recall, and F1 index of combined methods improved keywords extraction a lot as compared to the previous techniques. TCNFGD improves precision by 0.029, recall by 0.036, and F1 score by 0.037.

There are some better approaches that might have been used to get better combination results: differ GAN BERT which gives good results of intent classification using CLINC150 dataset. Keywords extraction using Extended TF gives better results than common TF. The improvement of extended TF than common TF is based on results by considering parts of speech, area of word frequency, word’s morphology, syntactical functions and location. Further the SVM model can be used on the results of combination methods for optimization of the results.

* + 1. Relationship to the proposed research work

We, in our project, have to find the intent of every query the user posts, in order to be able to find out what user actually wants the chatbot to do. Keywords extraction is finding out keywords from the given text and intent extraction is finding out the intent of the user from given text(query). Intent finding also needs keyword extraction from the given query. We have to find intents from queries of our training data. This research paper compares the performance of different keywords extraction methods and concludes that TCNGD methods is most suitable for keywords extraction. We can use this method or combination of this method with some other methods to find intents from user queries for our chatbot training.

## Research Item # 4

**“Developing FB Chatbot Based on Deep Learning Using RASA Framework for University Enquiries”**

* + 1. Summary of the research item

“This Chatbot’s functionality is helping students in several things such as curriculum information, admission for new students, schedule info for any lecture courses, students grade information, weather forecast information, and some adding features for Muslim worships schedule. It interacts with the users in Indonesian language which is a resource poor language like Roman Urdu. Here the chatbot only focuses on case studies at campus of the Magister Informatics FTI University of Islamic Indonesia. They used RASA framework for unifying and pipelining the Deep Learning model.

This Chatbot is developed by Deep Learning models. Deep Learning is based on RNN which has some specific memory saving schemes (specially LSTM which is integrated with RASA framework). RASA works on two main procedures namely RASA NLU and RASA Core. RASA doesn’t use Traditional approach which uses hard-coded rule-based templates and rules that process for responses which is too rigid by rules for users to make inquiries. This chatbot doesn’t need pattern matching. It is not using cloud provider service-based, so the customization is easier, scalable and no limited requests are there for user interaction like most of the other chatbots have.

Architecture of the RASA framework is such that, at the first stage, the message is received and forwarded to the interpreter, namely Rasa NLU to extract intents, entities, and other structured information. Then tracker is tracking, detecting, and maintaining the status of the conversation’s context through the message notifications it has received. Then policy manager receives context status from the tracker and chooses which action will be taken next. These actions are recorded by the tracker. Then actions are executed by sending a message to the user. If the action that has been executed is ignored by the user at a certain time, the process returns to the third step which is tracking. Response can be generated using API calls and database queries.

Intent classifiers, and Entity extractors exist in RASA NLU module. This chatbot used Supervised Embeddings as Intent Classifier. Entity Extractor used for this chatbot is CRF entity extraction. Both memoization Policy and neural network Policy were used by dialogue management module (also called RASA Core). The Memoization Policy just memorizes the conversations in training data. It predicts the next action with confidence 1 if this exact conversation exists in the training data, otherwise it predicts none with confidence 0. F1 score for intent classifier (supervised embeddings) was 0.98 and that of entity extractor (CRF) and dialogue policy maker was 0.92 and 0.95 respectively.”

* + 1. Critical analysis of the research item

The chatbot has been able to provide satisfactory performance in terms of the intent class, entity class and also the appropriate reply from the dialogue policy component was generated but this research is still in first stage of development and has only focused on case studies at campus of the Magister Informatics FTI, University of Islamic Indonesia. Too few intents were handled and accordingly, too few static responses were included. Response generation used only API calls, and static utterances. This chatbot should have been developed by extending the dialogue data and incorporating more features (intents) for helping the automation of tasks in the universities. The chatbot used RASA framework which is extremely powerful and flexible, and works with minimal data. RASA also allows to create a database to store the data required for response generation.

* + 1. Relationship to the proposed research work

This research paper is relevant to our project because it is created to resolve the queries of university students and uses a similar low resource language (Indonesian language) to Roman Urdu. Moreover, it is built using RASA framework and we are also using RASA framework for building our chatbot Rehnuma.

## Research Item # 5

**“Intelligent Assistants in Higher-Education Environments: The FIT-EBot, a Chatbot for Administrative and Learning Support”**

* + 1. Summary of the research item

This research work focused on building a chatbot called FIT-EBot whose purpose was to provide learning and administrative support at the Faculty of Information Technology of Ho Chi Minh City University of Science (FIT-HCMUS), Vietnam. Students can ask questions about services provided by the education system and FIT-EBot replies to these questions on behalf of academic staff. They used multiple artificial intelligence techniques like NER (named entity recognition) and text classification etc. to get better results.

In the approach they used to build the chatbot, they used retrieval-based model as response generation component like most of the other chatbots, to bring more flexibility in their services. The FIT-EBot was implemented using the DialogFlow framework. DialogFlow analyses input messages and generates appropriate responses using different AI techniques. Therefore, in order to recognize user’s intents and context information, and to be able to have intelligent conversations with the users, available messages of the FIT-HCMUS were enough for FIT-EBot to be trained on. FIT-EBot was integrated with Facebook Messenger and therefor, provided a better interaction channel to the users.

For identifying user intents, they used text classification techniques. They classified users’ text messages into different predefined categories (topics). Each topic represents a user intent. In order to train their modal, they identified 13 topics manually. Results of a survey helped them identify these topics (user intents). Whenever, a user asked a new question/query, the model automatically classified this new query into a suitable predefined category (intent).

After the intent recognition task in done, the next goal of FIT-EBot was to extract

context information from the query. NER (Named Entity Recognition) was used for this purpose. An annotated training data, with labeled word blocks was required for the application of this model. This trained model determined the labels of word blocks that were in the new user messages. Appropriate solutions were identified and sent as a response to the user for each identified user intent. For the intents that don’t require information, fixed and unchanged responses were generated and for the intents that require information, database was explored to generate appropriate responses.

The total number of messages for training and evaluating a classifier of 13 topics (user intent classes) were 1560, while the total number of instances serving a NER system of 3 entities (context information) were 870. For each experiment, the approach applied the 10-fold cross-validation method for the evaluation. They used F1 score for evaluating the classification and context extraction performance. The average results for both the user intent identification and the context information extraction were quite promising. The user intent identification scored F1 score of 82.33 and context information extraction scored F1 score of 97.3.

* + 1. Critical analysis of the research item

FIT-EBot shows good performance in terms of intent identification and context information extraction. It uses a retrieval-based model which can perform queries from API of the FIT-HCMUS database system and renew its knowledge by analyzing resources available from the FIT-HCMUS portals and user messages to provide a more flexible and interactive conversation for users. Despite the fact that they were able to produce good F1 scores, they only handled 13 topics (user intents) which is quite low for chatbot to achieve user satisfaction. Moreover, at the time of training, they manually classified text messages into topics (intents) and manually annotated context information in the text messages which is not feasible for a large training corpus. This classification of text messages at the time of training should have been automated by using some machine learning methods and this is one of the goals of our project.

* + 1. Relationship to the proposed research work

This research paper is relevant to our project because this chatbot is created for administrative and learning support for students just like our chatbot. Intent classification/identification and context information extraction (entity extraction) that they performed, are also required for the development of our chatbot. This paper presents quite good architectural design for a chatbot which will be of great help for the development of our chatbot.

## Comparison of Research Papers

Below in the table is the comparison of all the research papers. We compared evaluation matrices, models, methodology, limitations, and scores we observed in each paper.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Paper no | Evaluation Matrix | Model | Methodology | Limitations | Score |
| 1 | BLUE Score | Seq2Seq | Seq2Seq along with attention mechanism and reversed sentences | Handles only few static queries | 44.68 |
| 2 | NASA-TLX, no of email queries | Simple FAQ built using DialogFlow | Pre chatbot and post chatbot analysis of workload of staff | Handled only general queries. Personal queries are still asked via email. | 27.19% decrease in email queries |
| 3 | Precision, Recall, F1 | TCNGD | combination of co-occurrence and semantic relations methods |  | 0.390,0.705,502 |
| 4 | F1 | RASA | Supervised Embeddings as Intent Classifier and CRF as entity extractor | Too few intents and responses handled | 0.95 |
| 5 | F1 | Named Entity Recognition (NER) system, text classification | Intent identification, context extraction and response generation | Manual classification of text, handling only 13 topics | 82.33 for intent recognition.  97.3 for context extraction. |

# Proposed Approach

In this section, we will be discussing our approach to build our chatbot. As we are building a kind of chatbot that has never been built before in Roman Urdu and previously almost none to very little work has been done in this area, we had to devise a completely new approach on our own. We are using RASA framework for the development of our chatbot. **Usage of RASA for Roman Urdu and Hyper tuning of RASA for embeddings in Roman Urdu has never been done before.**

## Architecture of proposed solution

Figure 1 shows the architecture of our solution. A user query is classified into one of many user intents, and useful information and entities are extracted from it. To perform this task, we used **WhitespaceTokenizer, Regex Featurizer, LexicalSyntactic Featurizer, CountVectors Featurizer, DIET Classifier, and EntitySynonymMapper**. In our research, we tried and analyzed different combinations of tokenizers, featurizers, intent classifiers, and entity extractors like: Spacy tokenizer, Jieba tokenizer, ConverT tokenizer, LanguageModelTokenizer, Mitie tokenizer, ConverT Featurizer, Lectical Syntactic featurizer, Spacy featurizer, LanguageModel featurizer, Mitie intent classifier and entity extractor, Sklearn intent classifier and entity extractor, and keyword intent classifier, but none of the combinations was as good as the one discussed above. So we used that combination to achieve the desired goal.

After the intent of the query is classified and entities are extracted, the response selector then predicts the appropriate response. Depending upon the query, it predicts either an appropriate static response from many responses (Retrieval Action), or an API call, or a custom response generation (Custom Action) which uses entities from the query.

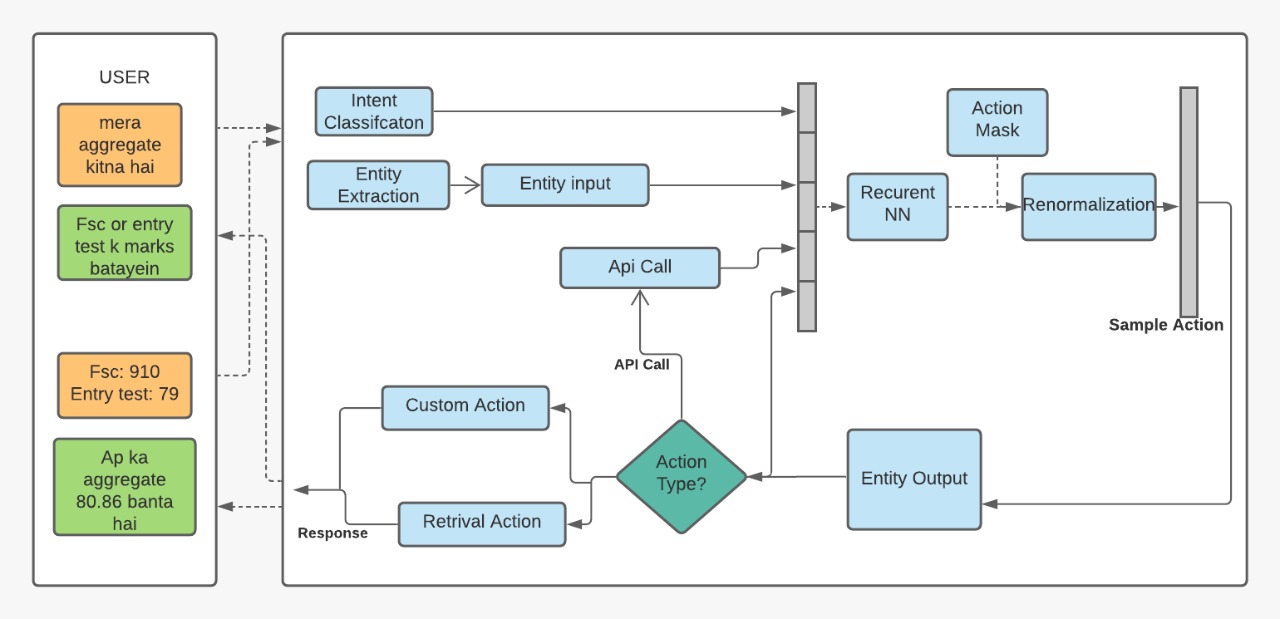


Figure 1. Architecture of Chatbot

As shown in Figure 2, The tracker keeps the state of the conversation. Policies take this state from the tracker, and predictions from the response selector and then decides which action to be taken next.

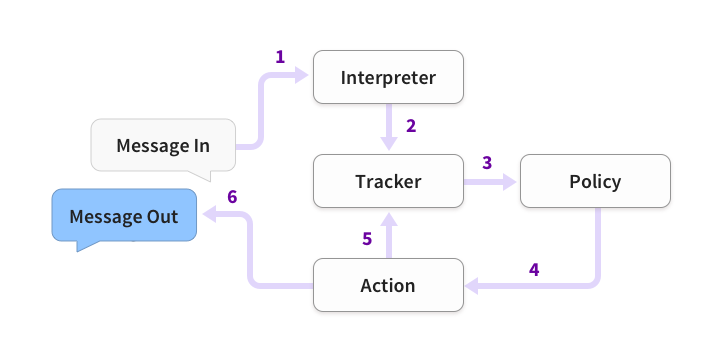


Figure 2. Data flow of Chatbot

## Data Collection

To train our model, we need training data which will contain the real queries in Roman Urdu posted by real users. Previously no dataset is available in Roman Urdu that we can use for training. So, for training purposes, we have to collect data from scratch. We planned to collect data through Facebook scrapping as well as to manually collect data (general queries asked in Roman Urdu about FAST, and queries asked in Roman Urdu about FAST admission process) through forms from batch 20 students. Both of these data collection methods come with challenges which will be discussed later in this report. Our collected data will be the first of its kind and will be helpful for future research.

## Data Cleaning

Facebook posts and comments are in raw form and contain a lot of irrelevant data. This data needs to be cleaned to acquire only useful queries. When data collected from Facebook, it contained most of the queries in English which need to be translated in Roman Urdu. To clean data, we have to handle spell variations and offensive language detection etc. Most of the techniques proposed for the normalization of data, and detection of offensive language are extremely domain specific and they tend to produce accuracy under strict constraints. We have to evaluate different machine learning modals, classifiers, and feature extraction methods, and then we will use the suitable techniques to solve these problems. We have to write some scripts to clean our collected data and plan to clean data using some ML techniques instead of manually cleaning data.

## Intent Generation

Our cleaned data will consist of thousands of sentences (user queries). Each query has an intent (user intent is what user wants). Manually assigning intents to each query is neither feasible nor is it a good approach as we want to automate our work.

We have done a lot of research on intent generation (or keywords extraction) and, are working on automatically extracting intents from user queries. These queries along with their intents will be used to train RASA so that when a user asks a new question, RASA is able to correctly identify its intent in order to be able to generate an appropriate response.

## Development of Web-based User Interface

Web based user interface will be developed and our modal will be integrated with it in order to give users an interactive and pleasant User interface.

# Implementation of proposed solution

## Data Collection (Scrapping and Manual Collection)

The availability of the data is the baseline for the creation of any chatbot. As dataset related to admission inquiries in universities (especially FAST) is not already available, we used the following methods for data collection:

* + 1. Scraping Facebook Pages/Groups

Facebook groups/pages are the platforms where most of the students post their queries regarding Fast and other universities. So, we decided to take 35 groups/pages related to admission inquiries. These groups/pages are of FAST and different other universities of Pakistan. Pages of other universities were chosen because mostly students’ queries are of same kind across different universities. Purpose behind this was to collect as much data as possible for training.

**Methodology**

The scrapper we used for data collection is brutalsavage from github.

Github Link: <https://github.com/brutalsavage/facebook-post-scraper>

The data related to our domain is quite specific. We needed to collect only relevant data. For this purpose, we specified almost 230 terms related to admission inquiry e.g., semester fee, aggregate, admission, CS, EE, criteria etc. We scrapped only the posts that contained these terms. The raw dataset was around 17000 posts, and 30000 comments. After cleaning, it was reduced to 3000 posts and 7000 comments.

**Challenges**

**Facebook New layout**

For data collection, the major step is to write a script to scrape data from pages or groups. Usually, Facebook scrappers are easily available on github but due to Facebook’s recently changed layout, existing scrapers were not useful for scrapping data anymore. We wrote a new script and this took a lot of time at the initial phase of data collection.

**Availability of Data**

The admission process usually happens during the months of July-September every year. So, the availability of related information was quite low. As we were specifically looking for data in Roman Urdu. This further complicated thing.

**Scrapper limitations**

We used “brutalSavage” scrapper for posts and comments scrapping. It took a lot of time for scraping. It usually takes an hour to scrap 200 posts from Facebook. Furthermore, we faced a lot of Facebook page crashes during scrapping. We had to scrap data from scratch all over again.  Facebook pages usually crash if we try to scrap more than 2000 posts from a particular group at once.

* + 1. ***Manual collection of data***

We created a google form and gave that form to the batch 20 students. We got more than 500 questions written in Roman Urdu regarding admissions at FAST from those students through that google form. This data was mostly cleaned already as the student wrote those questions that they had in their mind at the time of admission. So, it was the data coming from the real future users of the chatbot.

## Data Cleaning

Most of the data collected from Facebook pages was in English. Queries were either written in English or written in Roman Urdu mixed with English. We cleaned the data by using python code and extracted queries written in Roman Urdu only. We also translated English queries in Roman Urdu so that those queries could also be used for the training of our chatbot.

The output of scrapping was in json format like:

{"Post": hello kesy ho"", "Comments": {"Uzair Sheikh": { mn thk }, "Aroush Elahi": {ap ka kiya haal hai}}}

First of all, we separated the posts and the comments by using tokenization based on comma (,). Comments and posts were stored in separate files. Then we removed characters Like “/,#@\_)({} from posts and comments. Then we removed duplicate and irrelevant posts and comments. We checked relevance using substring matching. We identified more than 200 relevant terms like “semester fee”, “admissions “, “postgraduate”, “post graduate”, “post grad”, “open”, “khul”, “admission”, “admissions”. If a query contains more than 2 substrings. It was considered to be relevant.

After that Roman Urdu queries were separated from English queries and we used English dictionary to achieve this goal. If a line contained all English words, it was considered to be an English query otherwise it was considered to be a Roman Urdu query. We also had to translate English queries to Roman Urdu queries and for that we used <https://www.ijunoon.com/transliteration>. But the accuracy of ijunoon was almost 60% and we manually had to translate some queries from English to Roman Urdu.

## RASA 1.0 Installation and Migration to RASA 2.0

We chose RASA framework for the development of our chatbot. Initially we trained RASA on small number of queries falling in 30 different categories (intents) and performance of RASA was quite satisfactory when it came to classifying new queries into existing intent categories. We were Using RASA 1.0 but then came RASA 2.0 and we had to migrate to RASA 2.0 as RASA 1.0 stopped working properly after the release of RASA 2.0. There wasn’t enough documentation available regarding RASA 2.0 on the internet. On top of that RASA 2.0 is not backward compatible so we had to start from the beginning as RASA 2.0 was completely different from RASA 1.0 in terms of coding and implementation. We had to retrain RASA and previous training of RASA on 120 queries went in vain. RASA 2.0 was trained differently and we had to do all the previous work again and in a different manner this time.

## Intent Generation and Classification

We have trained RASA on **1300-1500 queries falling in** **50-60 different categories** (intents) and performance of RASA is quite satisfactory when it comes to classifying new queries into existing intent categories. For intent generation, we tried different algorithms which include the algorithms discussed in research item number three as well. The best accuracy we achieved by any algorithm was around 40%. As the best accuracy we were able to achieve, was only around 40% we had to manually assign rest of the queries to appropriate intents. RASA NLU is trained in such a way that it is able to identify the intent of the new unseen query with above **90% confidence** most of the times.

* + 1. Handling offensive language and spell variations

Our modal is also able to ***detect and handle offensive language, and handle spell variations.*** Any offensive language sent towards the bot is being detected by the bot and it is able to handle variations in the spelling of same words e.g. the following sentences:

*“FAST ka daaakhlllaa kabb kholay ga”*

and

*“Fast ka dakhla kab khuly ga”*

Will be considered the same even though the spelling of most of the words are different in both the sentences. To achieve this functionality, we used queries that contained different spelling variations of same words in our training data set. Although it’s not possible to train our model on every spelling variation of a word, but use of enough variations of word and use of appropriate tokenizers, featurizers, and intent classifiers helped achieving the desired goal.

* + 1. Handling out of the scope queries

We are handling around 50 to 60 intents (categories of queries) so if there is any new unseen query which does not belong to any existing intent, RASA will identify many intents for this new query and confidence of each intent will be too low. If the confidence of each intent identified by RASA is below a certain limit, RASA will eventually identify this query as out of the scope query. This is how out of the scope queries are also being handled without the chatbot being clueless about what to do if an out-of-scope query comes across.

## Follow up Questions and Response Generation

Once the modal has correctly identified/classified the intent of the query and extracted the entities from the query, the next step is to generate an appropriate response to the query. Response is being generated depending upon the query. Those queries who have static responses are given static responses and those queries whose responses are dynamic are given dynamic responses depending upon the entities extracted from the query. In case of incomplete information in the query, the chatbot can also ask questions from the user in order to get the information required for response generation.

* + 1. Static Responses

Some queries require only the static responses. For example, if a user asks “*FAST k Islamabad campus ki location kya hai.”* The bot will identify the intent of the query as *“location”* and will extract the entity (*Islamabad campus*). Now this query is static in a sense that the response to this query will be same for every user and it does not depend upon the user. The response to this query will be “*FAST Islamabad ki location* *A.K. Brohi Road, H 11/4 H-11, Islamabad* *hai*.” This reply will be same (static) for every user who asks for the location of Islamabad campus.

This functionality is achieved using DIET classifier and entity extractor which gives the best performance on Roman Urdu and correctly classifies the user query into an appropriate intent. After the intent of the query is classified and entities are extracted, the response selector then predicts the appropriate response. It predicts an appropriate static response from many responses (Retrieval Action).

* + 1. Dynamic Responses

Some queries require dynamic responses depending upon the entities sent through the queries. For example, if a user asks *“mery FSC mein 90% marks hein or entry test mein 70 marks aaye hein. Mera aggregate kitna bany ga*?” our chatbot will identify the intent of this query as “*aggregate\_calculation*” and will extract the entities (*FSC marks and Entry test marks*) from this query and will use these extracted entities to calculate the aggregate of the user. When the aggregate is calculated, the bot will reply to the user like this “*aap ka aggregate 80 ban raha hai.”* This reply will be different for different users depending upon their masks (entities extracted).

This functionality is achieved using DIET classifier and entity extractor which gives the best performance on Roman Urdu and correctly classifies the user query into an appropriate intent. After the intent of the query is classified and entities are extracted, the response selector then predicts the appropriate response. It predicts an appropriate dynamic response from many responses depending upon/ using the entities extracted (Custom Action).

* + 1. Asking Follow up Questions

Some queries do not have enough information for the chatbot to decide the appropriate response. To handle this kind of queries, chatbot needs to ask for further information. So, in case of incomplete information provided in the query, chatbot asks the follow up questions to get the required information. For example, if the user asks *“mera aggregate kitna banta hai”* then chatbot will ask for his or her FSc marks and entry test marks like this: *“apny FSC k marks batayen”* and then *“apny entry test k marks batayen”*. Once the user has entered both FSc marks and entry test marks, chatbot will call the custom actions and that custom function will calculate the aggregate using FSc marks and entry test marks and will reply to the user. To achieve this functionality, we had to write user stories in order to tell the chatbot how and in what sequence a user can talk to the chatbot. The tracker component of the chatbot keeps the state of the conversation.

* + 1. Remembering the Context of the Conversation

Chatbot also remembers the context and state of the conversation. For example, in the previous example, when chatbot asks for FSc marks and entry test marks, user enters his or her marks but doesn’t ask for the aggregate calculation again. Chatbot is able to remember that it is supposed to calculate the aggregate after the marks are entered by the user. This functionality is achieved by writing user stories in order to tell the chatbot how and in what sequence a user can talk to the chatbot. The tracker component of the chatbot keeps the state of the conversation.

## Development of Web based User Interface

Lastly, we needed to create a User Interface for our chatbot. For this purpose, we needed to connect our model with popup button. The main challenge in this phase was the connection of RASA with User Interface. For the connection, we needed webchat API to connect model with the popup button through socket.io. Through socket.io, all the real time conversation is possible between model and user interface. Web chat also provides some features for designing and alignment. All the user message events are handled by *user\_uttered* event and all bot messages handled through *bot\_uttered* event.

If we discuss the overall working of our solution in sequence, first of all, raw user query needed to be tokenized. We tried different tokenizers like Spacy tokenizer, Jieba tokenizer, Convert Tokenizer, language modal tokenizer, etc.  these tokenizers give good performance on resourceful languages like English but they were not useful for Roman Urdu. So, we used whiteSpaceTokenization. Then this tokenized query needed to be encoded into a feature vector for modal training, we used three different featurization techniques. LexicalSyntactic featurization which created lexical features and supervised word embeddings, Regex featurization using regular expressions, and countVector Featurization for bag of words representation of the user query.

For Intent classification and entity extraction, again there were many classifiers and entity extractors but they were not giving a good performance on Roman Urdu. So, we used DIET classification, and synonym mapping. DIET classification used CRF tagging. Synonym mapper mapped synonymous entities to the same entity. As shown in Figure 1, once the user query is correctly classified and entities are extracted, the Response Selector then predicts the appropriate response.  Depending upon the query, it predicts either an appropriate static response from many static responses, or an API call, or a custom response (Custom function calls) which uses entities from the query.

As shown in Figure 2, message is first interpreted to machine language. The tracker here keeps the state of the conversation. The Policies take two things: conversation state from the tracker, and predictions from the response selector. Then the policies decide which action to be taken next.

# Results and Discussion

If we talk about the results, we have trained our chatbot on about 1500 queries in such a way that it is able to identify and classify queries falling in 50 to 60 different categories (intents). Whenever, a new unseen query arrives, the modal is able to classify that query in its respective intent. When it classifies the query in a particular category, it also gives the “confidence” of this classification. Till now, the lowest confidence of a query classification we have seen is 72% and majority of the queries are classified with confidence above 90%.

Secondly our modal is able to handle spell variations, out of the scope questions and detect offensive language. As far as response generation is concerned, our chatbot is also able to give both static responses as well as dynamic responses in case of a dynamic query. It asks follow up questions in case of incomplete information provided in a query and also remembers the context of the conversation.

Although, when we talk to the chatbot and enter a query, RASA NLU module shows; that with how much confidence this particular query was classified into a particular intent. But this confidence is always for that particular query. RASA doesn’t provide any performance measure for the whole model. So, we had to test the model ourselves.

We took 110 real user queries which were not present in the training data and these queries were falling in 11 different intents/categories. It is not feasible to manually test the modal on queries falling in all 50 to 60 intents because we will have to manually test the modal on more than 500 queries and confusion matrix for 50 to 60 queries cannot be presented here. So, we tested our chatbot on these 110 new queries and made a confusion matrix. Below is the confusion matrix for 11 most searched and common intents.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **INTENTS** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| **1) Fee** | **7** |  |  |  |  |  |  |  |  |  |  |
| **2) Admission Requirements** |  | **8** |  |  |  |  |  |  |  |  |  |
| **3) List Date** |  |  | **9** | **1** | **1** |  |  |  |  |  |  |
| **4) Closing Merit** |  |  | **1** | **9** |  |  |  |  |  |  |  |
| **5) Admission Open** |  |  |  |  | **8** |  |  | **1** |  |  |  |
| **6) Test** |  |  |  |  |  | **6** |  | **1** |  |  |  |
| **7) Scholarship** | **1** |  |  |  |  |  | **8** |  |  |  |  |
| **8) Programs** |  |  |  |  | **1** | **1** |  | **8** |  |  |  |
| **9) Hostel** |  |  |  |  |  |  |  |  | **10** |  |  |
| **10) Transport** |  |  |  |  |  |  |  |  |  | **9** |  |
| **11) Out of Scope** | **2** | **2** |  |  |  | **3** | **2** |  |  | **1** | **10** |

If we look at the *FEE* intent, out of 10 fee related queries, model classified 7 of them correctly, 1 of them was classified as scholarship query and 2 of them were classified as out of scope queries. Similarly, if we look at the *Admission Requirement* intent, 8 of them were classified correctly but 2 of them were classified as out of scope queries. All 10 out of scope queries were correctly classified as out of scope queries. Same are the results of *Hostel* intent.

In the table below are the Accuracy, Precision, Recall, and F1 scores of these intents.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **INTENTS** | **Accuracy** | **Precision** | **Recall** | **F1-Scores** |
| **Fee** | **0.97** | **1** | **0.7** | **0.82** |
| **Admission Req.** | **0.98** | **1** | **0.8** | **0.89** |
| **List Date** | **0.97** | **0.82** | **0.9** | **0.86** |
| **Closing Merit** | **0.98** | **0.9** | **0.9** | **0.9** |
| **Admission Open** | **0.97** | **0.89** | **0.8** | **0.84** |
| **Test Pattern** | **0.95** | **0.86** | **0.6** | **0.71** |
| **Scholarship** | **0.97** | **0.89** | **0.8** | **0.84** |
| **Programs** | **0.96** | **0.8** | **0.8** | **0.8** |
| **Hostel** | **1** | **1** | **1** | **1** |
| **Transport** | **0.98** | **1** | **0.8** | **0.89** |
| **Out of Scope** | **0.9** | **0.48** | **1** | **0.65** |

We achieved an average Accuracy of 94%, Precision of 88%, Recall of 83%, and average F1 score was 84%.

So overall, the performance of the chatbot is very good.

# Conclusion

Our chatbot REHNUMA is now able to handle 50 to 60 different type of questions and its responses are not only static but also dynamic whenever dynamic responses are required. It is very difficult to predict everything a user can ask and the way the user will ask it. However, we were able to achieve our Goals and Objectives (discussed in section 2.2) and our chatbot can handle static and dynamic queries falling into more than 50 different categories. Also, existing NLP is not yet developed enough to be able to handle conversation in resource-poor languages like Roman Urdu. So, we researched and tried many different techniques, and based on our research and results, we found out that the techniques discussed in section 4.1 and section 5 give the best results for Roman Urdu.

# References

* Duong, Long. (2017). Natural language processing for resource-poor languages. Melbourne: Melbourne Research Publications.
* Yurio Windiatmoko, Ahmad Fathan Hidayatullah, Ridho Rahmadi. (2020). Developing FB Chatbot Based on Deep Learning Using RASA Framework for University Enquiries. Yogyakarta, Indonesia: arXiv.org.
* Xiangke Mao, Shaobin Huang, Rongsheng Li, Linshan Shen. (2020). Automatic Keywords Extraction Based on Co-Occurrence and Semantic Relationships Between Words. IEEE Access: IEEE
* Yogi Wisesa Chandraa, Suyanto Suyantoa,∗. (2019). Indonesian Chatbot of University Admission Using a QuestionAnswering System Based on Sequence-to-Sequence Model. West Java, Indonesia. Elsevier B.V
* Lee K., Jo J., Kim J., Kang Y. (2019) Can Chatbots Help Reduce the Workload of Administrative Officers? - Implementing and Deploying FAQ Chatbot Service in a University. In: Stephanidis C. (eds) HCI International 2019 - Posters. HCII 2019. Communications in Computer and Information Science, vol 1032. Springer, Cham. <https://doi.org/10.1007/978-3-030-23522-2_45>
* Ho Thao Hien, Pham-Nguyen Cuong, Le Nguyen Hoai Nam, Ho Le Thi Kim Nhung, Ho Thao Hien, Pham-Nguyen Cuong, Le Nguyen Hoai Nam, Ho Le Thi Kim Nhung, Le Dinh Thang. (2018). Intelligent Assistants in Higher-Education Environments: The FIT-EBot, a Chatbot for Administrative and Learning Support. [Proceedings of the Ninth International Symposium on Information and Communication Technology](https://dl.acm.org/doi/proceedings/10.1145/3287921).