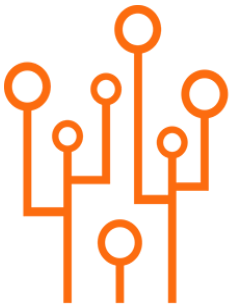


MORINGA

Discover · Grow · Transform

BUILDING A MORINGA SCHOOL CHATBOT USING NATURAL LANGUAGE PROCESSING

Chatbots are computer programs designed to simulate human conversation. They can comprehend and analyze human language, and then respond back to people while accomplishing particular tasks. For our project, we built a text-based chatbot using the FAQs from Moringa School. We specifically chose Moringa School because we noticed a significant lack of effective communication from the administration to the student body, especially during the onboarding process. This caused a lot of confusion and frustration for students and anyone else who had questions or concerns. By creating this chatbot, we hope to provide a more efficient and accessible means for individuals to get the information they need without experiencing the same communication issues that we did.

Our problem statement was as follows:

- Develop a chatbot using natural language processing (NLP) techniques that can efficiently handle and respond to customer queries related to Moringa school and the programs it offers to Kenyans with the goal of improving customer satisfaction.
- The chatbot should be capable of understanding and interpreting natural language inputs related to the common FAQs asked by students enrolling into Moringa school or those currently enrolled.
- It should also provide relevant and accurate responses that align with Moringa schools policies, procedures, and values.
- The project should explore and evaluate different NLP approaches, such as intent recognition, entity recognition and consider the integration of machine learning algorithms for continuous learning and improvement.
- The success of the project will be measured by the accuracy and efficiency of the chatbot's responses, as well as user satisfaction and engagement metrics, such as the number of successful interactions.

Our FAQ chatbot is packed with some pretty cool features! Firstly, the chatbot has natural language understanding, which means it can interpret and understand user queries just like a real person would. This makes for a more conversational and user-friendly experience. Additionally, the chatbot has lightning-fast response times, so you won't be waiting around for answers.

But that's not all! Our chatbot can be integrated into a range of communication channels like websites, social media platforms, and messaging apps, making it super convenient for users. And last but not least, our chatbot provides valuable analytics and insights into user behavior and preferences, which businesses can use to improve their operations and enhance the user experience.

Overall, our FAQ chatbot is a powerful tool that can revolutionize customer support and streamline operations for Moringa School.

METHODOLOGY AND APPROACH

First, we need to gather and organize data about frequently asked questions. To do this, we gathered information from Moringa's FAQ page and additional commonly asked questions by students.

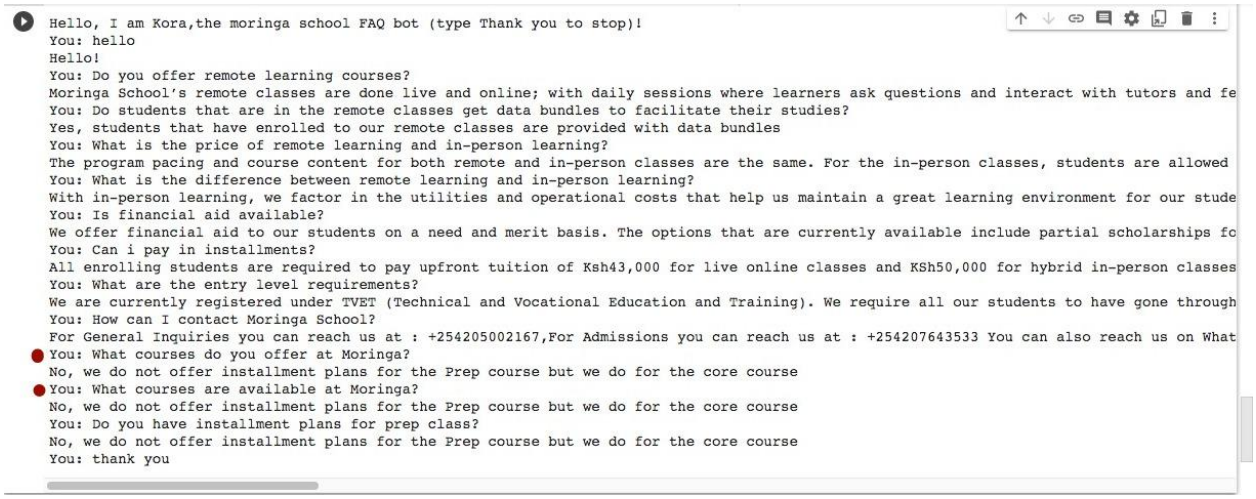
Next, we used natural language understanding (NLU) techniques to extract relevant information from the data. This involved processing the text to identify important details like entities, intents, and context. For this, we used NLU tools like the Natural Language Toolkit (NLTK).

```
!pip install nltk
!pip install tflearn
import numpy
import tflearn
import tensorflow
import random
import json
import nltk
nltk.download('punkt')
from tensorflow.python.framework import ops
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()
```

After the NLU step, we designed the chatbot's conversational flow.Once the chatbot is built, we deployed it to a communication channel. A few popular deployment platforms for chatbots include Facebook Messenger, Slack, and WhatsApp.
We decided to go with Streamlit as our platform of deployment. Throughout the development process, testing and evaluation were critical to make sure the chatbot was accurate, effective, and user-friendly.

CHALLENGES EXPERIENCED

There will always be challenges that come up during the design process. Two of the most common issues we faced were debugging and keyword recognition problems. It can be frustrating when things don't work as expected or when the bot fails to recognise the correct responses to the question asked.



The questions marked in red generated the wrong responses.

However, we found that the key to overcoming these challenges is persistence and problem-solving skills. When debugging, we took a systematic approach to identify the issue, such as reviewing code, checking for syntax errors, or testing different scenarios. For keyword problems, we brainstormed and used the vast tools and resources at our disposal to generate new ideas and alternatives.

While some issues may take more time and effort to resolve than others, we learnt that there is always a solution to be found. By staying curious and resourceful, we were able to push through these challenges and create an effective and functional chatbot that meets the needs of our clients and users.

FUNCTIONALITY

Firstly we imported some necessary libraries.

```
!pip install nltk
!pip install tflearn
import numpy
import tflearn
import tensorflow
import random
import json
import nltk
nltk.download('punkt')
from tensorflow.python.framework import ops
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()
```

Then we downloaded the required NLTK dataset, and loaded a JSON file containing the questions and answers that the chatbot will use to generate responses.

```
[4] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive

[5] import json
    with open('/content/FAQs') as file:

        data = json.load(file)
    print (data)
```

Then, the code processes the text data, tokenizes it into words, and applies stemming to normalize the words. It creates a "bag of words" for each question-answer pair, essentially a list of binary values indicating whether or not each word in the vocabulary appears in the question.

Next, the code defines and trains a neural network model using the processed data to recognize which question a user is asking and provide an appropriate response.

In detail the chatbot uses the trained neural network model to predict which category the input belongs to, based on the bag-of-words representation of the input.The model returns a probability distribution over all categories, and the chatbot selects the one with the highest probability. If the highest probability is above a certain threshold, the chatbot chooses a random response from the set of possible responses for that category and outputs it to the user.If the highest probability is below the threshold, the chatbot responds with a default message indicating that it did not understand the input, and prompts the user to ask another question.

```
[15] training = []
     output = []

     out_empty = [0 for _ in range(len(labels))]

     for x, doc in enumerate(docs_x):
         bag = []

         wrds = [stemmer.stem(w.lower()) for w in doc]

         for w in words:
             if w in wrds:
                 bag.append(1)
             else:
                 bag.append(0)

         output_row = out_empty[:]
         output_row[labels.index(docs_y[x])] = 1

         training.append(bag)
         output.append(output_row)
```

```
[16] training = numpy.array(training)
     output = numpy.array(output)
```

```
✓ 0s [17] ops.reset_default_graph()

     net = tflearn.input_data(shape=[None, len(training[0])])
     net = tflearn.fully_connected(net, 10)
     net = tflearn.fully_connected(net, 10)
     net = tflearn.fully_connected(net, len(output[0]), activation="softmax")
     net = tflearn.regression(net)

     model = tflearn.DNN(net)
```

```
✓ 5m [18] model.fit(training, output, n_epoch=1000, batch_size=5, show_metric=True)

     Training Step: 10999  | total loss: 0.00352 | time: 0.048s
     | Adam | epoch: 1000 | loss: 0.00352 - acc: 1.0000 -- iter: 50/55
     Training Step: 11000  | total loss: 0.00367 | time: 0.052s
     | Adam | epoch: 1000 | loss: 0.00367 - acc: 1.0000 -- iter: 55/55
     --
```

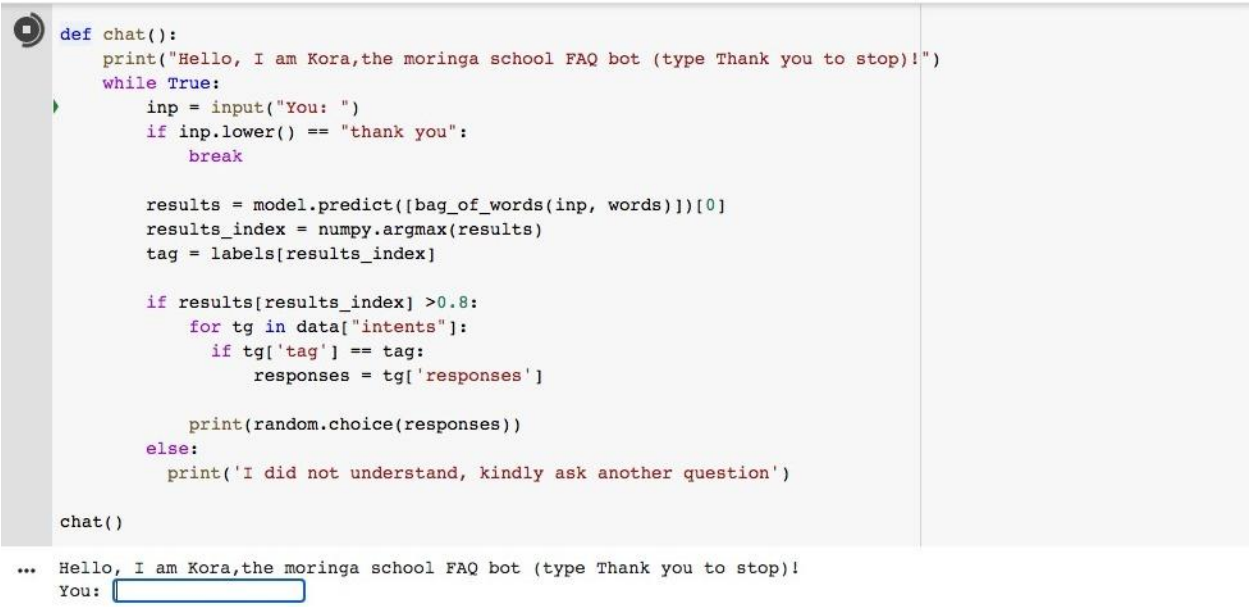
Finally, the code implements a chat function that prompts the user for input and uses the trained model to generate a response based on the input.

The conversation ends when the user types "Thank you".

Overall, the chatbot functions as a simple FAQ system that can answer questions within the domain of the provided dataset. It uses natural language processing and machine learning techniques to simulate a conversation with the user and provide relevant responses based on the user's input.

NLP TECHNIQUES USED

The FAQ chatbot uses various NLP techniques to understand and respond to user inquiries. Intent recognition helps it understand the user's request or intention based on a set of predefined intents.



```
def chat():
    print("Hello, I am Kora,the moringa school FAQ bot (type Thank you to stop)!")
    while True:
        inp = input("You: ")
        if inp.lower() == "thank you":
            break

        results = model.predict([bag_of_words(inp, words)])[0]
        results_index = numpy.argmax(results)
        tag = labels[results_index]

        if results[results_index] >0.8:
            for tg in data["intents"]:
              if tg['tag'] == tag:
                  responses = tg['responses']

            print(random.choice(responses))
        else:
          print('I did not understand, kindly ask another question')

chat()
```

```
Hello, I am Kora,the moringa school FAQ bot (type Thank you to stop)!
You:
```

Entity recognition enables it to identify important information in the user's input such as names or locations, which helps it provide more personalized and accurate responses.
Text classification helps it identify the topic or theme of the user's input, allowing it to provide relevant responses.
These NLP techniques contribute to the chatbot's ability to improve the user experience and support operations for Moringa school.

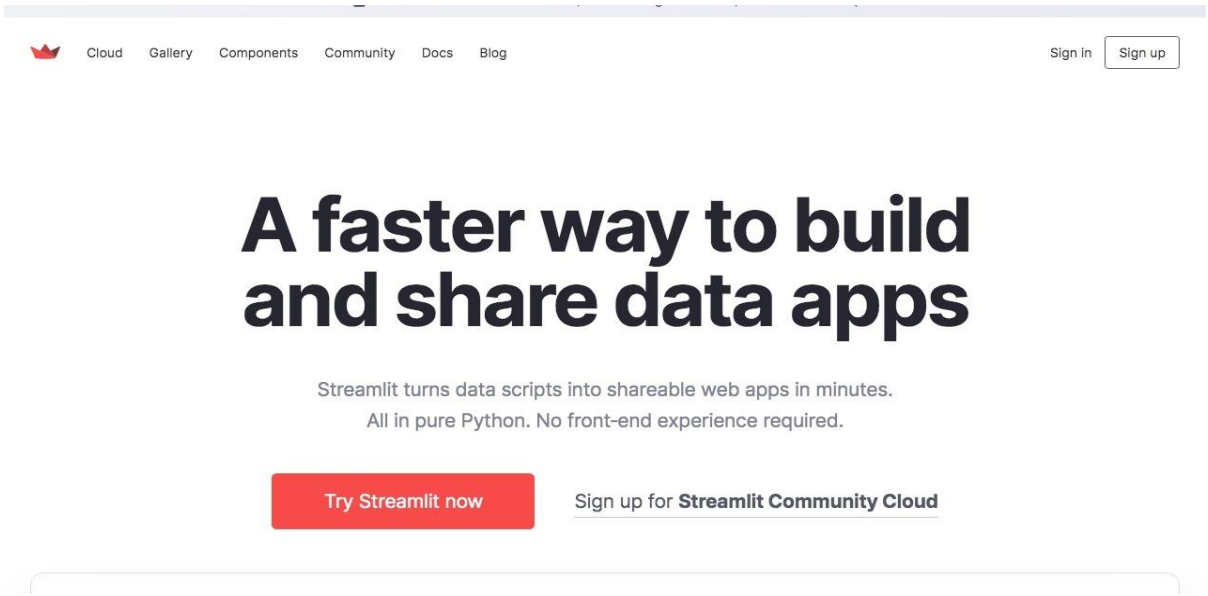
EVALUATION AND TESTING

When evaluating and testing the FAQ chatbot, it was found that the bot is capable of answering basic questions. We can place its accuracy rate at 80% based on our observations.

It answers 4 out of every 5 questions you ask it correctly. However, it may encounter syntax errors or struggle to provide relevant responses when faced with custom or complex questions that are not part of its predefined knowledge base. It is important to note that chatbots like this one require continuous improvement and training to expand their knowledge and capabilities. Regular testing and evaluation can help identify areas for improvement and ensure that the chatbot is meeting the needs of users. In order to maximize the effectiveness of the FAQ chatbot, it is essential to invest in ongoing maintenance, monitoring, and refinement to optimize its performance and ensure a positive user experience.

DEPLOYMENT

To determine the accuracy and effectiveness of the FAQ chatbot, we plan to deploy it on Streamlit, a platform for building data apps.



This will enable us to evaluate the chatbot's performance in a real-life situation and assess how well it is meeting the needs of users. While we have tested the chatbot in a controlled environment, it is important to validate its capabilities in a more realistic setting. This will allow us to identify any potential issues or limitations and make necessary adjustments to optimize its performance. We recognize that the true measure of success for the chatbot is how well it meets the needs of users and provides value to the organization. By deploying it on Streamlit, we can obtain valuable feedback and insights that will inform future enhancements and improvements. Our ultimate goal is to deliver a chatbot that is as accurate as possible, effective, and user-friendly, and deploying it on a real-life platform is a crucial step in achieving that goal.

CONCLUSION

In conclusion the FAQ chatbot is functional, but it still requires work and implementation. We hope to continuously improve the chatbot to recognise dialog management, provide personalization and learn from user interactions and improve its responses over time . Overall, the chatbot has the potential to feel more human-like as it evolves.