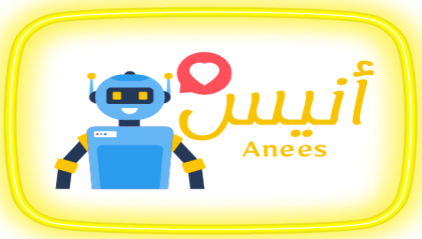
Cairo University
Faculty of Engineering
Department of Computer Engineering

Anees



A Graduation Project Report Submitted

to

Faculty of Engineering, Cairo University

in Partial Fulfillment of the requirements of the degree

of

Bachelor of Science in Computer Engineering.

Presented by

Ahmed Ashraf
Ahmed Sherif
Ahmed Magdy
Abdelrahman Ahmed Fadl

Supervised by

Prof. Mona Farouk

19/7/2022

Abstract

Many companies are starting to enter the industry of chatbots to answer frequently asked questions of their clients on their businesses. These chatbots should be intelligent enough to understand and answer the asked questions. The need for Arabic chatbot is highly increasing for many international companies so these companies have to keep up with this increase and afford another version of their chatbots that can speak Arabic. The number of chatbots that support Arabic is very small and most of them are domain specific that have poor accuracy in talking in general purpose conversations. Anees is an Arabic chatbot mobile application with a wide set of features like Friendly open-domain conversation with users on different topics, Sentimental Analysis to detect user's emotions which used to analyze the user behavior over a period of time, Intent Classification to determine the purpose of the user's message and accordingly Anees responds with the proper action which can be an event scheduling, weather forecast, recommendation about movies or some places or a general response. We used a variety of the Natural Language Processing and Machine Learning techniques for the language understanding, language generation, emotion classification, intent classification, and text processing to provide the necessary functionalities for the final product. The mobile application was developed using Python, SKlearn, PyTorch, Flask, MongoDB, and React Native. And was tested on Web, Android, and iOS environments with different conversations on different topics. The results are promising relative to known Arabic chatbots.

الملخص

فى الوقت الحالى العديد من الشركات تحاول بناء روبوت للدردشة للإجابة عن أسئلة عملائهم عن طريق موقعهم الالكتروني. هذه الروبوتات الخاصة بالدردشة يجب أن تُصمم ببعض من الذكاء لفهم ماذا يحتاج العملاء والإجابة عنه بشكل صحيح لكن معظم هذه الروبوتات الخاصة بالدردشة تكون خاصة بموضوع محدد، وايضا عدد روبوتات الدردشة التي تتحدث باللغة العربية قليل جدا وأيضا ذات دقة ضعيفة ،وعدد العملاء ذوو اللغة العربية يتزايدون فى العديد من الشركات الدولية لذلك يجب على تلك الشركات أن تواكب هذه الزيادات فى أعداد العملاء وتستطيع تقديم إصدارات خاصة من روبوتات الدردشة الخاصة بهم تستطيع التحدث باللغة العربية. أنيس هو روبوت دردشة (مساعد افتراضي) للهواتف الذكية والذي يحتوي على العديد من المزايا يمكنه أن يتحدث مع المستخدم فى العديد من المواضيع المختلفة باللغة العربية الفصحى وايضا باللغة العامية المصرية، يقوم بتحليل مشاعر المستخدم ويقوم بتقديم مقترحات بناء على مشاعر المستخدم على فترة من الزمن، يقوم بتحديد نية المستخدم من كلامه إذا ما كان يريد أن يتحدث فقط أم يريد تنفيذ عمل معين مثل تحديد احوال الطقس في بلد معين فى وقت معين، تسجيل ميعاد فى التقويم، أو اقتراح فيلم او اقتراح مكان معين على حسب رغبة المستخدم، لقد قمنا باستخدام العديد من التقنيات فى تصميم هذا التطبيق مثل المعالجة اللغوية الطبيعية وذكاء الآلة لتنفيذ تقنيات فهم اللغة العربية ولتنفيذ أيضا تقنيات توليد اللغة وتقنيات تحديد مشاعر المتحدث و نيته من كلامه وأيضا تقنيات معالجة النصوص لتقديم الوظائف اللازمة للمنتج الأخير ليستطيع فهم كلام المستخدم والرد عليه بشكل صحيح، تطبيق الهاتف تمت برمجته بواسطة بايثون و سك ليرن و بايتورش و فلاسك و مونجو داتا بيز و ري أكت وتم اختبار هذا التطبيق على المتصفح و هواتف الاندرويد وهواتف الايفون عن طريق العديد من المحادثات التي تمت مع أنيس. والنتائج مبشرة مقارنة بروبوتات الدردشة التي تتحدث باللغة العربية.

ACKNOWLEDGMENT

We would like to show how much we are grateful to our supervisor Prof. Mona Farouk for her great support, guidance, advice and help. We also would like to express our thanks to our professors and TAs for their great support and effort throughout the last 5 years to help us grow and how you encouraged us to face challenges and pass them. Finally, we want to thank our families and friends for their continuous support.

Table of Contents

List of Figures

List of Tables

List of Abbreviation

API	Application programming interface
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
DF	Document Frequency
DL	Deep Learning
GPT	Generative Pre-trained Transformer
IDF	Inverse Document Frequency
LM	Language Modeling
LSTM	Long Short-Term Memory
ML	Machine Learning
MSA	Modern Standard Arabic
NER	Named Entity Recognition
NLG	Natural Language Generation
NLP	Natural Language Processing
NLU	Natural Language Understanding
NN	Neural Network
POS	Part of Speech
RNN	Recurrent Neural Network
SWOT	Strength Weakness Opportunities Threats
TF	Term Frequency

List of Symbols

d	Document
e	Exponential function
D	Corpus
n	Number of terms in document
N	Number of documents in corpus
p	Probability
PPL	Perplexity
t	Term
W^T	Weights of features
x_n	Feature vector of data point n
y_n	Label of data point n

Contacts

Team Members

Name	Email	Phone Number
Ahmed Magdy Ahmed	amagdy.omran7@gmail.com	+2 01288166163
Ahmed Sherif Ahmed	ahmedsherifelminshawy@gmail.com	+2 01120945664
Ahmed Ashraf	ahmed.ashraf.cmp@gmail.com	+2 01060675041
Abdelrahman Ahmed Fadl	abdelrhmanfdl99@gmail.com	+2 01033783948

Supervisor

Name	Email	Number
Prof. Mona Farouk	mona_farouk_ahmed@yahoo.com	+2 01005042029

This page is left intentionally empty

Chapter 1: Introduction

Anees is an Arabic chatbot. Arabic chatbots became very important as many companies had many Arabic clients. Anees can talk to users in general especially in Arabic and also support some friendly tasks such as scheduling a meeting or appointment in the calendar, getting the weather for the user, recommending movies or places to the user based on his/her preference. Also trying to support his/her emotions.

1.1. Motivation and Justification

Chatbots became important for many business companies as they save money and time for them as they can answer some questions of users but Chatbots that speak in Arabic language are not many and they have poor accuracy, and many Arabic people want to communicate with these companies but can have some problems with other languages so making a version of chatbot became essential. Also most of these chatbots speak on specific topics or a specific domain.

1.2. Project Objectives and Problem Definition

Our objective is to build an Arabic chatbot that can speak to users in different topics or an open-domain conversation rather than a specific domain. Our objective is to build a chatbot that you can consider your friend that you can talk with about anything in your mind while the existing chatbots in Arabic are topic-oriented or task-oriented only.

1.3. Project Outcomes

Anees is an Arabic chatbot mobile app that can provide some features:

- Friendly conversation
- Sentimental Analysis
- Tasks as :
    - Weather Forecast
    - Schedule meetings
    - Recommendation of movies or locations

Anees also gather some information from the user behavior and conversations to provide better performance and functionalities.

1.4. Document Organization

The document covers the most important parts and details of our project in order to help the readers understand the project. It is divided into 6 chapters. Chapter 1 (Introduction) includes a short introduction about the project, motivation, project objectives, and the project outcomes.

Chapter 2 (Market feasibility study) includes our targeted customers, competitors, business case, and financial analysis. Chapter 3 (literature survey) includes background on each module, comparative study of previous work, and the final implemented approach.

Chapter 4 (Architecture) includes a block diagram, every module, its functionality and components. Chapter 5 (System Testing and Verification) includes testing setup, testing schedule, testing for each module, and integration Testing. Finally, Chapter 6 includes the challenges and a conclusion about the project and what we learned through the development of Anees. Also, there are some appendixes for the user guide, use case, code documentation, and feasibility study.

Chapter 2: Market Feasibility Study

The global intelligent virtual assistant market size was valued at USD 5.82 billion in 2020. It is expected to expand at a compound annual growth rate (CAGR) of 28.5% from 2021 to 2028. The need for improved efficiency across service-based companies and the integration of Artificial Intelligence (AI) powered virtual assistants among various devices such as tablets, computers, and smartphones, among others, is anticipated to boost the market growth [1].

2.1. Targeted Customers

Our targeted customers are Arabic speaking people who want a private assistant that they can share their ideas and speak freely with. Anees helps the users through its proper responses and tries to analyze their behavior and emotions to recommend them some movies, places, …etc. Anees can help people to say whatever they want without hesitation or fear.

2.2. Market Survey

There are machine learning competitive products in the virtual assistant world. One of the known Arabic virtual assistants are:

- Google Assistant
- Rafiq

2.2.1. Google Assistant

Figure 2.1: google assistant

This is one of Google's products, it supports many arabic dialects.

Pros:

- Provides many services like search engine, calendar, story telling, …
- Supports vocal and text responses.
- Relatively very accurate conversation.

Cons:

- Privacy concerns.
- Need permissions for almost all privileges.

2.2.2. Rafiq

Figure 2.2: Rafiq

Rafiq is a task oriented chatbot that can perform many tasks in many arabic dialects.

Pros:

- Speaks with multiple arabic dialects (Egyptian, Syrian, …etc).
- Has voice support (not only chat).
- Perform a wide variety of tasks.

Cons:

- Does not support natural language understanding.
- Task-oriented. It doesn't support open-domain conversations.

2.3. Business Case and Financial Analysis

2.3.1 Business Case

Our product consists of a mobile app that is available on Google Play for Android and App Store for iOS. Based on our market survey, most of the Arabic Virtual Assistants can be downloaded for free either from App store for iOS or Google Play Store for Android. But some of them have some features that are locked but can be unlocked by subscription. This premium subscription of our competitors is about $1.99 per month which is around 37 EGP per month. So we decided to make the download of the app for free with some premium features such as game recommendation, books and novels recommendation, …etc for $ 1 per month which is around 18 EGP per month.

2.3.2 Financial Analysis

The financial analysis for our project based on the previous business case is presented in Table 2.1 below. Shown in the table are the Capex which are the one-time spending such as the costs of purchasing computers for our employees, which would be in the first year, and then again after two years when we increase our number of employees from four to ten. The table also includes the Opex which are the recurring payments such as the GCP cost each 6 months for the training process, salaries for employees, rent and website hosting costs.

It is expected that the advertising and marketing specifically on social media platforms to target the different generations would help in making the app more popular and used. Most of the well known applications now started by getting known among teengers and social media users. From the statistics on similar chatbots released in the past couple of years, our plan for advertisement on different social media platforms, and our local distribution of the application, we can have a minimum of 500 users by the first year. Suppose that in the first year, the 500 users for the application each paid $2 on the application, then the total subscription profit in 1st year could be $1000. As years go by, more users would subscribe and more profit could be made. In addition, ads profit. On the other hand, the cash out would include the cost of the cloud server hosted on GCP that would be used in training of the models, and the website host to put it online. Also, there are the salaries of the employees and office rent and bills. Lastly, the cost of the advertising on other websites. More details are shown in Table 2.1.

	Year 1	Year 2	Year 3	Year 4	Year 5
Computers	$5,000		$6,500		
Host and GCP	$1,500	$2,000	$2,500	$3,000	$3,500
Salaries	$16,000	$16,000	$40,000	$40,000	$40,000
Rent and Bills	$1500	$1500	$3500	$3500	$3500
Advertising and Marketing	$200	$200	$300	$300	$300
Total Cash Out	$24,200	$19,700	$52,800	$46,800	$47,300
Subscriptions Profit	$1,000	$5,000	$25,000	$60,000	$100,000
Ads Profit	$2,000	$6,000	$20,000	$45,000	$50,000
Total Cash In	$3,000	$11,000	$45,000	$105,000	$150,000
Revenue	-$21,200	-$8,700	-$7,800	$58,200	$102,700

Table 2.1: Financial Analysis of Anees

Chapter 3: Literature Survey

The main topics on which the project is built on: Natural Language Understanding, Natural Language Generation, Emotion Classification and Intent Classification

3.1.Background on Natural Language Understanding

It is the process of making some preprocessing on text and extraction of information. It includes preprocessing, tokenization, part of speech, and stemming.

3.1.1 Preprocessing

In this step we want to

- Remove all punctuations
- Remove diacritics
- Normalize text so similar letters are the same such as: ي ⇐ ى

3.1.2 Tokenization

Tokenization is the process of extracting words(tokens) from text then removing stop words as they have no effect on data such as pronouns e.g:

انا ذاهب الى المدرسة ⇐ (انا , ذاهب , الى , المدرسة)

After removing Stop words the remaining tokens will be (ذاهب , المدرسة)

3.1.3 Name Entity Recognition

Name Entity Recognition (**NER**) is the process of labeling the key words in text, like Person, Place, Organization, ….etc. By extracting the key words, we are getting closer to know the intention of the sentence.
Examples:

- أحمد ⇐ B-Pers
- الزمالك ⇐ Org
- أحمد شريف ⇐ (أحمد : B-Pers), (شريف : I-Pers)

3.1.4 Part of Speech

Part of Speech is the process to classify a token into a noun or verb using data stored. Continue from the last example tokens:
(ذاهب , المدرسة) <= ((اسم) المدرسة , (فعل)ذاهب)

3.1.5 Stemming

Stemming is the process to remove additional letters in words (tokens) so as to get a common word for different shapes for a word. It is done by having a set of all the words in the language and having all the possible prefixes and suffixes in the language.

For each word that gets into the stemming process it is tried to remove each suffix and each prefix and if a word is found in the set of all the words we have so it is the stem of the word[2].
Consider the following examples:

- Set of prefixes → {ال,ي,ت}, Set of suffixes -> {وا,ون,وين}
- The set of words → {لعب,أكل}
- Word needs to be stemmed -> يأكلون
  - It has 'ي' as a prefix so we can remove it and also it has 'ون' so we remove it
  - It is being reduced to 'أكل' which is a word in the set of words so the stem of يأكلون is أكل
- Another word → تاج
  - It has ت as a prefix so we can remove it
  - It is being reduced to اج which is not a word in the set so the word تاج is its own stem

3.1.6 Lemmatization

Lemmatization is to return a word to its root by removing any additions to the word. The paper in [3] has a method to build fast and accurate lemmatization for Arabic language.

3.2. Background on Emotion Classification

Emotion Classification is the process of analyzing text to extract the emotion of the user based on text. There are many techniques to classify the text into emotions such as DL techniques and ML techniques. The used technique is based on TF-IDF and Logistic Regression

3.2.1. TF-IDF

First we make some NLU techniques on text then we get a set for all words in the dataset gathered so each word in the set is a feature in the extracted features. Then for every word we get its TF in each document which represents the weight of each word in the document:

$$TF(t,d) = \frac{\sum_{i=1}^{n} f(t,t^i)}{n+1} \quad t^i \in d,\ |d| = n$$

$$f(t,t^-) = 1\ if\ t = t^-$$

After that, we get DF which is the number of documents that contains the word:

$$DF(t) = \sum_{i=1}^{N} g(t,d^i)\ d^i \in D,\ |D| = N$$

$$g(t,d^i) = 1\ if\ t \in d^i$$

Finally, we calculate IDF:

$$IDF(t) = log\frac{N}{DF(t)+1}$$

then the each feature will be $TF(t,d)$ * $IDF(t)$.

3.2.2 Logistic Regression

Logistic Regression is a supervised learning technique to predict the probability of a binary event. The logit function used is $\theta(a)$:

$$\theta(a) = \frac{1}{1+e^{-a}}$$

Logistic Regression is based on maximizing likelihood (P(Class|Data)) which is also minimizing negative log likelihood.

∇Ein(W) = min(-ln(P(Class|Data))) = min ($ln(1 + e^{-yW^TX})$)

W: Weights of features

But there is no analytical solution for ∇Ein(w)=0 so we use Gradient Descent which is a method for optimization which will converge to the global minimum then the algorithm will be :

1. Iterate on Max iterations
2. Compute Gradient Descent : ∇Ein(W) = $-\frac{1}{N}\left(\sum_{n=1}^{N}\frac{y_n x_n}{1+e^{y_n W^T x_n}}\right)$
3. update weights w(i + 1) = w(i) - $\eta\nabla$Ein(W)

3.3. Background on Intent Classification

3.3.1 Sequential models

Sequential models are models that their input and output are data sequences. They are used in text analysis, time series and many other things. Sequence modeling is

the process that produces a sequence of data from input. So it is very helpful in text prediction as it predicts the next word based on the sequence of previous words in the text. The most known and helpful sequential models are RNN and its variants [4][5].

3.3.2 NN

It is a model that tries to mimic how the human brain works. It consists of many neurons that take output from previous neurons with different weights and generate an output that will be used in the next neuron as in Figure 3.1, after that the weights are updated using a back propagation algorithm to decrease the error of the model.

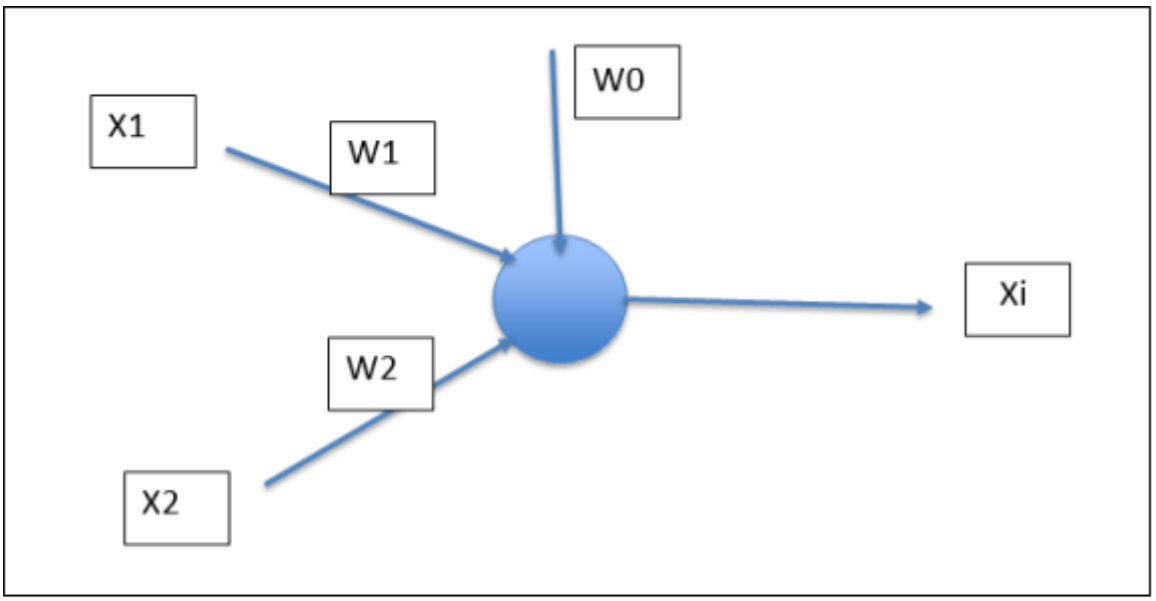


Figure 3.1: An illustration of a neural network neuron

$Xi = f\left(\sum_{j=1}^{N} W_j X_j + W_0\right)$ where Xi is the output of neuron, Wj is the weights of each input, W0 is the bias, f is the activation function. Activation functions can be:

- Relu
- Softmax
- Tanh
- Sigmoid

3.3.3 RNN

RNN is used a lot in NLP as it is very good with sequential data. It is better than CNN as it has internal memory so its weights are shared over time.

RNN can be used in different tasks:

- One To One: It is the classical feed-forward neural network as illustrated in Figure 3.2.
- One To Many: It is an image captioning model as it generates caption and text depending on the components of image as illustrated in Figure 3.2.
- Many To One: This can be used in emotion classification as it takes a sequence of words in a text and classifies the text as a single emotion as illustrated in Figure 3.2.
- Many to Many: This can be used in Text translation such as in Google translate and can also be used in text generation as it takes a sequence of input and outputs a sequence of output as illustrated in Figure 3.2.

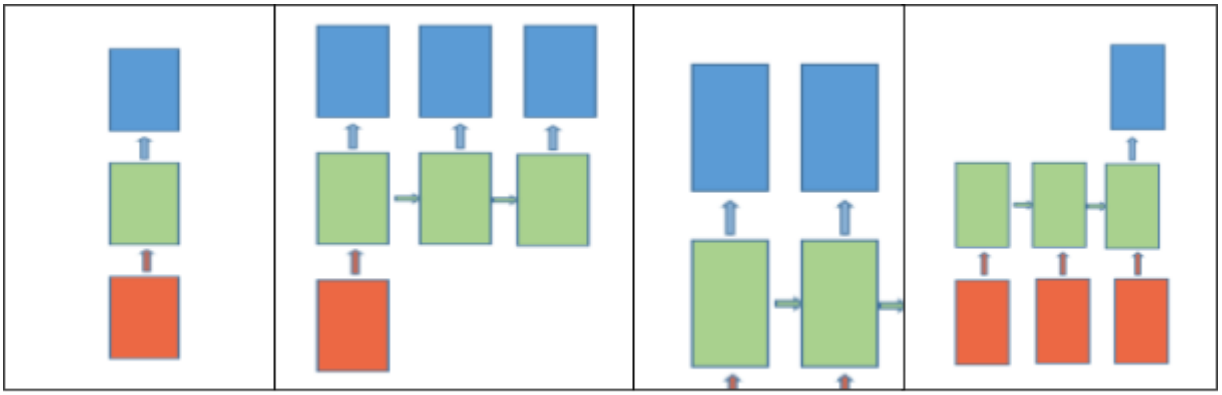


Figure 3.2: One To One, One To Many, Many To One, and Many To Many RNNs respectively

Despite the importance and powerfulness of RNNs, They are not very good at dealing with long range dependencies. This problem is Called vanishing Gradient Problem.

3.3.4 LSTM

LSTM is a popular deep learning technique and used in many popular NLP applications like SIRI. It was also the solution for the vanishing gradient problem as it modifies the hidden layers of RNN so they can remember the input for a longer time.

3.3.5 Bigrams

Bigram is a sequence of 2 adjacent tokens that are extracted from the text.

3.3.6 Edit Distance

In NLP you cannot guarantee that the user is going to write all of his words with the right spelling. There must be some errors in the words so if you are looking for a specific word you should be able to search for it in the text given not exactly as it is but maybe with some changes.

Edit distance is a way of measuring how dissimilar 2 words are by calculating the number of edits (inserts, deletes, replaces) from one word to reach the other word. Consider the following examples:

سريع -> ساعة

- سريع -> سيع (removing ر)
- سيع -> ساع (replacing ي with ا)
- ساع -> ساعة (inserting ة)

So, the edit distance between (سريع) and (ساعة) is 3.
if 2 words are so similar (edit distance ratio to the length of the word is less than a threshold) we consider these 2 words the same. For example,(حقيبة -> حقيبة) they would be considered the same word.

3.4. Background on Natural Language Generation

We are trying to develop an open-domain chatbot which can generate a natural response reflecting the conversation context. The meaning of "multi-turn" is very important to explain because it directly affects the difficulty level of the possible solution.

Assuming two speakers, 1 and 2, exchanging one utterance each is called a single-turn. So, the multi-turn dialogue consists of several turns as explained in Figure 3.3. Another important consideration is that each speaker should understand the context of the conversation as explained in Figure 3.4. So, we need to generate a proper response according to current input, but also refer to the overall contexts of the dialogue properly.

3.4.1 RNN

There are several approaches to generate natural language for the multi-turn dialogue chatbots. One of the traditional approaches in the literature is the Recurrent based approaches, such as RNN, which are still presented in the literature as the baselines for more complex better approaches like Mensio et al., 2018 [6], Chen et al., 2018 [7], and Serban et al., 2016 [8].

3.4.2 Seq2Seq

A sequence-to-sequence model is a model that takes a sequence of items (words, letters, features of an image…etc) and outputs another sequence of items. Seq2Seq models were introduced in 2014[9][10] and achieved a lot of success in tasks like machine translation as explained in Figure 3.5. Google Translate started using such a model in production in late 2016.

Figure 3.3: The difference between single-turn and multi-turn dialogues

Figure 3.4: The difference between context-aware and context-unaware dialogues

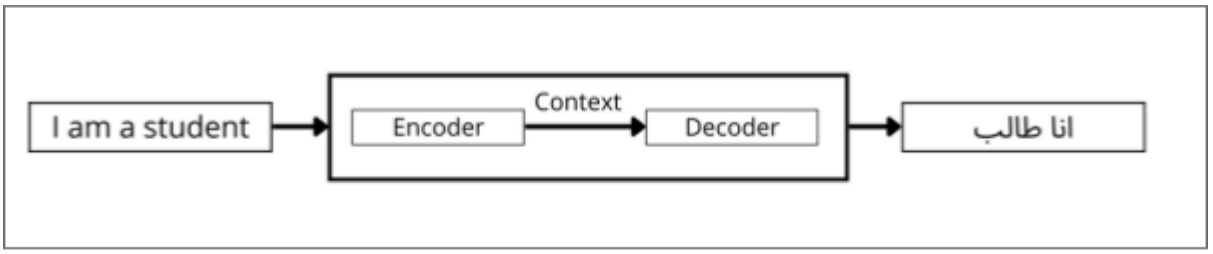


Figure 3.5: An example of a seq2seq translation model

The model is composed of an encoder and a decoder. The encoder processes each item in the input sequence, it compiles the information it captures into a vector called the context. After processing the entire input sequence, the encoder sends the

context over to the decoder, which begins producing the output sequence item by item. The encoder and decoder tend to both be RNNs.

The size of the context vector is a hyperparameter that is basically the number of hidden units in the encoder RNN and usually in real world applications the context vector would be of a size like 256, 512, or 1024.

3.4.3 Attention

RNN based sequence-to-sequence models have garnered a lot of traction ever since they were introduced in 2014 and were adapted in a variety of NLP tasks such as Machine Translation, Text Summarization, Speech Recognition, Question-Answering System, Language Generation, and so on. In 2015, Luong et al. [11], took a further step to enhance the performance of seq2seq models with the addition of the Attention Mechanism. Attention is a technique that allows the model to focus on the relevant parts of the input sequence as needed.

The attention mechanism depends mainly in two steps; the first step is to pass all the hidden states generated by the encoders to the decoders instead of passing the last hidden state only. The second step is to focus on the parts of the input that are relevant to a decoding time step.

An illustration of the focus step in explained in Figure 3.6 in which we:

1. Look at the set of encoder hidden states it received where each encoder hidden state is most associated with a certain word in the input sentence.
2. Give each hidden state a score.
3. Multiply each hidden state by its soft maxed score to amplify hidden states with high scores, and drown out hidden states with low scores.

3.4.4 Transformer

In 2017, Vaswani et al. [12], introduced the Transformer architecture in their popular paper, Attention Is All You Need. "The Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution" [12].

Transduction means the conversion of input sequences into output sequences. The idea behind Transformer is to handle the dependencies between input and output with attention and recurrence completely. The Transformer’s model architecture is illustrated in Figure 3.7. The biggest benefit comes from how The Transformer lends itself to parallelization.

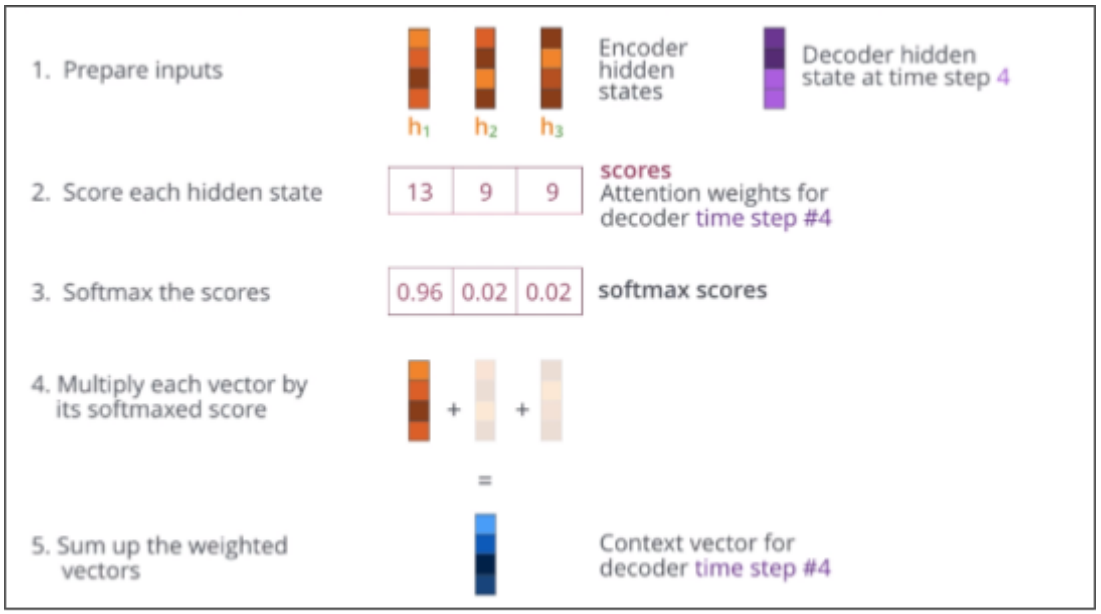


Figure 3.6: An explanation of the attention mechanism

The encoding component is a stack of encoders and the decoding component is a stack of decoders of the same number. The number of the encoding stack is a hyperparameter that can be tuned during training. Figure 3.8 shows an encoding and decoding stack consisting of 6 components which is the same size used in the paper.

Each encoding component consists of two layers. It receives a list of vectors as input to process this list by passing these vectors into the self-attention layer, then into a feed-forward neural network, then sends out the output upwards to the next encoder. One benefit of the Transformer is that the feed-forward layer does not have dependencies between the paths of each word vector thus the various paths can be executed in parallel while flowing through the feed-forward layer. Figure 3.9 shows an illustration of the internal layers of the encoder and decoder components.

Self-attention is the method the Transformer uses to bake the understanding of other relevant words into the one we're currently processing. To calculate the self-attention of a word, we create a Query vector, a Key vector, and a Value vector. These vectors are created by multiplying the embedding by three matrices that we trained during the training process. These vectors are used to calculate the word score by taking the dot product.

The paper further refined the self-attention layer by adding a mechanism called multi-headed attention. This improves the performance of the attention layer in two ways:

1. It expands the model's ability to focus on different positions.
2. It gives the attention layer multiple representation subspaces.

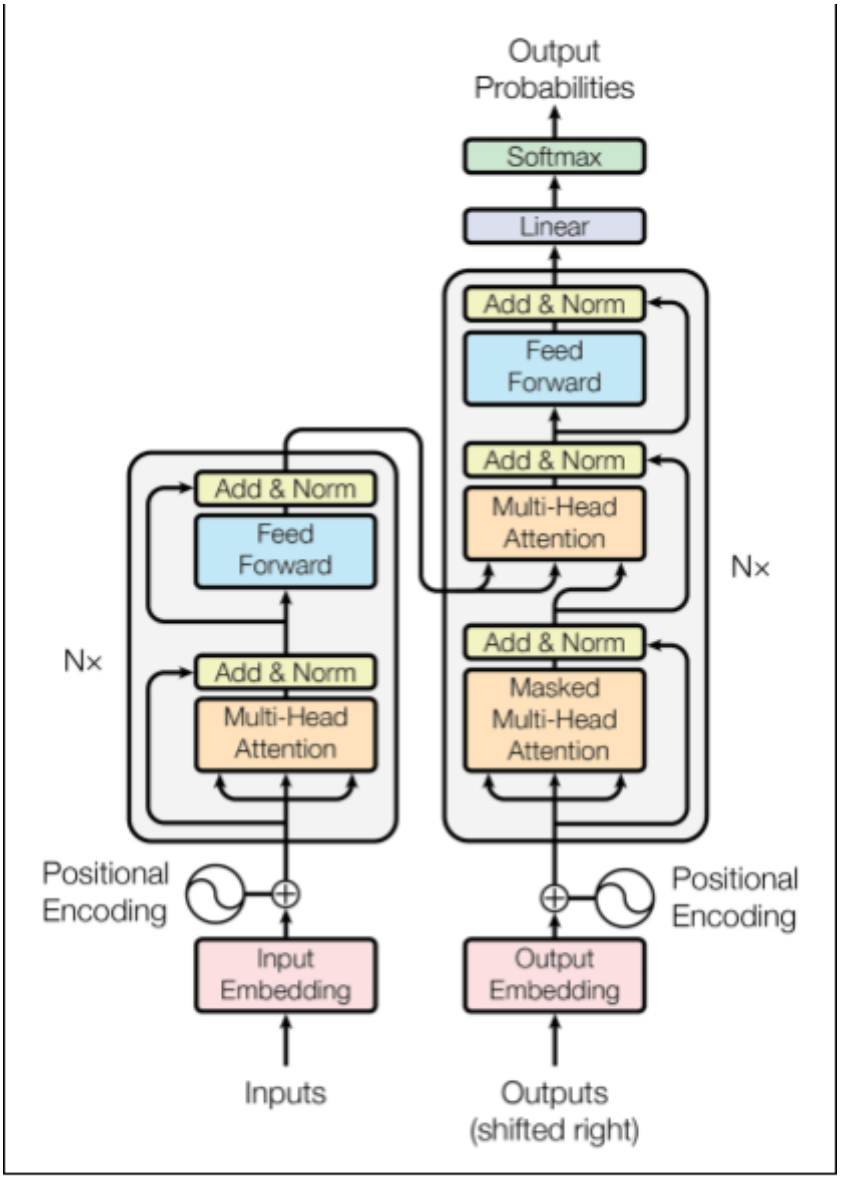


Figure 3.7: The Transformer's model architecture

With multi-headed attention we have not only one, but multiple sets of Query/Key/Value weight matrices. Each of these sets is randomly initialized. Then, after training, each set is used to project the input embeddings into a different representation subspace. The Transformer uses eight heads in the paper. The feed-forward layer is expecting a single matrix (a vector for each word). So we concat the matrices then multiply them by an additional weights matrix as illustrated in Figure 3.10.

The self-attention layer in the decoder differs from the one in the encoder. The decoder, however, uses what is called masked multi-head self-attention. This means that some positions in the decoder input are masked and thus ignored by the

self-attention layer because the decoder should not know which word comes after the predicted word.

The encoder-decoder self-attention layer works very similarly to the self-attention layer in the encoder. However, the query vector comes from the previous masked self-attention layer, the key and value vector come from the output of the top-most encoder. This allows the decoder to take into account all positions in the input sequence of the encoder.

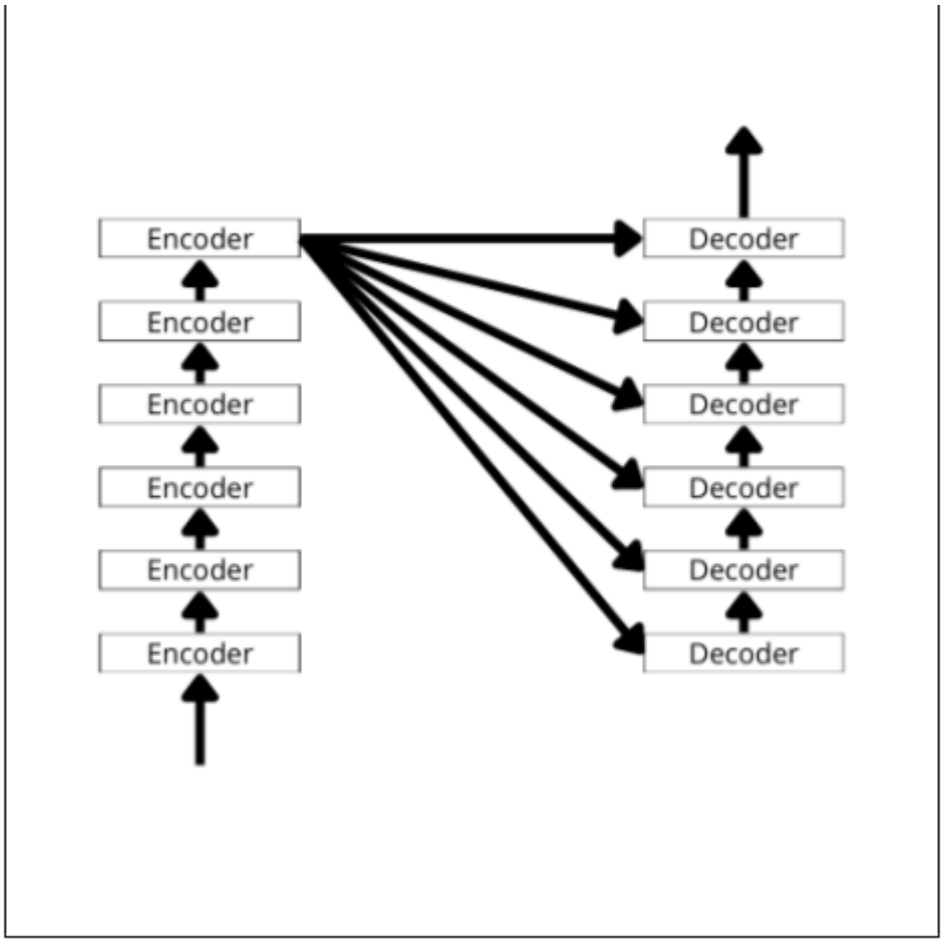


Figure 3.8: The encoders-decoders stack in the Transformer architecture

To sum up, the Transformer model is a new kind of encoder-decoder model that uses self-attention to make sense of language sequences. This allows for parallel processing and thus makes it much faster than any other model with the same performance.

The Transformer model can easily refer to all positions through matrix multiplication and extract only the necessary information through the attention scores. That is, we can get context vectors through the multi-head attention in utterance or history level to make better actions efficiently. They thus paved the way for modern language models such as BERT, ReCoSa, and GPT.

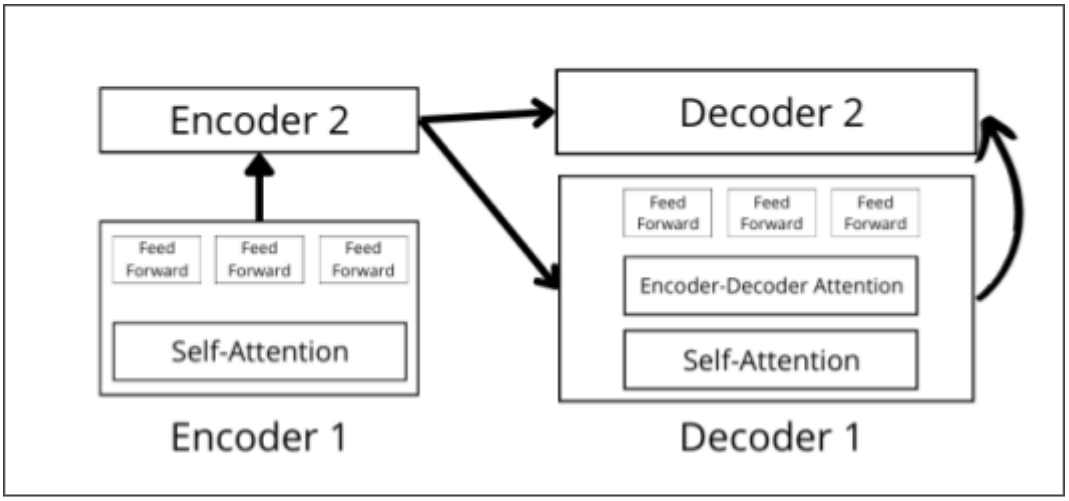


Figure 3.9: The internal of the encoder and decoder components

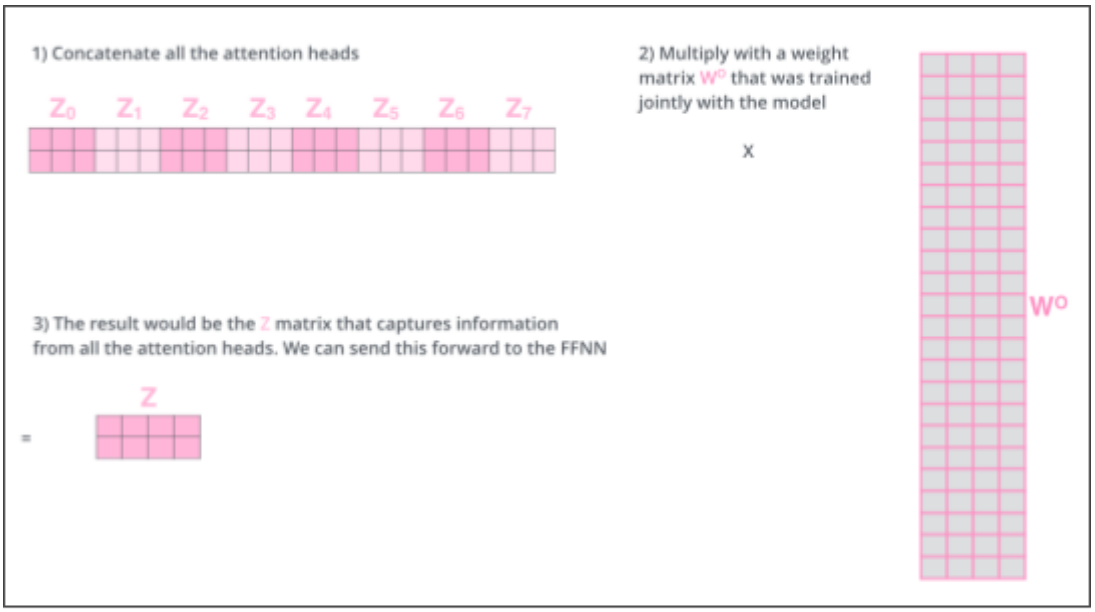


Figure 3.10: The result matrix of the multi-head attention

3.5. Comparative Study of Previous Work

We read about multiple approaches for open-domain multi-turn conversation that're based on the Transformer architecture as it's considered one of the state-of-art approaches in the literature. Three approaches specifically we found more interesting and clearer enough to understand. We will describe each paper's approach and the differences between them.

In 2019, Zhang et al. [13], introduced the ReCoSa architecture as illustrated in Figure 3.11. The decoder, in blue, is actually the same as the original since it applies

Masked Multi-head attention to the current generated sequence so far and after that attention with the encoder output.

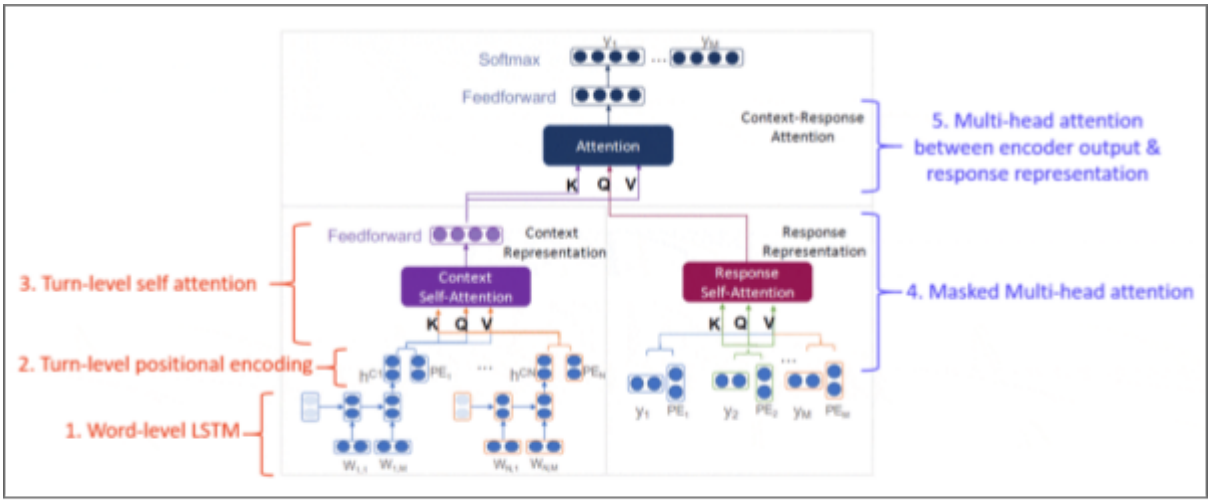


Figure 3.11: The ReCoSa architecture

In the encoder, in red, the word-level encoding is conducted by an LSTM and the last hidden state from this becomes the utterance embedding. The turn-level positional encoding is used to reflect the temporal sequence of each turn. The subsequent course consists of multi-head attention between each turn, similar to the original encoder, where we can obtain the encoder output that reflects the importance of each history.

The results show that the model fits very slowly. Compared to the results shown in the paper, the model is under-trained.

In 2018, Google released the BERT architecture [14] and in 2019, OpenAI released the GPT2 architecture [15]. The GPT-2 was trained on a massive 40GB dataset called WebText that the OpenAI researchers crawled from the internet as part of the research effort and the model has many variants starting from 117M parameters and 768 dimensions going up to 1542M parameters and 1024 dimensions.

The GPT2 is built using transformer decoder blocks. BERT, on the other hand, uses transformer encoder blocks as illustrated in Figure 3.12. The one key difference between the two is that GPT2, like traditional language models, outputs one token at a time.

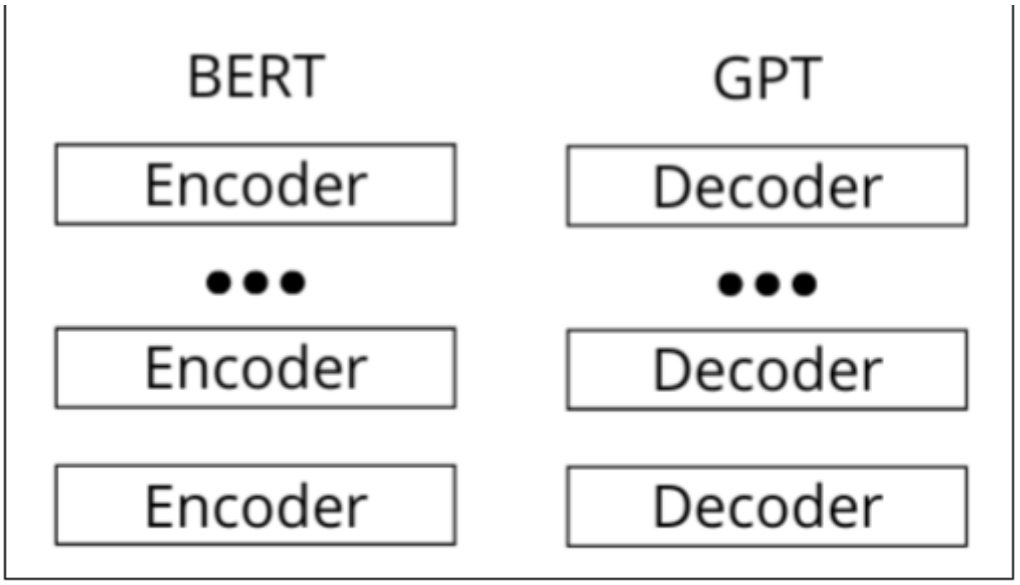


Figure 3.12: BERT vs GPT architectures

BERT and GPT-2 perform quite differently depending on the problem. For example, for a token prediction problem, for a fixed sequence length of 100 tokens, BERT performs best when the masked oken is between positions 5 and 95, while GPT-2 tends to continually improve as context length increases. Interestingly, when the final token in the sequence is to be predicted, BERT's performance falls off dramatically, while GPT-2 performance remains stable [16].

For Emotion Classification there were many approaches such as TF_IDF with logistic regression and LSTM deep learning model, LSTM approach is more accurate and more reasonable as it is excellent in data sequence while TF_IDF and ML depends on statistical values calculated from data which can be very misleading.

3.6. Implemented Approach

NLU module was implemented from scratch, did not use nltk or any other NLP models from text preprocessing as their performance in Arabic language is very bad.

Emotion Classification was implemented using TF_IDF and logistic regression, which was implemented from scratch, but the implemented logistic regression had less accuracy so we used the sklearn model as it also takes less time.

Intent Classification was implemented using a sequential LSTM model not TF_IDF with logistic regression as LSTM had better accuracy and Logistic regression suffered from overfitting.

Weather and Schedule used time extract which was implemented from scratch using bigrams , Weather used NER extracted from NLU to get location then used API to get weather. Schedule then extract content from the text using tokens not used in time extract.

For the places recommendation module, we use Google Places API because it can be considered as one of the best available API's. It provides all the services we need with a limited free quota.

And for the movies recommendation module, we use Item-based Collaborative Filtering because it can give better results than the user-based one, especially when we have a good amount of high quality data.

The Natural Language Generation module was implemented using the GPT2 architecture implemented by Huggingface [17] because the GPT2 resulted in the best performance based on the test results and the comparative study we conducted in the previous section. Also, Huggingface fine-tuned the GPT2 with the original language modeling task and the binary classification which determines whether the given response is a proper one or not.

Chapter 4: System Design and Architecture

Anees is an Arabic chatbot that can speak on different topics, analyze emotions of users, and do specific tasks. First we take the text written by the user, apply some preprocessing and extract tokens, also we extract NER and POS. Then we use the preprocessed text to classify the user's intention if he/she wants to speak in general or wants to do a specific task. If intent is general, a deep learning model will reply ,but if the text has an NER of organization and the text is a question we use a search module. If the intent is weather, we extract location and time from it and use an api to get the weather of this location at the given time. If intent is scheduled, we extract the time and content to be scheduled in the calendar. If intent is a recommendation. then we use another intent classification model to classify if this recommendation is for movies or locations.

4.1. Overview and Assumptions

Anees project consists of 6 major modules that need to be understood before starting implementing such a thing. The modules are natural language understanding, emotion classification, intent classification, weather/schedule, recommendation, and natural language generation. Apart from that, there is the connection between each module and the other. Each of these modules are described in detail in the following section, and how they all connect to each other, and how they represent the system architecture.

Some assumptions are also considered in order to simplify some of the hardships of the implementation. In the GPT, due to the complication of its design and architecture and its need for a heavy-working machine to train on, we simplify a few things in it. The GPT's maximum length is 1024, we decreased it to only 256 and then decreased the batch size to 8 instead of 32. Other modules have some assumptions and constraints described in each module below.

4.2. System Architecture

In this section we will discuss the architecture of the project which consists of 8 modules as shown in Figure 4.1.

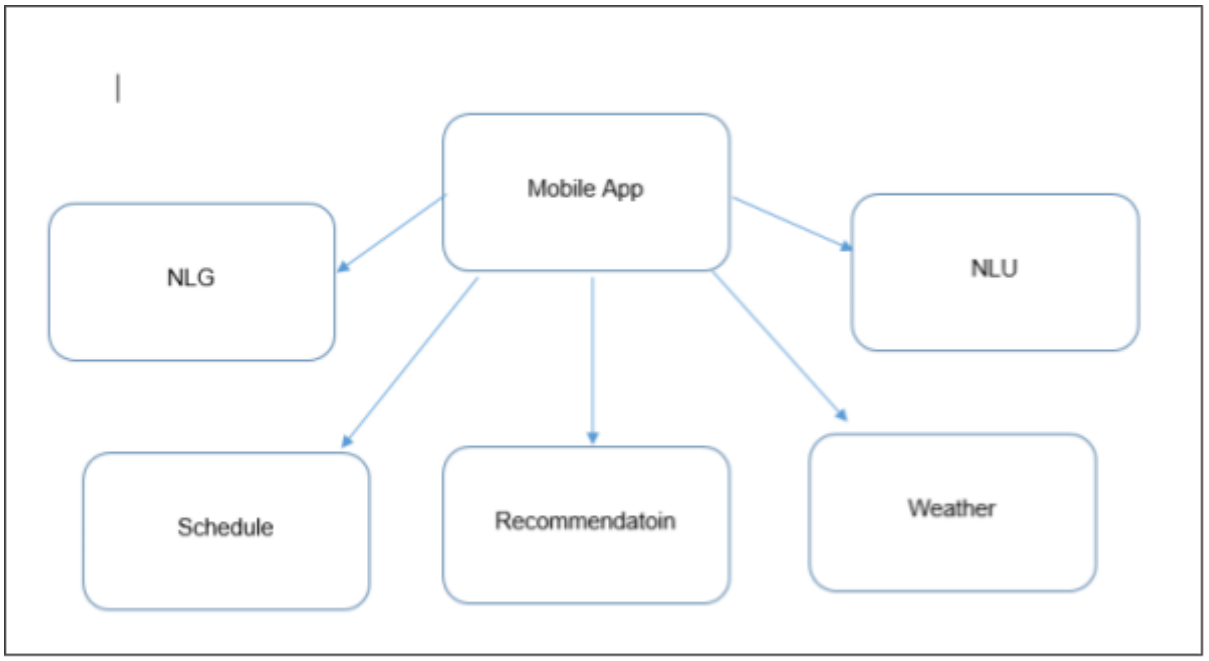


Figure 4.1: Simplified block diagram

4.2.1. Block Diagram

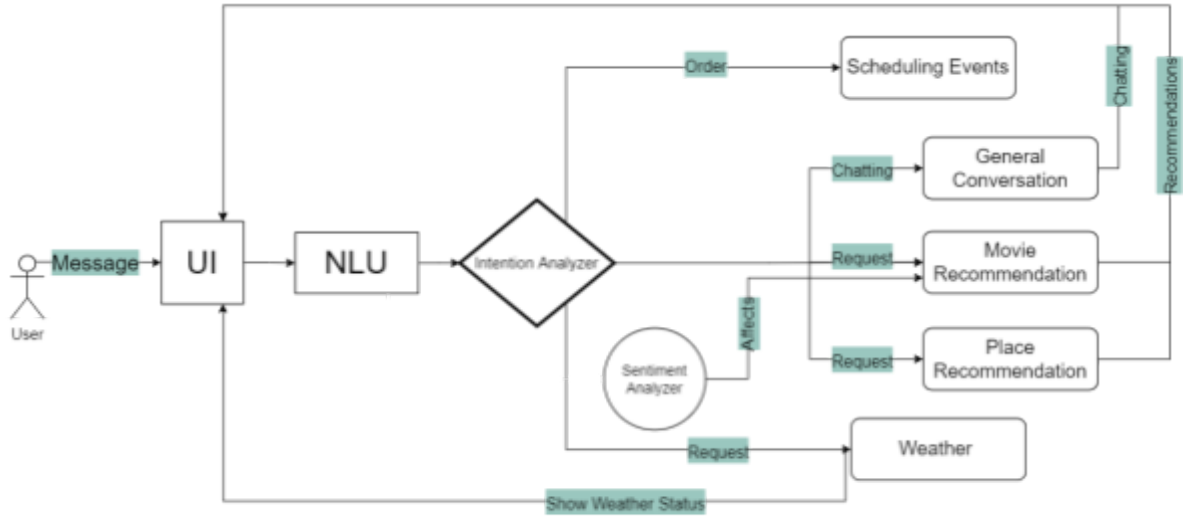


Figure 4.2: Detailed block diagram

4.3. Natural Language Understanding Module

This module is responsible to preprocess text, normalize it, extract tokens and NER

- **Input:** The text written by user
- **Output:** The preprocessed text, tokens, NER, POS, and tokens after stemming

4.3.1. Functional Description

It has many functionalities that helps in extracting features from text to be used in performing tasks and the functionalities are:

- Preprocessing
- Tokenization
- NER
- POS
- Stemming

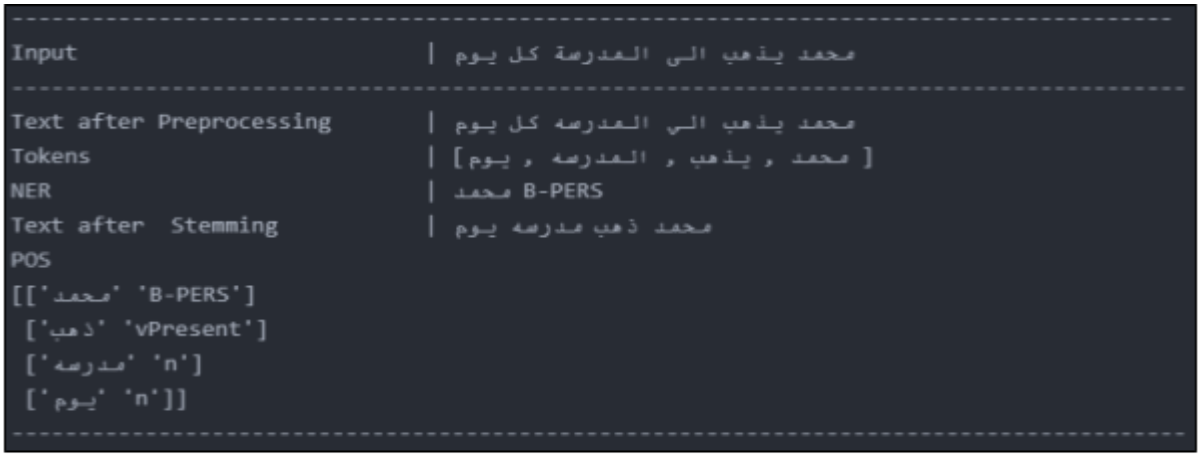


Figure 4.3: NLU functionalities

4.3.2. Modular Decomposition

It is composed of many components which are:

- Preprocessing
  - Remove all punctuation
  - Remove diacritics
  - Normalize letters
  - Change all numbers to Arabic numbers
- Tokenization
  - Extract tokens from text
  - Remove stopwords from tokens
- NER
  - Extract Person, Organizations, Locations
- POS
  - Classify token as verb or noun
- Stemming
  - Remove added suffix and prefix letters to token

4.3.3. Design Constraints

Text should be in Arabic and also with no mistakes so can be recognized by datasets and models used.

4.4. Emotion Classification Module

This module is responsible for extracting the emotion of the user from the written text to be used in conversation and recommendation modules.

- **Input**: The text
- **Output:** The emotion of the user

4.4.1. Functional Description

It classifies the text written by the user into one of the following emotions (joy, surprise, fear, anger, sadness), the text is classified after performing NLU techniques.

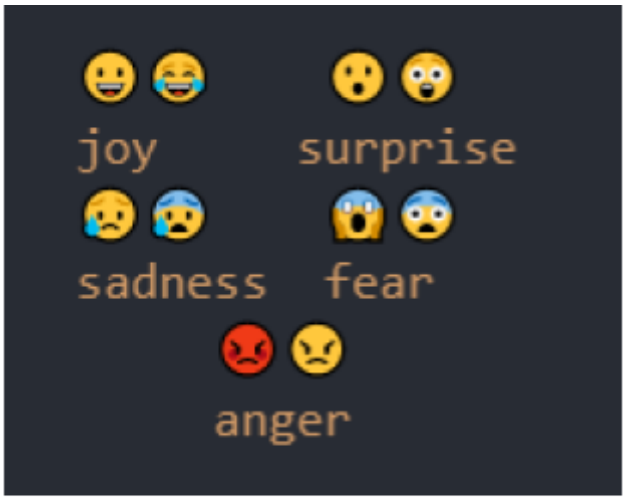


Figure 4.4: The classified emotions

4.4.2. Modular Decomposition

This module is composed of some stages:

- NLU techniques as discussed before
- TF_IDF: to determine the most important words for each emotion
- Logistic Regression: ML model used to predict the probability of each emotion

4.4.3. Design Constraints

Non-empty text with at least 1 word not from stopwords like (‘فى’) and it is preferable to be Arabic text so the module can work correctly.

4.5. Intent Classification Module

This module is a critical module in the system as it determines the intent behind the text so we direct to the suitable module or task to run.

- **Input:** The text
- **Output:** The intent of the user

4.5.1. Functional Description

This module is used to extract the intent of the user from the text written. The intents are:

1. General
2. Weather
3. Schedule
4. Recommendation
    a. Locations
    b. Movies

4.5.2. Modular Decomposition

First some NLU techniques (Preprocessing and Tokenization) is applied to text to get rid of Stopwords and normalize the text. A Deep Learning Sequential LSTM model with an output Dense layer of 7 units and softmax activation function was trained on a dataset after using a tokenizer to build vocabulary to be used and converting text into sequence, then this model is used to classify the text intent to be directed to the suitable module or task to run.

If the intent is to recommend another LSTM model but with a dense layer of units 2 is used to determine if the recommendation is locations or movies.

4.5.3. Design Constraints

Non-empty text with at least 1 word not from stopwords like (‘فى’) and it is preferable to be Arabic text written in a correct way so the module can work correctly.

4.6. Weather, Schedule

This module is a meant to perform the task itself after knowing the intent of the user if it is to get the weather or to schedule a meeting in the calendar

- **Input:** The intent of the sentence only, the tokens and the stemmed tokens
- **Output**: The text that should be sent to the user as a response to his need

4.6.1. Functional Description

This module is divided into 2 submodules:

- Time extraction module
- Content extraction module based on the intent

4.6.2. Modular Decomposition

The module extracts the time the task is meant to be performed in (when to schedule the meeting or when to get the weather) by searching for keywords that express time like (شهر , يوم , ساعة) etc and using the edit distance between the tokens of the word and each one of these time expressing words we find the closest word to these time keywords.

Then get the bigrams of the tokens and process only those with time expressing keywords in and try to find a **number** around this keyword also using the edit distance and if found set this time by this number. Consider the following examples:

- (4 الساعة) set the hour to be 4
- (بعد 5 ساعات) increment the hours by 5 hours

Figure 4.5: Extract time

After extracting the time and marking the tokens used now it is time for the content extract module:

- First of all, if the intent is **scheduling** it extracts the title to be stored in the calendar event by
  - Removing the tokens used in time extract
  - Removing any order verbs from the sentence
  - Getting the longest subsequence of the unused tokens

Figure 4.6: Extract content

- If the intent is **weather** we should get the location needed to get the weather in using the NER to find the place in the sentence if not any place specified use his current location of the user.

Figure 4.7: Extract weather

4.6.3. Design Constraints

The module could go wrong if the words are misspelled with more than 1 character, The time given should be in the future not the past as we schedule a future event or get the weather at any time in the next 5 days.

4.7. Search and Places recommendation Module

This module is responsible for recommending several types of places to the user in different ways.

- **Input (two types of requests)**
  - The location of the user
    - Radius of recommendation area around the location.
    - Type of place the user wants to visit.
  - Or just a plain search text.
    - The text can include the location of needed places and the type.
- **Output:** A list of recommended places with some detailed information about each place like name, rating, location, price-level, …etc.

4.8.1. Functional Description

It recommends some nearby places or places in a specific location. Also, It can depend on some attributes like place type and can be recommended based on location distance of importance. Finally, plain text can be used to recommend things like **"مطاعم في مدينة الجيزة"**.

4.7.2. Design Constraints

It can only return 60 places as a maximum.

4.8. Movies Recommendation System Module

It is the module responsible for recommending proper movies to the user based on his/her request.

- **Input (two types of requests)**: A movie to get similar to it based on rating similarities or some categories that the movies are related to.
- **Output:** A list of recommended movie names.

4.8.1. Functional Description

It supports two kinds of recommendations. The first one can recommend a list of movies based on a given movie that the user likes. It does so based on ratings. While we have a set of movies and know ratings given to them from a set of users, we recommend a list of movies have a combination of ratings by users that is similar (the combination) to the combination of ratings to the given movie.

So, If a set of users give similar combinations of ratings to movies X and Y, then these movies can be similar. We use KNN to get the most similar movies to the given movies. The second way of recommendation is done using categories. Given a set of categories we can recommend a list of movies those are labeled by this set of categories. We use relevance values given to each pair of movie and categories to know how much the movie is relevant to this category.

4.8.2. Design Constraints

Data amount is a major factor. The data is mainly defined as movies, ratings and users who rated these movies. To build the model, we need to build a pivot matrix which is a 2D matrix where movies take an axis and users take the other while cells are filled by the rating values.

To make the model supporting more movies with higher accuracy, we need to increase the number of movies and users which highly affect the size of the matrix, so we can not use all data we have as we have limited resources. Also, we use the KNN model that will have response time increasing while we increase data size.

4.9. Natural Language Generation Module

In this section, we will discuss the NLG module which is responsible for generating the proper responses for the input utterance considering the context of the conversation.

- **Inputs**: The current utterance that's written by the user and the history or previous turns between model and the user.
- **Outputs**: A response to the utterance with the same conversation context.

4.9.1. Functional Description

Generation of a proper response according to current input, but also how to refer to the overall contexts of the dialogue properly.

Figure 4.8: Response generation example

4.9.2. Modular Decomposition

The NLG module can be divided into 5 submodels:

- Data Loader
- Preprocessing
- Trainer
- Nucleus Sampling
- API

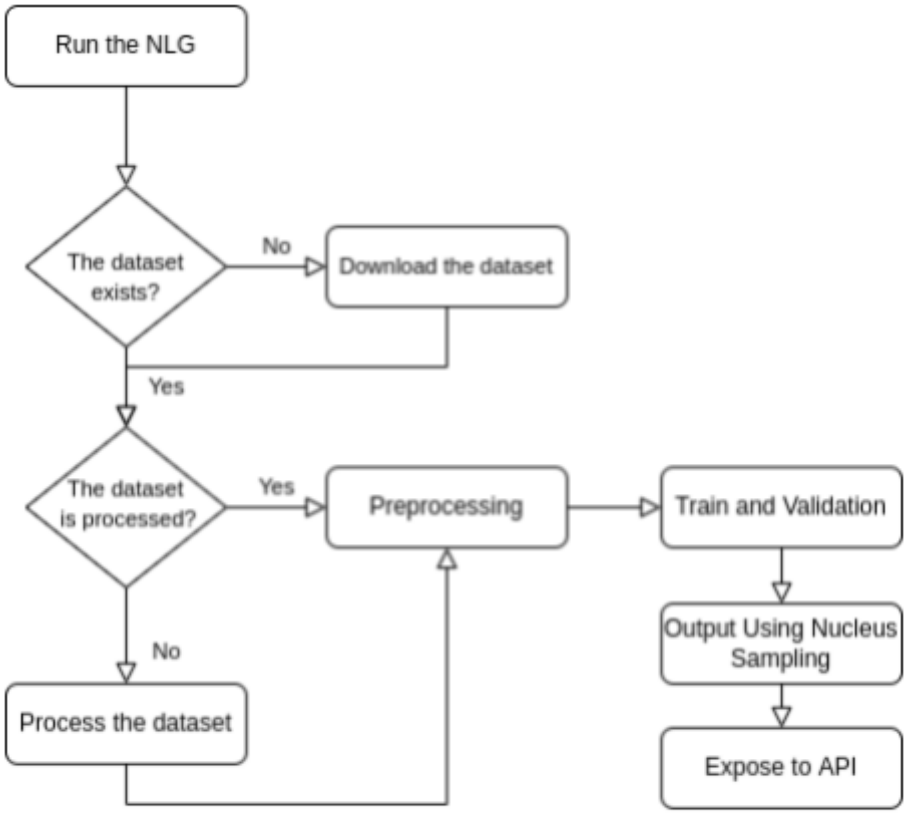


Figure 4.10: A flowchart of the NLG module

4.9.2.1 Data Loader

One of the major problems in deep learning models is the size of the data used for training. After a detailed search in the available public datasets for the dialogue training, we decided to use different four datasets; DailyDialog [18], EmpatheticDialogues [19], Persona-Chat [20], and BlendedSkillTalk [21]. Also, we tried to train an Egyptian dialect model so we built an additional private dataset using our private chats on Messenger and WhatsApp. We divided the public datasets into 0.85 and 0.15 for the training and validation sets respectively. The private dataset was divided into 0.90 and 0.10 for the training and validation set respectively based on the number of conversations.

	# of dialogues in training set	**# of utterances in training set**	**# of dialogues in validation set**	**# of utterances in validation set**
DailyDialog	11150	87467	1968	15512
EmpatheticDialogues	19628	84674	3464	14912

Persona-Chat	16046	21283	2832	37788
BlendedSkillTalk	5786	76435	1022	13482
Total	52620	461449	9286	81694
PrivateChats	42488	262440	4720	23600

Table 4.1: The details of the datasets size

The processing of the dataset construction includes unifying the format of all datasets into a fixed format and translation of the combined dataset because the original one is in English so we had to translate it into MSA. Finally, the result of this submodel is two files for the training and validation sets.

4.9.2.2 Preprocessing

We implemented custom preprocessing for the utterances used as inputs to the generation model. The preprocessing included:

- Remove URLs
- Remove emojis
- Remove HTML tags
- Remove extra spaces
- Remove repeating characters
- Remove non-arabic characters
- Remove the tashkeel of words
- Remove Hindi digits and replace it with Arabic ones

Finally, each utterance is tokenized and the model outputs two files for each input file; the first one is the processed utterances and the second one is the tokenized utterances.

4.9.2.3 Trainer

The trainer is responsible for training the model and validating the results. So, we put inputs to the GPT2 Language Modeling Head model after the preprocessing. We added the perplexity calculation to evaluate the training and validation.

The original loss used in the GPT2 model is the cross entropy loss which increases as the predicted probability diverges from the actual label. The cross entropy loss is given by:

$$-\sum_{c=1}^{M} y_{o,c} log(p_{o,c})$$

The perplexity is an evaluation method for Language Modeling which indicates how the model chooses the next tokens with high probabilities. In other words, it indicates the quality of language modeling. Perplexity is the reciprocal of normalized test data's probability as explained in the following equation. We can see that the higher the probability is, the lower the perplexity becomes, which means that the model performance is more decent.

$$PPL = \sqrt[n]{\frac{1}{\prod_{i=1}^{N} P(w_i|w_1, w_2, ..., w_n)}}$$

We can calculate the perplexity using the cross entropy loss by getting the negative log loss normalized to the input sequence length and the value inside this negative log is the joint probability which has already passed through the softmax function. So by taking the exponent, the perplexity can calculated by:

$$e^{Loss} = e^{-\frac{log\ P}{N}} = e^{log\ p^{\frac{-1}{N}}} = p^{\frac{-1}{N}} = \sqrt[n]{\frac{1}{N}}$$

4.9.2.4 Nucleus Sampling

The trainer outputs decoded sequences so we need a proper decoding algorithm. We used the Nucleus Sampling a.k.a. Top-k algorithm because we found it more appropriate for the open-ended generation. Nucleus Sampling chooses the next word by extracting the subgroup that accounts for a certain mass portion of total word distribution and sampling one randomly from it. This algorithm helps to choose high-quality words within a wide set of options. We implemented it based on the implementation by Thomas Wolf, the science lead at Hugging Face [22].

4.9.2.5 API

The API works as the interface for other modules to use the natural language generation module. So, we expose two public API endpoints to use the MSA and Egyptian Dialect models. An endpoint expects the current utterance and the previous history.

4.9.3. Design Constraints

The module constraints are mainly related to the training process and the hardware used for training the GPT2 mode. We can summarize the constraints as the following:

- Maximum batch size: 8
- Maximum Length: 256
- Maximum number of turns: 5

4.10. Interface Module

The interface module is responsible for introducing the chatbot for the end user in the form of a mobile application that the user can download from Google Play Store for Android or App Store for iOS. The module introduces an interface for the login, signup, chatting, ratings, and other features provided on the app.

4.11. Backend & Database Module

In this module we made the backend that connects the android application to the modules to send the conversations and organize the database. We used Python Flask as our backend server because we implemented our modules in Python so it would be easier to integrate the backend with the modules.

We used MongoDB as a NoSQL database because it is fast and it is schema less which helped us in adding or removing attributes from the collections. The backend has several endpoints:

- {POST , "/getResponse" } : get the response on a specific input text
- {POST ,"/login"}: login with a username and a password
- {POST ,"/signup"}: signup a new user
- {POST ,"/history"} : get the messages history for a specific user
- {PUT ,"/update_movie_rating"} : update the rating of a movie recommended and its categories
- {POST ,"/emotions"} : check the most used emotion and act according to it
- {PUT ,"/schedule_cancel"}: cancel the scheduling of an event as the calendar is full at this time

Chapter 5: System Testing and Verification

Testing is the process to know if the actual results are similar to the expected results or not and to make sure that the system has fewer bugs and is working correctly.

Each module was tested alone to know the performance of each and if it can be modified or its accuracy is fine. Then the whole system was tested on a mobile application by making a conversation with Anees and seeing if all modules are integrated correctly.

5.1. Testing Setup

Our project depends heavily on human evaluation of the output responses in all of the stages. Although we use some metrics such as the cross entropy and the perplexity to evaluate the quality of the generated text, it is hard to rely on numbers only because numbers could be deceiving. For example, theoretically, the lower perplexity is better but it also may mean that the size of the options are small so there are not many good options to choose from. So, human perspective evaluation is a major factor to determine what was a good result instead of relying completely on numbers.

5.2. Testing Plan and Strategy

To assess the quality of Anees we tested the core modules, each with a dataset taken from different sources in addition to the human evaluation by the team and our friends because as we explained that most outputs are textual outputs, which makes the most meaningful test by human judgment.

5.2.1. Module Testing

5.2.1.1 Natural Language Understanding Testing

This module was tested manually by giving it some text as input and seeing if it works correctly and making preprocessing, tokenization, NER, POS and stemming on the text given.

5.2.1.2 Emotion Classification Testing

Collected data was divided into 0.8 training data and 0.2 validation data and fitted the TF_IDF on the training data, then transformed validation using the fitted TF_IDF, then trained a logistic regression model on training data.

Accuracy on positive and negative labels was 85% which was very good, but on all 5 used emotions, the accuracy was not promising as the labels are 5 more than 2 (positive and negative). For example there are 2 labels considered positive joy and surprise, but many sentences can be considered both.

5.2.1.3 Intent Classification Testing

Collected data was divided into 0.8 training data and 0.2 validation data then data was preprocessed by removing stopwords, then a tokenizer was fitted on training data, and to build vocabulary dictionaries, then each sentence is converted to sequence of numbers as models cannot understand strings.

A LSTM sequential model was trained with dense layer of 7 units (6 intens + 1 default) and softmax as activation function and the accuracy was 99% on training and 98% on validation, but this accuracy was on the data collected and when tested using some random sentences it appeared that the accuracy is lower.

5.2.1.4 Weather and Schedule Testing

This module was tested manually by giving it some text as input and seeing if it works correctly and extracted time and content correctly for schedule task and extracted time and location correctly for weather tasks before using API to get weather.

5.2.1.5 Search and Places Recommendation Testing

This module was tested manually by giving it some text as input and seeing if it worked correctly and returned results corresponding to what the user wanted.

5.2.1.6 Movies Recommendation Testing

This module was tested manually by giving it some text as input and seeing if it works corrected. First categories are extracted from text and movie names are also extracted if the user mentioned a movie in text, then recommendation is done based on what was extracted from text.

5.2.1.7 Natural Language Generation Testing

The module was tested using both human evaluation to evaluate the quality of the generated responses and how much they proper for the conversation context. Also

we used the cross entropy loss and perplexity as we discussed to evaluate the numerical results as illustrated in Figure 5.1.

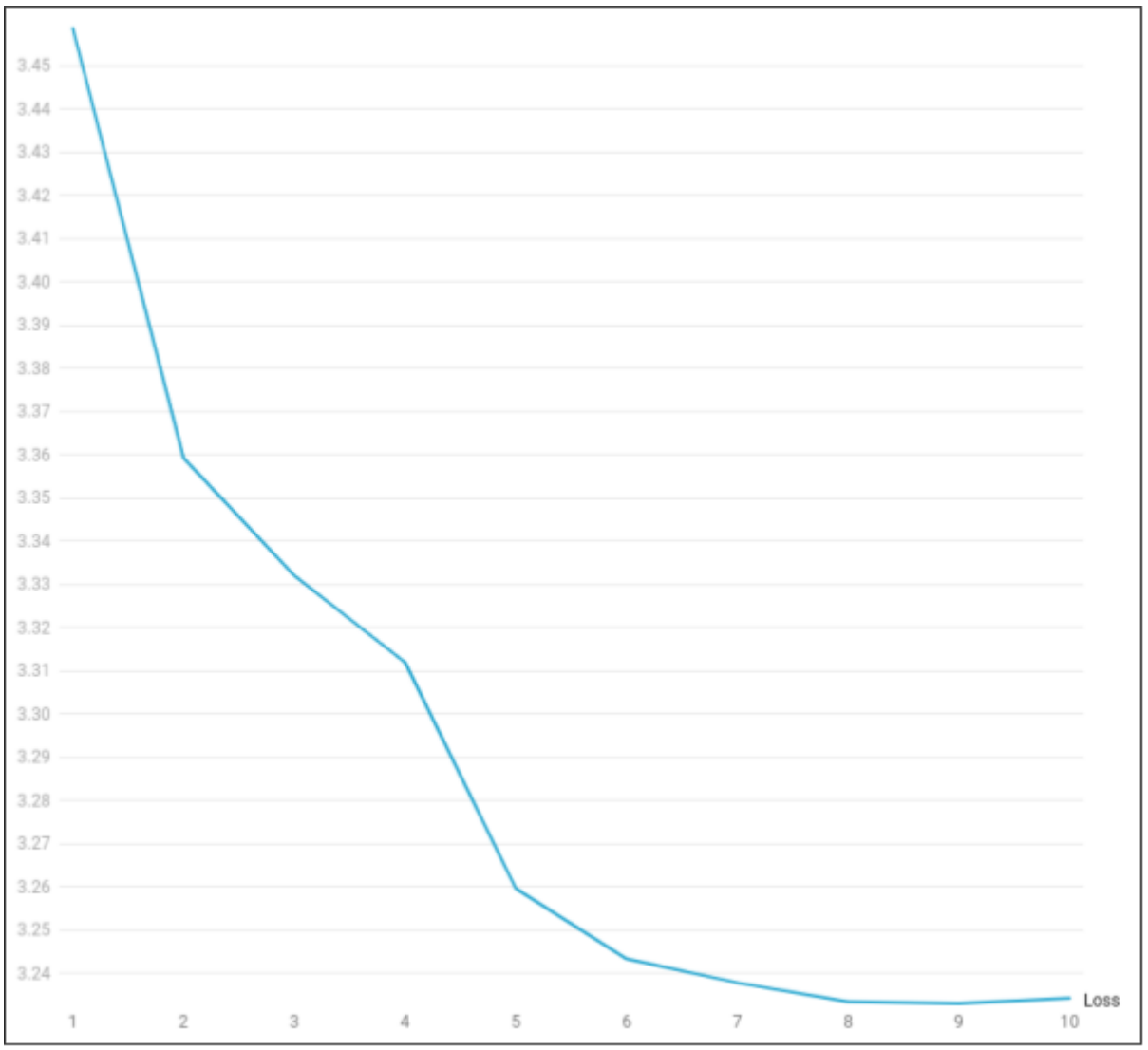


Figure 5.1: The evaluation of the language generation model

5.2.2. Integration Testing

After merging all our modules together and testing the whole project integrated using the Interface developed using React Native connected to an API developed using Python Flask, we noticed that the final results are highly dependent on few things:

- The length of the conversation. As the length of the conversation grows, the quality of the generated responses become better and more context-aware.

- The tasks provided by the chatbot like the emotion classification and the intent classification can be improved dramatically if we used deep learning instead of classical machine learning approaches but this is for future work.
- The amount of the dataset specifically for the Egyptian Dialect. This is a major limitation for the results as we need to collect more data and invest more time to get better potential of the datasets we have.
- The hardware resources; the GPU we train on. A fast one with a large memory size would be of a huge advantage.

Although our final project results were limited due to our resources, if the project is going for the market, we would have a paid server on a large cloud like Amazon Web Server or Google Cloud Platform, which would make it easier for the GPT2 to train on each user in a shorter time with a large maximum length and larger maximum turns, which would give a much better results.

5.3. Testing Schedule

Tested Module	Testing Date
NLU	Feb 2022
Sentimental Analysis	Mar 2022
Intention	May 2022
Language Generation	July 2022
Tasks	July 2022

Table 5.1: Testing schedule

5.4. Comparative Results to Previous Work

5.4.1. Language Generation Module

There are few public models that we can compare our model to. We found two popular implemented models for open-domain conversation and both of them are based on a BERT architecture that's trained for the MSA language and called AraBERT [23]. The first model and second models can be found in [24] and [25] respectively.

The common metrics between the three models are the cross entropy loss so we will compare based on it. Also, an important difference between our model and the others is that they trained for only single-turn responses while we are training for multi-turn open-domain conversation.

	Anees	**Model 1**	**Model 2**
Loss	3.2343	3.0164	2.7963

Table 5.2: Natural Language Generation comparison

Anees has a decent performance compared to other models considering the resource limitations that we discussed previously which indicates that Anees has a good potential for future improvements.

5.4.2. Natural Language Understanding Module

Text = "محمد يذهب إلى المدرسة"

Techniques	Our NLU	NLTK	SPACY
Tokens	[المدرسة,إلى ,يذهب ,محمد]	[المدرسة,إلى ,يذهب ,محمد]	[المدرسة,إلى ,يذهب ,محمد]
NER	[{Person : محمد}]	[{ORGANIZATION : محمد, ORGANIZATION : يذهب }]	[{Country : يذهب}]
Stemming	المدرسة --> مدرسة يذهب --> ذهب محمد --> محمد	المدرسة --> مدرس يذهب --> يذهب محمد --> محمد	محمد ==> محمد يذهب ==> يذهب المدرسة ==> المدرسة

Table 5.3: Language Understanding comparison

5.4.3. Emotion Classification Module

We used 2 approaches, one using TF_IDF with logistic regression and the other was LSTM sequential model, but we chose TF_IDF with logistic regression as TF_IDF has better accuracy and we thought it is better to use models depends on statistics

rather than deep learning in this module also because we wanted to use less deep learning in the project.

Models	Accuracy on 5 emotions	Accuracy on +ve and -ve
TF_IDF with Logistic regression	60%	85%
LSTM	55%	81%

Table 5.4: Emotion Classification comparison

5.4.4 Intent Classification Module

Also in this module, we compared TF_IDF with logistic regression and LSTM but here LSTM was better as the statistical approach is not good here so we used a deep learning model to have better accuracy as this module has a critical impact on the project.

Models	Accuracy on validation data	Accuracy when testing
TF_IDF with Logistic regression	96%	very poor as it gets most text as general
LSTM	**99.5%**	better but had problem with recommendation

Table 5.5: Intent Classification comparison

Chapter 6: Conclusions and Future Work

Virtual assistants are pretty complex systems that require good knowledge in different fields and hard work to build. By considering chatbots, they are highly dependent on the data used to train and how huge and clean it is. And by considering the Arabic chatbot (specifically the egyption one), we can find that there are real problems in both. The quality of data is really poor following few standards. Also, the available amount of data is relatively small to train a chatbot with a good accuracy. So, data is the major problem we faced while building Anees.

6.1. Faced Challenges

The major problem we faced is data amount and quality, specially for the Egyptian dialect. We also faced some problems and restrictions with resources for training and how much time it will consume, while we have a limited time. Another problem is that the egyptian dialect has a very different vocab and sentence structure than the standard arabic which made it harder to process the egyptian text.

The GPT model used in Anees is a large complicated one that needs high resolution data of large amounts, and these constraints make it harder to train on the available resources we have. It needs a fast GPU of large memory size which we didn't have. We used a 8GB GPU, which still was not able to save all the data at once. We were limited by many aspects; the data size, the number of epochs to train on, and time. All these things affected the output of the GPT.

Also, most of our implementations depend mainly on classical approaches techniques, it was difficult to find robust classical techniques resources that produce good quality results, due to the remarkable growth in the deep learning field,which made the classical approaches resources were rare and not up to date and not always accurate or correct.

Finally, to produce relatively good responses, we were not always able to do that because of the limitations of the classical approaches. So the performance was limited.

6.2. Gained Experience

We have learnt good experiences in the field like NLP, ML, and DL.
One of the most important skills we needed is Problem Solving to handle the difficult problems we faced (e.g. lack of data).

Reading dozens of research papers to achieve working results helped us to be exposed to multiple methods in every submodule, and to see the evolution in the techniques through time. We also noted the difference of the development and output between the classical approaches and deep learning approach. Despite the deep learning approach producing better results, at some points, it was more complicated to develop the classical approaches.

Finally, getting exposed to new technologies such as Transformers, GPT, NLP techniques, and working on a large project with multiple modules with a clear workflow as Anees.

6.3. Conclusions

We concluded that the topic of NLP needs more work specially in the Arabic language. Also the goodness of data and the amount of it are the most important aspects in training an NLP model.

6.4. Future Work

Anees has a good potential to develop more features in the future. We aim to improve the performance of general conversation and specifically for the Egyptian dialect and then support other Arabic dialects. Planning for including the gained data from the incoming conversations to retrain Anees and improve the accuracy.
We could also add the feature of the user using his voice to talk to Anees.

References

[1] Grand View Research, https://www.grandviewresearch.com/industry-analysis/intelligent-virtual-assistant-industry#, 2019

[2] Hamood Alshalabi, Sabrina Tiun, Nazlia Omar, Fatima N.AL-Aswadi, Kamal Ali Alezabi, "Arabic light-based stemmer using new rules", Journal of King Saud University - Computer and Information Sciences, 25 August 2021

[3] Hamdy Mubarak, "Build Fast and Accurate Lemmatization for Arabic", QCRI, Hamad Bin Khalifa University (HBKU), May 2018

[4] Vijaysinh Lendave, "A Tutorial on Sequential Machine Learning", Analytics India Magazine, 17 November 2021

[5] Ian Goodfellow, Yoshua Bengio, Aaron Courville, "Deep Learning", MIT Press, 2016

[6] Martino Mensio, Giuseppe Rizzo, Maurizio Morisio, "The Rise of Emotion-aware Conversational Agents: Threats in Digital Emotions", Companion of the The Web Conference, April 2018

[7] Hongshen Chen, Zhaochun Ren, Jiliang Tang, Yihong Eric Zhao, Dawei Yin, "Hierarchical Variational Memory Network for Dialogue Generation", Web Conference, April 2018

[8] Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio1, Aaron Courville, Joelle Pineau, "Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models", School of Computer Science, McGill University, 6 April 2016

[9] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, "Sequence to Sequence Learning with Neural Networks", Advances in Neural Information Processing Systems 27, 2014

[10] Kyunghyun Cho, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation", EMNLP, 3 Sep 2014

[11] Minh-Thang Luong, Hieu Pham, Christopher D. Manning, "Effective Approaches to Attention-based Neural Machine Translation", EMNLP, 20 Sep 2015

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, "Attention Is All You Need", 6 Dec 2017

[13] Hainan Zhang, Yanyan Lan, Liang Pang, Jiafeng Guo, Xueqi Cheng, "ReCoSa: Detecting the Relevant Contexts with Self-Attention for Multi-turn Dialogue Generation", Association for Computational Linguistics, July 2019

[14] Jacob Devlin, Ming-Wei Chang, "Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing", Google AI Blog, 2 November 2018

[15] OpenAI, "Better Language Models and Their Implications", 14 February 2019

[16] Luke Salamone, https://lukesalamone.github.io/posts/bert-vs-gpt2/, "BERT vs GPT-2 Performance", 21 June 2021

[17] Radford, Alec and Wu, Jeff and Child, Rewon and Luan, David and Amodei, Dario and Sutskever, Ilya, "Language Models are Unsupervised Multitask Learners", Hugging Face, 2019

[18] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, Shuzi Niu, "DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset", IJCNLP, 11 October 2017

[19] Hannah Rashkin, Eric Michael Smith, Margaret Li, Y-Lan Boureau, “Towards Empathetic Open-domain Conversation Models: a New Benchmark and Dataset”, ACL, 28 August 2019

[20] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, Jason Weston, “Personalizing Dialogue Agents: I have a dog, do you have pets too?”, NASA ADS, 25 September 2018

[21] Eric Michael Smith, Mary Williamson, Kurt Shuster, Jason Weston, Y-Lan Boureau, “Can You Put it All Together: Evaluating Conversational Agents' Ability to Blend Skills”, ACL, 17 April 2020

[22] Thomas Wolf, https://gist.github.com/thomwolf/1a5a29f6962089e871b94cbd09daf317, Hugging Face, 4 May 2019

[23] Tarek Naous, Wissam Antoun, Reem A. Mahmoud, Hazem Hajj, “Empathetic BERT2BERT Conversational Model: Learning Arabic Language Generation with Little Data”, Sixth Arabic Natural Language Processing Workshop, 19 April 2021

[24] AUB MIND, https://github.com/aub-mind/Arabic-Empathetic-Chatbot, 23 January 2022

[25] Ahmed Soror, “Open-domain Conversation chatbot in Arabic”, German University in Egypt, 23 June 2021

Appendix A: Development Platforms and Tools

In this appendix we will show the hardware platforms and the software tools that we used to implement in our project

A.1. Hardware Platforms

We trained our models in using the hardware of

- **Google Colab Pro**
- **Google Cloud Platform**

A.2. Software Tools

A.2.1. Programming languages

- **Python**: it is used for building models and services.
- **Javascript**: used in building the mobile UI.

A.2.2. Libraries and Frameworks

- **sklearn**: used to build and train models.
- **numpy**: used for better data manipulation in python.
- **pandas**: used for better data representation.
- **Flask**: used in building the backend services.
- **React Native**: used in building the UI.
- **Google Places API**: used for places and location retrieval.
- **MongoDB**: a nosql database used to store the profile and messages of user
- **Tensor Flow and PyTorch:** used for the deep learning models training

A.2.3. Platforms and Tools

- **GitHub**: code version control.
- **Visual Studio Code**: code editor.

Appendix B: Use Cases

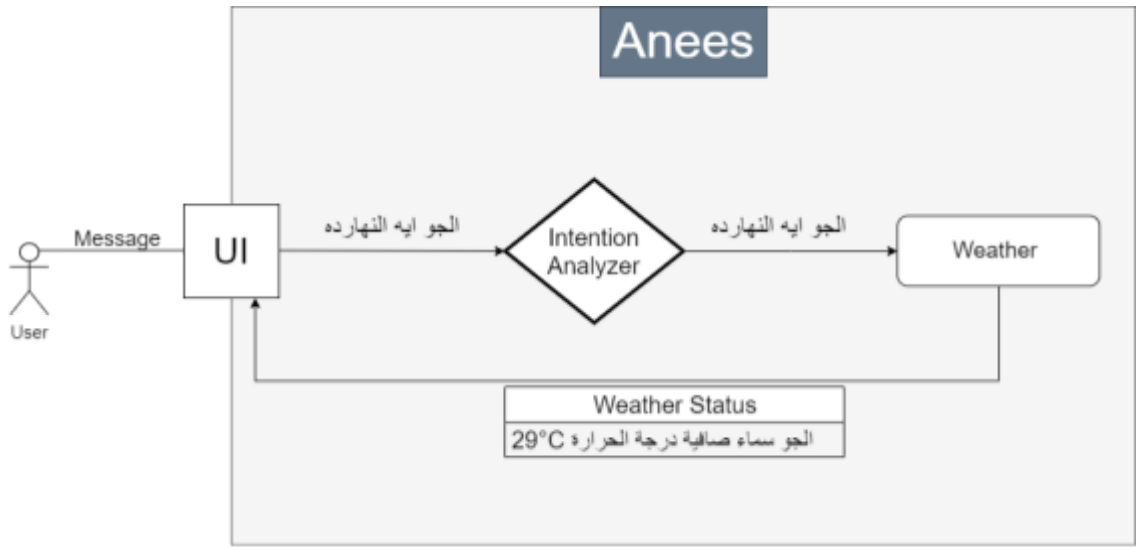


Figure B.1: Weather module use case

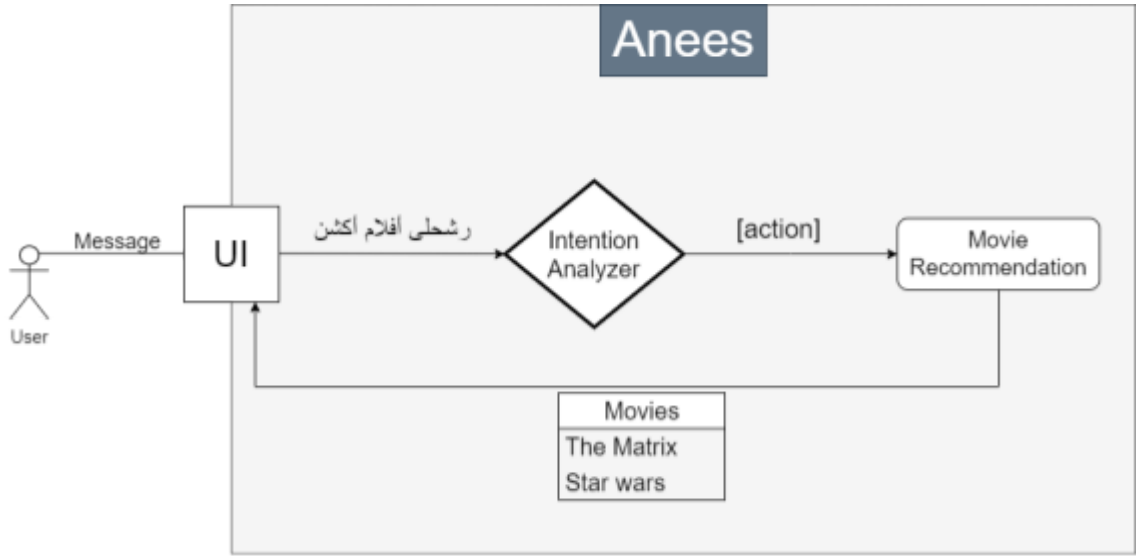


Figure B.2: Movie recommendation use case

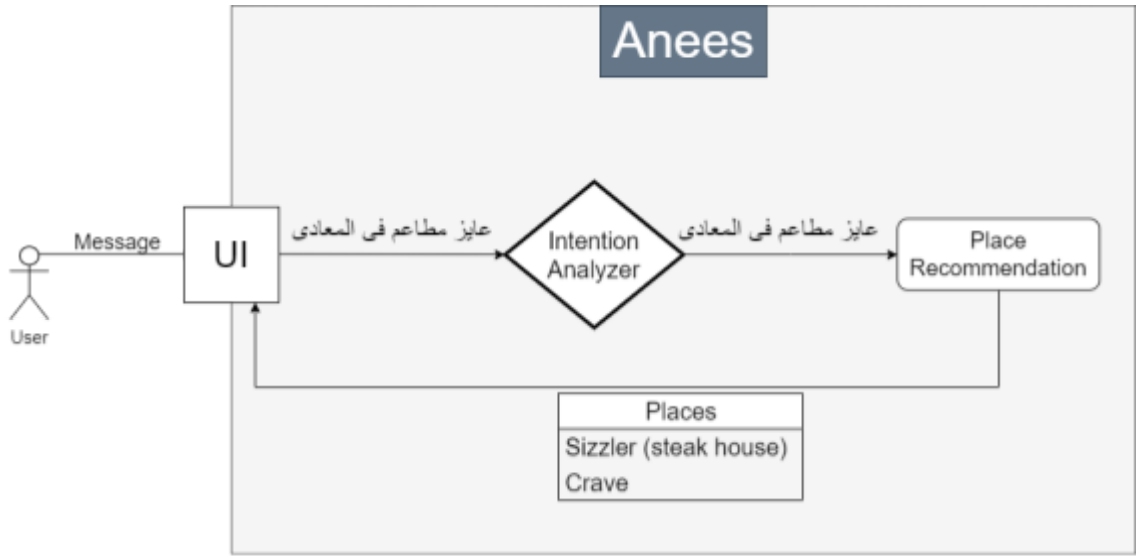


Figure B.3 : PLace recommendation use case

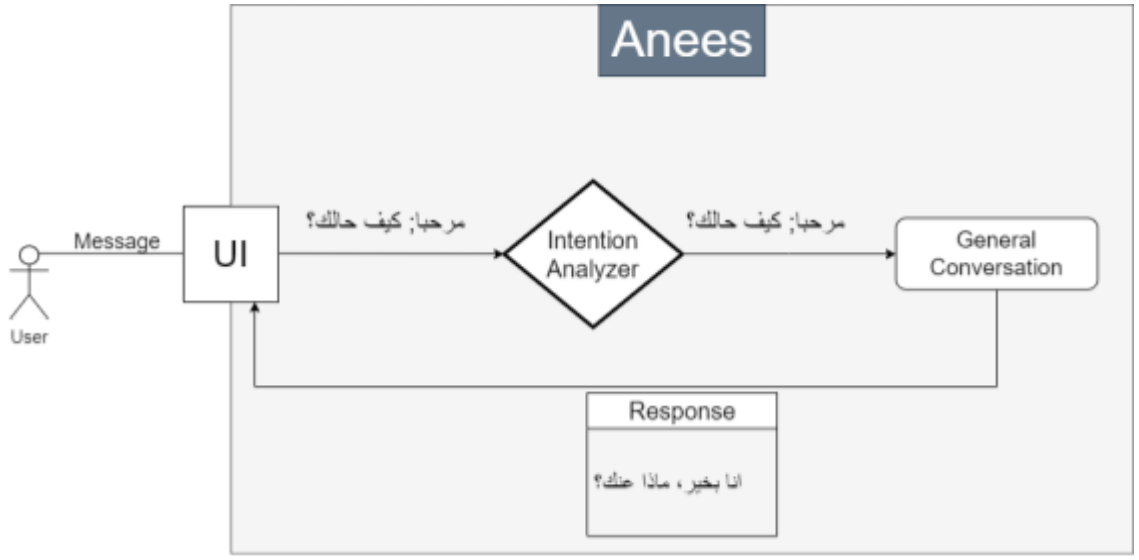


Figure B.4 : General conversation use case

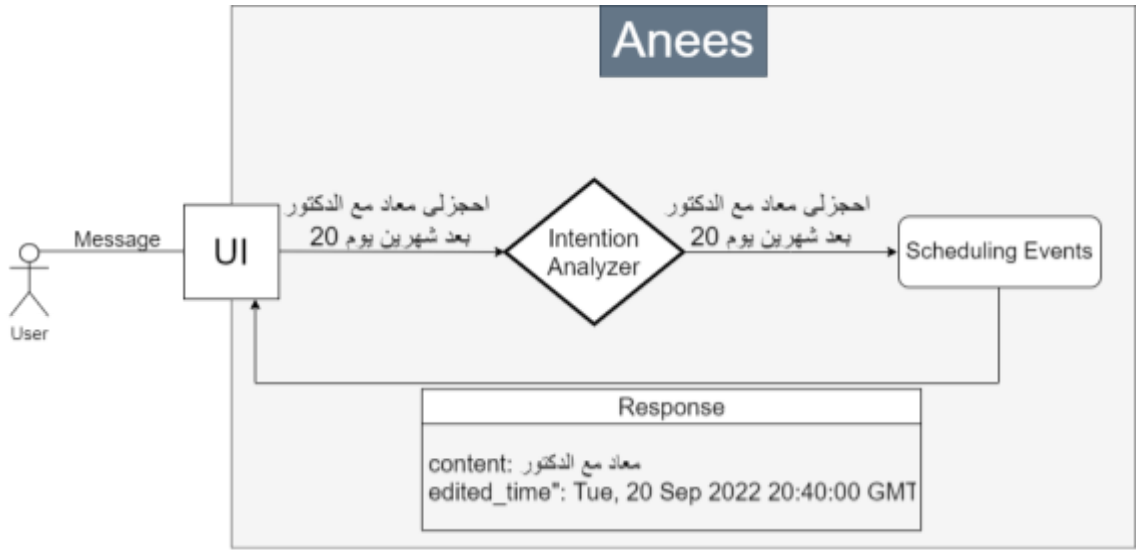


Figure B.5 : Scheduling event use case

Appendix C: User Guide

Our app is easy to use. the main page in Figure C.1 is the login page where user logins in by pressing "يلا نتكلم"if has an account if not click on "لسه جديد ؟ اعمل اكونت جديد" to sign up.

Figure C.1: Anees Login Page

Inside the chat you can talk with Anees either in general or to do specific task, the figure shows a conversation with Anees in general

Figure C.2: Anees Chat Screen

There are 5 tasks you can let Anees do for you. The first one is to get weather:

Figure C.3: Anees Weather example

The second one is to schedule a meeting/appointment:

Figure C.4: Anees event schedule example

The third one is to search:

Figure C.5: Anees searching online example

The fourth one is to ask for recommendation for location:

Figure C.6: Anees place recommendation example

The last on is the movies recommendation:

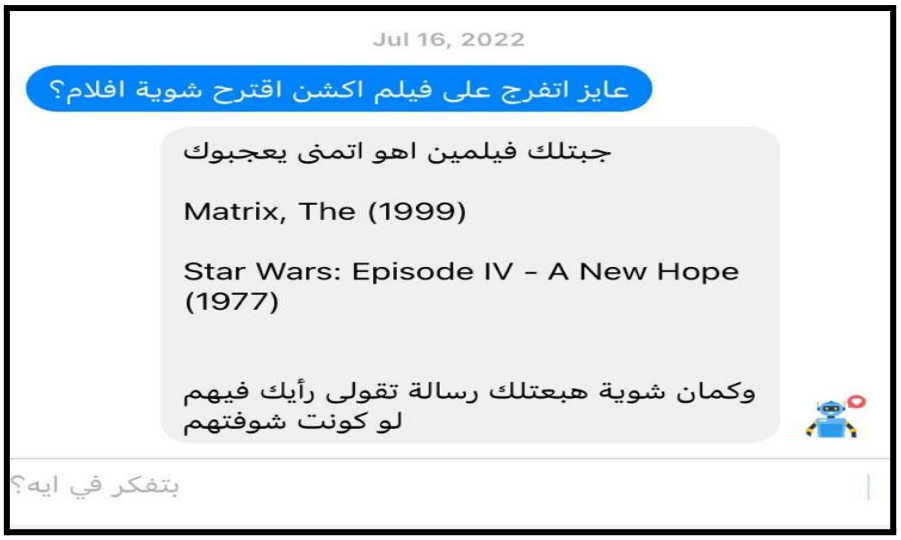


Figure C.7: Anees movie recommendation example

After Anees recommends movies by some time, a notification will be sent to ask the user rate the movies that were recommended.

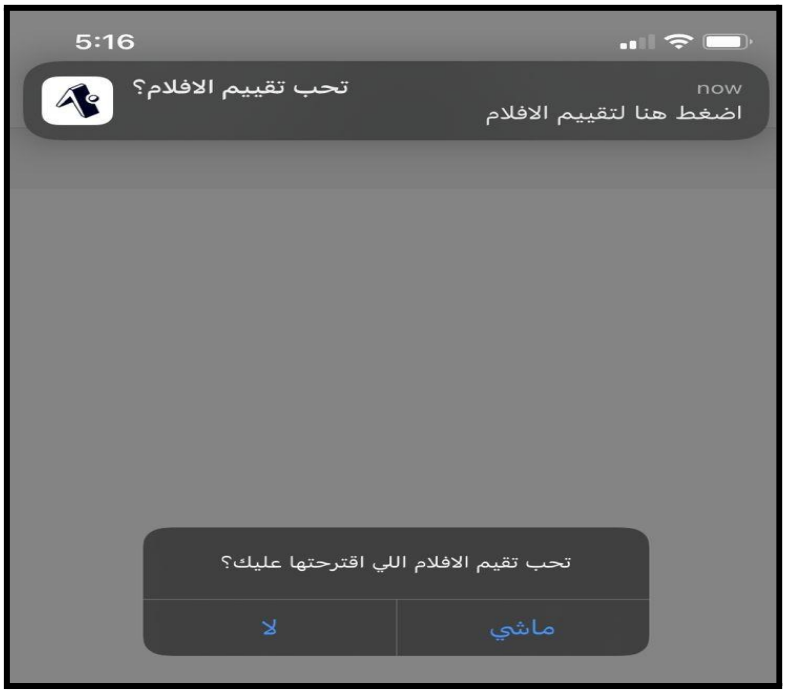


Figure C.8: Anees movie rating notification

If the user chose to rate then a rating page will be loaded to take a rate from the user out of 5:

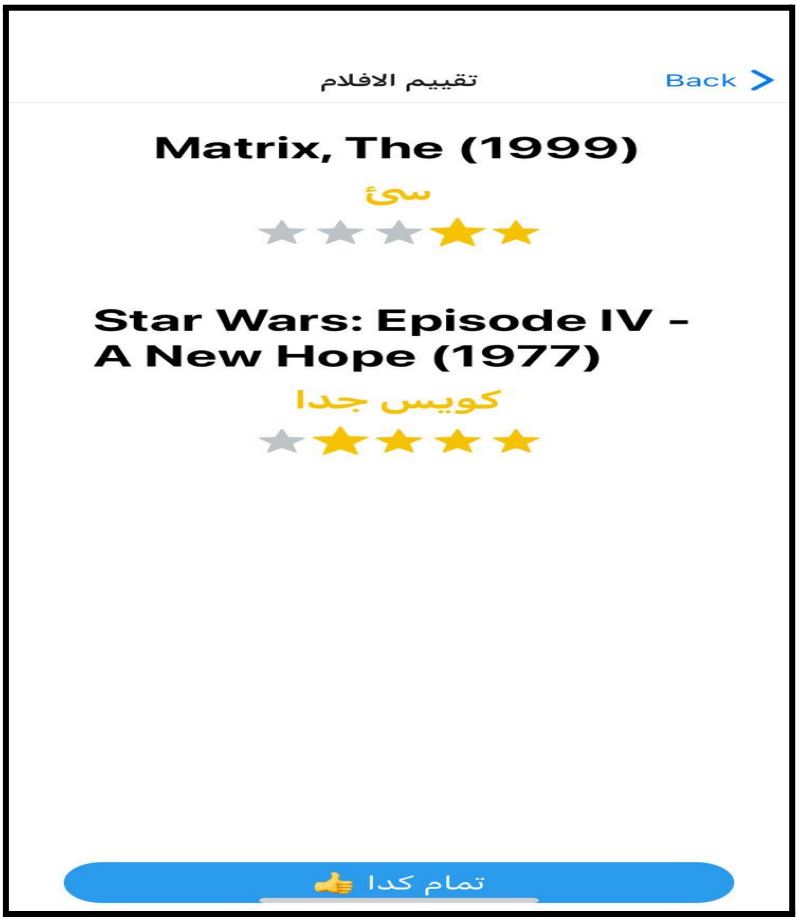


Figure C.9: Anees movie rating screen

Then to logout just type “سلام” in chat:

Figure C.10: Anees Goodbye message

Appendix D: Code Documentation

NLU(text, stopwords, ner_instance, verbs, nouns)

Applies the NLU module

Parameters

- text: input text written by user
- stopwords: list of stopwords
- ner_instance: model of NER
- verbs: list of verbs (used in stemming and POS)
- nouns: list of nouns (used in stemming and POS)

Returned values

- text: text after applying preprocessing
- tokens: list of tokens after removing stopwords
- ents: NER dictionary (person . location , organization)
- token_verb_noun: dictionary of POS + NER

time_extract.main(tokens,token_verb_noun)

Extracts time from the text
Parameters

- tokens: list of tokens returned from NLU function
- token_verb_noun: dictionary of POS + NERreturned from NLU function

Returned values

- edited_time : the time that is extracted from the sentence.
- tokens_used: the tokens used in extracting the time in order not to use them in the content extraction.
- filtered_tokens: the tokens after removing the tokens used in time extract

content_extract.get_schedule_content(text, tokens_used, filtered_tokens)

Extract content to be saved in Calendar

Parameters

- text: preprocessed text returned from NLU function
- token_used: returned from time_extract.main function
- filtered_tokens

Returned values
- content: title to be saved in calendar

movie_recomm.recommend_given_categories(categories, relevance, top_k)

Get movies to be recommended

Parameters
- categories: list of movies' categories
- relevance: list of relevance of each category
- top_k: number of movies to be returned

Returned values
- movies: dataframe of movies

movie_recomm.get_movie_id(movie)[0]

Get movie id to be used in category

Parameters
- movie: title of movie

Returned values
- movie_id: id of movie

movie_recomm.general_recommendation(movie_id)

Get movies to be recommended
Parameters
- movie_id: id of movie returned from grt_movie_id function

Returned values
- movies: dataframe of movies

Appendix E: Feasibility Study

E.1 Technical Feasibility

The project is technically possible as we have done it and also we can add more features as discussed before and there are many reasons for that as :

- There is much research in the field of NLP.
- There are many applications like Anees with high performance in English language.

E.2 Economic Feasibility

The project can be afforded as creating the project did not cost much except for getting a server to run some deep learning models. Also it will increase the profit as discussed in chapter 2.

E.3 Legal Feasibility

Our project is legally feasible for many reasons as:

- It depends on publicly available libraries and free software.
- All datasets used are public , free and also do not require a license

E.4 SWOT Analysi

- Strengths
  - Good accuracy
  - User-friendly.
  - support user emotions
- Weaknesses
  - Need more data for training
  - not good at egyptian dialect
- Opportunities
  - Adding extra features
  - improve existing models
  - Adding new data
- Threats
  - Strong competitors as Google assistant is extraordinary
  - Lack of Egyptian conversational datasets.