Endsem Report

On

Conversational AI Models

Submitted by:-

HARSH PANDEY **2019B3A70489P**

Abhijith Kannan **2019B5A70688P**

Submitted to:-

Dr. Yashvardhan Sharma

Associate Professor,

Computer Science and Information Systems Department

BITS-Pilani, Pilani Campus

Birla Institute of Technology and Science, Pilani (Rajasthan)- 333031

December, 2022

INDEX

Project 1: Sarawati Temple FAQ Chatbot

BACKGROUND

What is conversational AI?

Conversational artificial intelligence (AI) refers to tools that allow users to communicate with virtual assistants or chatbots. They act like people by recognizing voice and text inputs and translating them into other languages. They do this by using a lot of data, machine learning, and natural language processing. Despite its limited scope, conversational AI is a very beneficial technology for businesses, increasing their profitability. Even though AI chatbots are the most common type of conversational AI, there are still a lot of other ways to use AI in the business world. Several instances include: online customer support, accessibility, HR processes, health care, Internet of Things (IoT) devices, and computer software.

While the majority of AI chatbots and applications still only have basic problem-solving capabilities, they can speed up and save money on routine customer care exchanges, freeing up staff time for more complex client engagements. Conversational AI applications that can successfully imitate human conversations have made customers happier.

How are chatbots made?

NLP and machine learning make up the majority of the technology that powers chatbots.

When a user asks a question to a chatbot, a number of intricate algorithms analyze the input they get, decipher what the user is asking, and then decide on an appropriate response.

Chatbots have to rely on algorithms that can figure out the complexities of both spoken and written words. Some chatbots operate so effectively that it is hard to tell whether a user is a computer or a person. But it may be hard for robots to understand complicated conversations where there are a lot of figures of speech.

Motivation to use chatbots in businesses

1. Time-saving

All of your customers should get excellent customer service. But some clients will have easier questions. A chatbot or virtual assistant can handle everyone's demands without overburdening you and your staff.
AI chatbots can handle simple customer support problems while you and your team handle more complicated ones. It reduces wait times on both ends.

2. Improved accessibility

You cannot serve consumers 24/7. Conversational AI on social media solves this. Chatbots may assist customers after hours. It addresses a logistic issue and demonstrates how chatbots save time, but there's more.
Due to its accessibility, conversational AI might make your consumers feel more at ease. Midnight may be the only moment someone has to resolve a problem.

3. Assist buyers

Conversational AI can resolve customer support tickets. It may also boost sales.
Machine learning personalizes client experiences. A conversational AI platform could suggest products or add-ons to customers that they may not have seen or thought of.

4. Sell off-hours

Conversational AI's accessibility helps clients make purchases. Online businesses benefit from 24/7 sales. Customers' shipping, sales, and product questions while reps aren't accessible may only hinder that.
Chatbots or virtual assistants solve this rapidly. It may help anybody who needs a question addressed before checking out, since it's open 24/7. It speeds up sales and reduces the chance of buyers abandoning their purchases.

5. Language obstacles gone

Conversational AI reduces language boundaries. Most chatbots and VAs include language translation software. They can detect, comprehend, and produce most languages.
No language barrier hinders customer service interactions. A multilingual chatbot opens your company to more clients.

OBJECTIVE

to design a chatbot for answering the FAQs related to the Saraswathy Mandir. This chatbot will also be available in all common languages. It will be a FAQ chatbot for Saraswathy Mandir that can answer questions in more than one language.

Proposed Techniques and Algorithms

Some of the top models for creating a multilingual chatbot we considered were:
BERT:
BERT uses transformers, which are ways to pay attention that learn how words (or subwords) in a text fit together in their context. The basic design of Transformer is made up of two separate parts: an encoder that reads the text input and a decoder that predicts a job. Since BERT's goal is to make a language model, it only needs the encoder mechanism. In a publication published by Google, the precise operation of the transformer is disclosed.

The Transformer encoder reads the full sequence of words at once, in contrast to directional models, which read the text input sequentially (from right to left or left to right). Even though it would be better to say that it doesn't go anywhere, it is thought of as bidirectional. This trait allows the model to understand the context of a word based on its surroundings (to the left and right of the word).

GPT-3:
GPT-3 is the third iteration of OpenAI's GPT language models. The size of the GPT-3 is what distinguishes it from earlier variants. GPT-3 is 17 times larger than GPT-2 and around 10 times larger than Microsoft's Turing NLG model with 175 billion parameters. In the same way that I talked in my first essay, GPT-3 has 96 attention blocks, and each attention block has 96 attention heads. In other terms, GPT-3 is essentially a scale replica of a transformer.

GPT-3 was trained with a mix of the following huge text datasets, all of which were taken from the original publication that presented this model:

- Wikipedia Corpus
- Common Crawl
- WebText2
- Books1
- Books2

The final set of data included all of Wikipedia, a large number of web pages, and a huge number of books. Researchers trained GPT-3 to produce text in English in a number of different languages using this dataset, which has hundreds of billions of words.

RASA:
Rasa is a program that uses Python and NLU to create personalized AI chatbots (NLU). Rasa (NLU) gives you a framework for making AI chatbots that can understand natural language. The

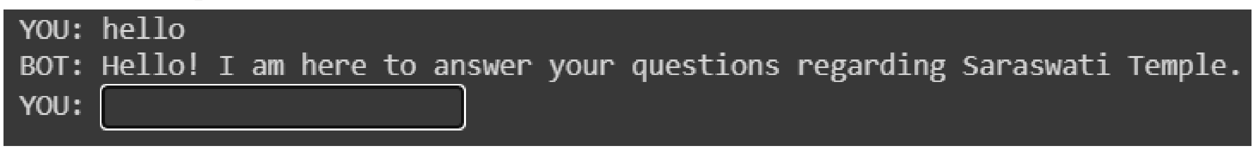
user may also add custom actions and train the mod. Rasa chatbots have been installed on Slack, Microsoft Bot, and Facebook Messenger, among other places.

Rasa is composed of two primary parts:

Rasa NLU is an open-source natural language processing solution for intent categorization (figuring out what the user wants), entity extraction from the bot in the form of structured data, and helping the chatbot understand what the user is saying.
Rasa Core is a chatbot framework that uses machine learning to manage conversations. Instead of if/else statements, it uses structured information from the NLU to predict the best thing to do. Under the hood, it also makes use of reinforcement learning to enhance action prediction.
In other words, Rasa NLU's job is to read the user's structured data, and Rasa Core's job is to figure out what the chatbot should do next. Rasa Core and Rasa NLU may be utilized individually and are not dependent on one another.

EXPERIMENTS AND RESULTS

Our chatbot is set up to answer all the most common questions about Saraswathy Mandir on the Bits Pilani campus.



Our model is able to classify the user input into 7 major intents/classes

- **Location of the temple**
- **Timing of the temple**
- **Significance of the temple**
- **How to reach the temple**
- **When was the temple built**
- **What is the entry cost of the temple**
- **What are the other attractions near the temple**
- **Basic intents like greet,goodbye,check for bot etc**

Based on what the user says, the model can figure out the user's intent and put it into one of the above categories. For each intent, we have programmed the chatbot to respond using a specific response. We rewrote the answer in different ways and mixed up the output to make the chatbot more interesting and interactive.

By layering the Google Translate API on the input and output sides, we are able to make this bot support almost all of the spoken languages. For demonstration purposes, we've also added a layer that recognizes the input and language and sends back output in English. Through this, we have made our chatbot multilingual.

Explanation below: The chatbot processes our input and returns the most appropriate response.

- **Intent extraction**

```
# Adding the NLU components to the pipeline in config.yml file
config = """
language: "en_core_web_md"
pipeline:
- name: "SpacyNLP"                  # loads the spacy language model
- name: "SpacyTokenizer"            # splits the sentence into tokens
- name: "CRFEntityExtractor"        # uses the pretrained spacy NER model
- name: "SpacyFeaturizer"     # transform the sentence into a vector representation
- name: "SklearnIntentClassifier"   # uses the vector representation to classify using SVM
- name: "EntitySynonymMapper"               # trains the synonyms
policies:
- name: MemoizationPolicy
- name: TEDPolicy
  max_history: 5
  epochs: 10
# a confidence >= core_fallback_threshold
core_fallback_threshold: 0.68                      # Confidence threshold for the `core_fallback_action_name` to apply.
core_fallback_action_name: "action_default_fallback"     # The action will apply if Loading... ction was predicted with
enable_fallback_prediction: True                   # a confidence >= core_fallback_threshold
"""
```

  - The model understands in input using the **SpacyNLP** language model. The synonyms are also learnt from the input using **EntitySynonymMapper.**
  - Then the text is converted into tokens using tokenizers. (**SpacyTokenizer** in our case)
  - Now our model does feature extraction on the tokens. This is done using **CRFEntityExtractor** and **Space Featurizer**
  - Now our model uses the extracted features to classify the user input into the different available intents. We use **SklearnIntentClassifier** for this task.

- **Policies used for responding**
  RASA uses many policies to decide which action to take at each step in a conversation. There are machine-learning and rule-based policies that our chatbot uses in tandem.
  - MemoizationPolicy : the model remembers the story from the training data. When the current conversation matches the stories from stories. yml the model predicts the next action from the matching stories with a confidence of 1.0
  - TED (transformer embedding dialogue policy) a multi-takk architecture for next action prediction and entity recognition. The architecture consists of several transformer encoders which are share for both tasks.

- ○ RulePolicy : handles conversation parts that follow a fixed behavior (like business logic). It makes predictions based on some rules that are present in the training data.

- **Giving Response**

```
utter_info_timings:
- text: "The temple is open everyday from morning 8am to evening 7pm (except for special holidays). "

utter_info_history:
- text: "The temple was suggested by SD Pande, who on his trips to the \n
         universities abroad saw the students going to the church to seek solace.\n
         He gave the idea  to construct a temple for the Goddess of knowledge\n
         on the campus to the trustees who  readily acknowledged it. \n
         Thus the construction began in 1956. It was inaugurated four years later in 1960."

utter_significance:
- text: "The temple is dedicated to Saraswati (the divinity of learning) with \n
         engravings of every kind of major deities and personalities on its outside. \n
         It is constructed of beautiful marble, which was specially brought \n
         from Makrana, also in Rajasthan. Incorporating elements of Indo-Aryan \n
         scheme of temple construction, the temple design is inspired by the \n
         Khandariya Mahadeo Temple at Khajuraho."
```

- ○ The model uses the expected intent and gives the result as one of the programmed responses from the domain. yml file
- ○ Multiple paraphrased responses are given for each intent and the model chooses from one of them to respond.
- ○ This keeps the model more dynamic and makes user interaction more interesting.

Demonstration

In the way shown for the English language, the model gives results that are very accurate.

```
YOU: hello
BOT: Hello! I am here to answer your questions regarding Saraswati Temple.
YOU: where is the temple
BOT: The temple is located inside Bits Pilani premises infront of T-lawns.
YOU: what are nearby attractions near temple
BOT: Birla Museum, Student Activity Centre, Sky Lawns, and the Clock Tower.
YOU: when is the temple open
BOT: The temple is open everyday from morning 8am to evening 7pm (except for special
YOU: what is the price of the ticket to temple
BOT: It is absolutely free to visit.
YOU: how to reach the temple
BOT: Pilani can be reached either by road or by rail. Nearest railway stations
 are Chirawa on Western railway (16 Km) and Loharu on Northern railway (24 Km).
 There are regular bus services between Delhi and Pilani and Jaipur and Pilani.
 The buses leave Delhi from Inter- State Bus Terminal (ISBT) and Kashmiri gate.
 From Jaipur buses leave from Rajasthan State Roadways bus stand, Sindhi camp.
 Campus is very close to Pilani bus stand.
YOU: what other activities can I do near the temple?
BOT: The temple is open everyday from morning 8am to evening 7pm (except for special
```

A demonstration of the model supporting multilingual inputs is shown below. In the given example, the model is given inputs in the “Hindi” and “Korean" languages, respectively, according to the configuration file.

WORK UPDATE

WORK COMPLETED

- Explored various models for implementing conversational AIs.
- Created a FAQ chatbot for BITS Pilani’s Saraswati Temple.

WORK REMAINING

- Implement the model in respective languages instead of using translate api for a common intermediate language.
- Implement the fallback policy in rasa, which gives a 'I am not sure' response when the confidence of any of in intent is not high enough.
- Automatically generate the stories and intent from the training data

Project 2: Arabic Chatbot

The task was to create an Arabic chatbot. The company C-Zentrics had an immediate problem at hand, i.e., to design an Arabic chatbot that is trained on the Arabic language dataset itself. They had shown their chatbot in Middle Eastern markets, and the feedback was that they needed to train the whole chatbot in the Arabic dataset because translating to an intermediate language changed how accurate the results were. The majority of people in the area who call or chat with customer service prefer Arabic to English.

So, this project looks at how to train the chatbot model in Arabic in the best way, how to test and analyze the Arabic bot model, and how to deal with the different kinds of Arabic.

Proposed Techniques and Algorithms

We used transformer-based models for the project since they have pretrained models in Arabic. There are several advantages to using transformer models for chatbots:

1. Ability to handle long-range dependencies: Chatbots often need to understand the context of a conversation and make decisions based on that context. Transformer

models are able to capture long-range dependencies in the input data, which makes them well-suited for handling such context-dependent tasks.

2. Efficient parallelization: As was already said, transformer models can run the computations for the self-attention layers in parallel, which makes it easy to train them on large datasets. This can be useful for chatbots, which may need to be trained on a large dataset of conversational examples.

3. Strong performance on natural language tasks: Transformer models have achieved state-of-the-art results on a variety of natural language processing tasks, including machine translation and language modeling. This suggests that they may be effective at handling the complexities of natural language in chatbot conversations.

4. Ability to incorporate additional information: Transformer models can be augmented with additional input features, such as user profile information or external knowledge, which can be useful for personalized chatbots or those that need to access external knowledge sources.

5. Easy to fine-tune: Pre-trained transformer models, such as BERT and GPT, have become widely available and can be fine-tuned for specific tasks with relatively little data and computation. This makes it relatively easy to leverage the power of transformer models for chatbot applications.

6. Versatility: Transformer models can be used for a variety of chatbot tasks, including generating responses to user inputs, classifying user inputs into predefined categories, and generating natural language summaries of input data.

7. Robustness to input variations: Transformer models are generally robust to variations in the input data, such as misspellings or variations in word order. This can be useful for chatbots, which may need to handle a wide range of user inputs.

Some of the models we implemented are:

AraBert-Arabic-SQuADv2-QA

Arabert is a pre-trained language model for the Arabic language. It is based on the BERT (Bidirectional Encoder Representations from Transformers) model, which is a transformer-based architecture for natural language processing tasks.

Researchers from the Qatar Computing Research Institute (QCRI) and the Allen Institute for Artificial Intelligence (AI2) worked together to make Arabert. It was trained on a large set of Arabic text and is designed to be fine-tuned for many NLP tasks, such as language modeling, question answering, and text classification.

Arabert can be used to pull out feature representations from Arabic text, just like other pre-trained language models. These representations can then be used as input for a task further down the line. This can be useful for building chatbots or other NLP systems for the Arabic language.

```
from transformers import ElectraForQuestionAnswering, ElectraForSequenceClassification, AutoTokenizer, pipeline
#from preprocess import ArabertPreprocessor
#import ArabertPreprocessor
#prep_object = ArabertPreprocessor("araelectra-base-discriminator")
arabert_prep = ArabertPreprocessor("aubmindlab/bert-base-arabertv2")
question= arabert_prep.preprocess("ما هي جامعة الدول العربية ؟")
#question = prep_object('ما هي جامعة الدول العربية ؟')
context = arabert_prep.preprocess('''
جامعة الدول العربية هيمنظمة إقليمية تضم دولاً عربية في آسيا وأفريقيا.
لاقات الثقافية، الجنسيات ووثائق وأذونات السفر والعلاقات الاجتماعية والصحة. المقر الدائم لجامعة الدول العربية يقع في القاهرة، عاصمة مصر (تونس من 1979 إلى 1990)
''')
# a) Get predictions
qa_modelname = 'ZeyadAhmed/AraElectra-Arabic-SQuADv2-QA'
cls_modelname = 'ZeyadAhmed/AraElectra-Arabic-SQuADv2-CLS'
qa_pipe = pipeline('question-answering', model=qa_modelname, tokenizer=qa_modelname)
QA_input = {
    'question': question,
    'context': context
}
CLS_input = {
    'text': question,
    'text_pair': context
}
qa_res = qa_pipe(QA_input)

#removing this line temporarily
#cls_res = cls_pipe(CLS_iput)

threshold = 0.5 #hyperparameter can be tweaked
## note classification results label0 probability it can be answered label1 probability can't be answered
## if label1 probability > threshold then consider the output of qa_res is empty string else take the qa_res
# b) Load model & tokenizer
qa_model = ElectraForQuestionAnswering.from_pretrained(qa_modelname)
cls_model = ElectraForSequenceClassification.from_pretrained(cls_modelname)
tokenizer = AutoTokenizer.from_pretrained(qa_modelname)
```

```
/usr/local/lib/python3.8/dist-packages/urllib3/connectionpool.py:842: InsecureRequestWarning: Unverified HTTPS req
```

```
print(qa_res["answer"])
```

```
آسيا و+ أفريقيا
```

Araelectra-base-discriminator

ELECTRA is a method for self-supervised language representation learning. It can be used to pre-train transformer networks with minimal computation. Like the discriminator of a GAN, ELECTRA models are trained to tell the difference between "real" input tokens and "fake" input tokens made by another neural network. AraELECTRA achieves state-of-the-art results on the Arabic QA dataset.

```
from arabert.preprocess import ArabertPreprocessor
from transformers import pipeline

prep = ArabertPreprocessor("aubmindlab/araelectra-base-discriminator") #or empty string it's the same
qa_pipe =pipeline("question-answering",model="wissamantoun/araelectra-base-artydiqa")

text = " ما هو نظام الحكم في لبنان؟"
context = """
العربيّة هناك وجود فعّال للمسيحيين في الحياة العامّة والسياسيّة. هاجر وانتشر أبناؤه حول العالم منذ أيام الفينيقيين، وحاليًّا فإن عدد اللبنانيين المهاجرين يُقدَّر بضعف عدد اللبنانيين المقيمين
روب والنزاعات على مر العصور تجلت بحروب أهلية ونزاع مصيري مع إسرائيل. ويعود أقدم دليل على استيطان الإنسان في لبنان ونشوء حضارة على أرضه إلى أكثر من 7000 سنة
بنان عدّة حضارات وشعوب استقرت فيه منذ عهد الفينيقين، مثل المصريين القدماء، الآشوريين، الفرس، الإغريق، الرومان، الروم البيزنطيين، العرب، الصليبيين، الأتراك العثمانيين، فالفرنسيين
"""

context = prep.preprocess(context)# don't forget to preprocess the question and the context to get the optimal results
result = qa_pipe(question=text,context=context)
# """
# {'answer': 'ديمقراطي جمهوري طوائفي',
#  'end': 241,
#  'score': 0.4910127818584442,
#  'start': 219}
# """
```

```
print(result['answer'])
```

```
ديمقراطي جمهوري طوائفي
```

bert_qa_arap_qa_bert_large_v2

```
document_assembler = MultiDocumentAssembler().\
setInputCols(["question", "context"]).\
setOutputCols(["document_question", "document_context"])

spanClassifier = BertForQuestionAnswering.\
pretrained("bert_qa_arap_qa_bert_large_v2","ar").\
setInputCols(["document_question", "document_context"]).\
setOutputCol("answer").setCaseSensitive(True)

pipeline = Pipeline().setStages([document_assembler,spanClassifier])

example = spark.createDataFrame([["أين يعيش محمد ؟", "اسمي محمد وأنا أعيش في سوريا"]]).\
toDF("question", "context")

result = pipeline.fit(example).transform(example)
```

```
[ ] print(result.select("answer").first()[0][0]["result"])
```

```
سوريا
```

Bert_qa_arap_qa_bert_v2

```
document_assembler = MultiDocumentAssembler().\
setInputCols(["question", "context"]).\
setOutputCols(["document_question", "document_context"])

spanClassifier = BertForQuestionAnswering.\
pretrained("bert_qa_arap_qa_bert_v2","ar").\
setInputCols(["document_question", "document_context"]).\
setOutputCol("answer").setCaseSensitive(True)

pipeline = Pipeline().setStages([document_assembler,spanClassifier])

example = spark.createDataFrame([["أين يعيش محمد ؟", "اسمي محمد وأنا أعيش في سوريا"]]).\
toDF("question", "context")

result = pipeline.fit(example).transform(example)
```

```
print(result.first()[4][0].result)
```

```
سوريا
```

Bert_qa_arap_qa_bert

```
document_assembler = MultiDocumentAssembler().\
setInputCols(["question", "context"]).\
setOutputCols(["document_question", "document_context"])

spanClassifier = BertForQuestionAnswering.\
pretrained("bert_qa_arap_qa_bert","ar").\
setInputCols(["document_question", "document_context"]).\
setOutputCol("answer").setCaseSensitive(True)

pipeline = Pipeline().setStages([document_assembler,spanClassifier])

example = spark.createDataFrame([["أين يعيش محمد ؟", "اسمي محمد وأنا أعيش في سوريا"]]).\
toDF("question", "context")

result = pipeline.fit(example).transform(example)
```

```
print(result.first()[3][0].result)
```

```
اسمي محمد وأنا أعيش في سوريا
```

Dataset used for pre-training

The pretraining data used for the new AraBERT model is also used for AraGPT2 and AraELECTRA.

The dataset consists of 77GB or 200,095,961 lines or 8,655,948,860 words or 82,232,988,358 chars (before applying Farasa Segmentation)
Following are some of the sources for the data: -

- OSCAR unshuffled and filtered.
- Arabic Wikipedia dump from 2020/09/01
- The 1.5B words Arabic Corpus
- The OSIAN Corpus
- Assafir news articles

EXPERIMENTS AND RESULTS

We have analyzed all 5 of the above-mentioned models and compared their performance solely based on their pre-trained knowledge. After comparing their F1-scores, recall, and precision, we came to the conclusion that "bert_qa_arap_qa_bert_large_v2" is performing the best.
The performance rank might differ after fine-tuning since other models were very close behind in their performance; thus, we will be fine-tuning all of the above models.

Future work

- The models should be fine-tuned with the recently provided dataset.
- The dataset provided in English should be converted into Arabic language using Google translate API and this dataset should be fed into our model.
- The fine-tuned models should be analyzed for accuracy and the best model should be implemented for user interaction.
- We can also implement a dedicated UI for the model thus improving the user interaction experience.

Codebase Links

Mandir chatbot

In the English language:
https://colab.research.google.com/drive/1pTuJysI3IXaeKJHxNDULzCoCuE2CE-7b#scrollTo=PyQVtGjWUxdK
Multi-lingual chatbot:
https://colab.research.google.com/drive/1UBQ4IjMReBp0jJIKuA962XBRltV_P1Nw#scrollTo=Z988OF_zQprn

Transformer chatbot Implemented in Arabic language
Bert_qa_arap_qa_bert_large_v2 , "bert_qa_arap_qa_bert_v2" and "bert_qa_arap_qa_bert"
https://colab.research.google.com/drive/1RoHVy60oZdSny_t9Ry3KdgWKYY09yyHy#scrollTo=77iTtMYeBw4F
Araelectra-base-discriminator
https://colab.research.google.com/drive/1GVgmuY0aHUIU1YpiaQyoentB47aYjwEf#scrollTo=5ld9yVgWAQin
AraElectra-Arabic-SQuADv2-QA
https://colab.research.google.com/drive/10uRB_TOCHFp8B2DVx-OWG-ZrahatWVk_#scrollTo=dsiIzH6m1O8_

Presentation Links

Midsem Presentation :
https://www.canva.com/design/DAFSArQtOS0/YGxNhas-oWhdLWNXPWrB-A/view?utm_content=DAFSArQtOS0&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton
Comphre Presentation:
https://www.canva.com/design/DAFUpzbblcA/SFb3f1s0fYKr-g0u61ghCQ/view?utm_content=DAFUpzbblcA&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

REFERENCES

- (2017) Rasa: Open Source Language Understanding and Dialogue Management (Tom Bocklisch, Joey Faulkner, Nick Pawlowski, and Alan Nichol).) https://arxiv.org/abs/1712.05181
- (2020) An Analytical Study and Review of open source Chatbot framework, Rasa (**RakeshKumarSharma**)https://www.researchgate.net/publication/342537790_An_Analytical_Study_and_Review_of_open_source_Chatbot_framework_Rasa
- (2020) An Intelligent Chatbot System Based on Entity Extraction Using RASA NLU and Neural Network (Anran Jiao) https://iopscience.iop.org/article/10.1088/1742-6596/1487/1/012014/meta
- https://towardsdatascience.com/building-a-chatbot-with-rasa-3f03ecc5b324

- https://rasa.com/docs/rasa/
- https://github.com/aub-mind/arabert
- AraELECTRA: Pre-Training Text Discriminators for Arabic Language Understanding https://arxiv.org/abs/2012.15516
- https://huggingface.co/ZeyadAhmed/AraElectra-Arabic-SQuADv2-QA
- AraBERT: Transformer-based Model for Arabic Language Understanding https://arxiv.org/abs/2003.00104
- AraBERT: Transformer-based Model for Arabic Language Understanding https://www.researchgate.net/publication/339642234_AraBERT_Transformer-based_Model_for_Arabic_Language_Understanding