# AI BASED CHATBOT FOR ANSWERING FAQS

A PROJECT REPORT

*Submitted by*

**Sai Sudha Panigrahi**
(CB.EN.U4CSE17051)

**MAY - 2019**

# CERTIFICATE

## ACKNOWLEDGEMENT

I would first and foremost express my deep sense of gratitude to the computer Division, EIG, IGCAR for giving me the opportunity of interning at one of the best research institutes in the country. I would like to thank the Director IGCAR ,A. K. Bhaduri for the approval and the head of the computer division, Shri.R.Jahadeesan for providing all the resources required for the accomplishing the project.

Furthermore, I would extend my sincere thanks to Mr. Subba Raju whose valuable guidance and depth of knowledge helped me in building the project from the foundation.


.

**Table of Contents**

## SUMMARY

An AI based FAQ chatbot aims tocreate a chatbot framework and build a conversational model for credit cards and bank accounts based FAQ's. This project is a part of Interning at Indira Gandhi Centre for Atomic Research, Kalpakkam. The question is posed in Natural Language. Natural Language Question Answering is recognized as a capability with great potential.This has been specifically developed pertaining to the needs and user requirements of IGCAR.

## ABSTRACT AND MOTIVATION

The aim of this project is to build a framework that can answer the posed question in an efficient and accurate methodology.

It spares the users from the difficulty of contacting the administrator for various frequently asked questions, thus saving time, energy and giving the user a convenient interface.

This project aims at delivering the knowledge in user intended form by applying concepts of intelligence. In this world of automation, faster and more accurate systems are more preferred because time lost is money and resources lost.

Hence with this motivation the system was developed. The vast amounts of multi-disciplinary data available in various forms should be made available to the employees and should be made aware of the understanding that quick and accurate information at the right time can be of great use to the organization.

.

## OBJECTIVE

Automation is the driving force of this generation. Automation has seeped into every possible application there is.

This project aims to create a chatbot framework and build a conversational model for bank based FAQ's.

The chatbot needs to handle simple queries about credit cards like eligibility, about different cards like visa, master, rewards, annual fees,reward points, benefits ,how to apply,cash back options, CVV numbers and so on.

We also want it to handle some contextual responses.

This framework can also be used for any generic purpose FAQ's by providing it with the necessary training data**.**

## PROJECT DESCRIPTION, MODULES AND CODE SNIPPETS

FIRST IMPLEMENTATION

### Project Description

Our first implementation included using NLTK(Natural Language Toolkit is a leading platform for building Python programs to work with human language data) and the scikit library
Text Pre- Processing with NLTK

text data is all in text format (strings). However, the Machine learning algorithms need some sort of numerical feature vector in order to perform the task. So before we start with any NLP project we need to pre-process it to make it ideal for working

Basic text pre-processing includes:

- Converting the entire text into uppercase or lowercase
- Tokenization
- Removing Noise
- Removing Stop words
- Stemming
- Lemmatization

## Bag of Words

After the initial preprocessing phase, we need to transform text into a meaningful vector (or array) of numbers. The bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things

A vocabulary of known words.

A measure of the presence of known words.

The intuition behind the Bag of Words is that documents are similar if they have similar content. Also, we can learn something about the meaning of the document from its content alone

## TF-IDF Approach

A problem with the Bag of Words approach is that highly frequent words start to dominate in the document but may not contain as much "informational content". Also, it will give more weight to longer documents than shorter documents.

One approach is to rescale the frequency of words by how often they appear in all documents so that the scores for frequent words like "the" that are also frequent across all documents are penalized. This approach to scoring is called Term Frequency-Inverse Document Frequency, or TF-IDF for short, where:

TF = (Number of times term t appears in a document)/(Number of terms in the document)

IDF = 1+log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in.

Tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection

## Cosine Similarity

TF-IDF is a transformation applied to texts to get two real-valued vectors in vector space. We can then obtain the Cosine similarity of any pair of vectors by taking their dot product and dividing that by the product of their norms. That yields the cosine of the angle between the vectors. Cosine similarity is a measure of similarity between two non-zero vectors. Using this formula we can find out the similarity between any two documents d1 and d2.

Cosine Similarity (d1, d2) = Dot product(d1, d2) / ||d1|| * ||d2||
where d1,d2 are two non zero vectors.

## MODULES AND CODE SNIPPETS:

**Pre-processing the raw text**

```
lemmer = nltk.stem.WordNetLemmatizer().
def LemTokens(tokens):
   return [lemmer.lemmatize(token) for token in tokens]
remove_punct_dict = dict((ord(punct), None) for punct in string.punctuation)
def LemNormalize(text):
 return       LemTokens(nltk.word_tokenize(text.lower().translate(remove_punct_dict)))
```

**Generating Response**

```
def response(user_response):
   robo_response="
   sent_tokens.append(user_response)
   TfidfVec = TfidfVectorizer(tokenizer=LemNormalize, stop_words='english')
   tfidf = TfidfVec.fit_transform(sent_tokens)
   vals = cosine_similarity(tfidf[-1], tfidf)
   idx=vals.argsort()[0][-2]
   flat = vals.flatten()
   flat.sort()
   req_tfidf = flat[-2]
   if(req_tfidf==0):
      robo_response=robo_response+"I am sorry! I don't understand you"
      return robo_response
   else:
      robo_response = robo_response+sent_tokens[idx]

      return robo_response
```

## SCREEN SHOTS OF CHATBOT IN ACTION :

```
what is a chatbot?
ROBO: design
the chatbot design is the process that defines the interaction between the user and the chatbot.the chatbot des
igner will define the chatbot personality, the questions that will be asked to the users, and the overall inter
action.it can be viewed as a subset of the conversational design.
what are chatbots used for?
ROBO: chatbots are typically used in dialog systems for various practical purposes including customer service o
r information acquisition.
who is alan turing?
ROBO: background
in 1950, alan turing's famous article "computing machinery and intelligence" was published, which proposed what
 is now called the turing test as a criterion of intelligence.
what is eliza?
ROBO: while eliza and parry were used exclusively to simulate typed conversation, many chatbots now include fun
ctional features such as games and web searching abilities.
```

## SECOND IMPLEMENTATION

### Project Description

Although the above implementation gave some results, its accuracy was not sufficient for a fully functioning chatbot.

So, a different approach was tried to improve the accuracy and give reliable results.

This approach constituted of 3 steps:

- transformation of conversational intent definitions to a Tensorflow model
- build a chatbot framework to process responses
- incorporating basic context into our response processor

# Little about Tensorflow

Created by the Google Brain team, TensorFlow is an open source library for numerical computation and large-scale machine learning. TensorFlow bundles together machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor. It uses Python to provide a convenient front-end API for building applications with the framework, while executing those applications in high-performance C++.

TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training.

# Modules and code snippets

### Transforming Conversational Intent Definitions to a Tensorflow Model

```
for intent in intents['intents']:
    for pattern in intent['patterns']:
        w = nltk.word_tokenize(pattern)
        documents.append((w, intent['tag']))
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
words = [stemmer.stem(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
classes = sorted(list(set(classes)))
```

### Forming the documents of words into tensors of numbers.

```
for doc in documents:
    bag = []
    pattern_words = doc[0]
    pattern_words = [stemmer.stem(word.lower()) for word in pattern_words]
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1
    training.append([bag, output_row])
random.shuffle(training)
```

#### Building the Chatbot Framework

Producing a bag-of-words from user input

```
def bow(sentence, words, show_details=False):
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)
    return(np.array(bag))
```

### Response processor

```
ERROR_THRESHOLD = 0.25
def classify(sentence):
```

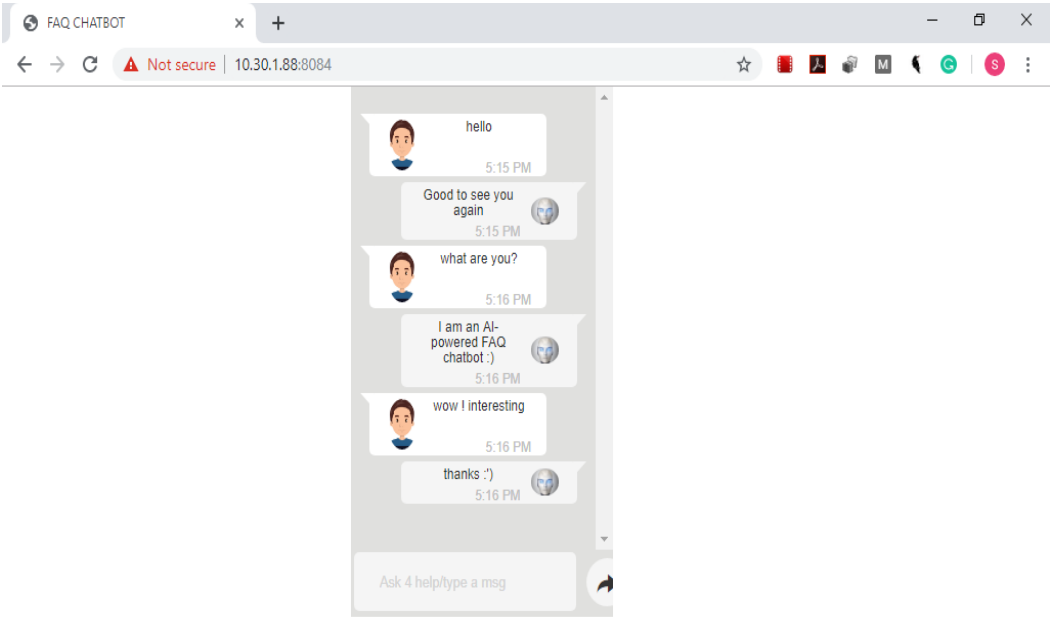
```
        results = model.predict([bow(sentence, words)])[0]
        results = [[i,r] for i,r in enumerate(results)if (r>ERROR_THRESHOLD)]
                if(0==len(results)):
                        return 0
                else:
                        results.sort(key=lambda x: x[1], reverse=True)
                        return_list = []
                        for r in results:
                                return_list.append((classes[r[0]], r[1]))
                        return return_list


def response(sentence, userID='123', show_details=False):
    results = classify(sentence)
    if results==0:
        return("i didn't get you, can you be more specific? :)")
    if results:
     while results:                  //useful in case of contextualization,lesser probability result can be
chosen
            for i in intents['intents']:
                if i['tag'] == results[0][0]:
                        return(random.choice(i['responses']))
```

## Contextualization

To achieve this we will add the notion of ‘state’ to our framework. This is comprised of a data-structure to maintain state and specific code to manipulate it while processing intents.

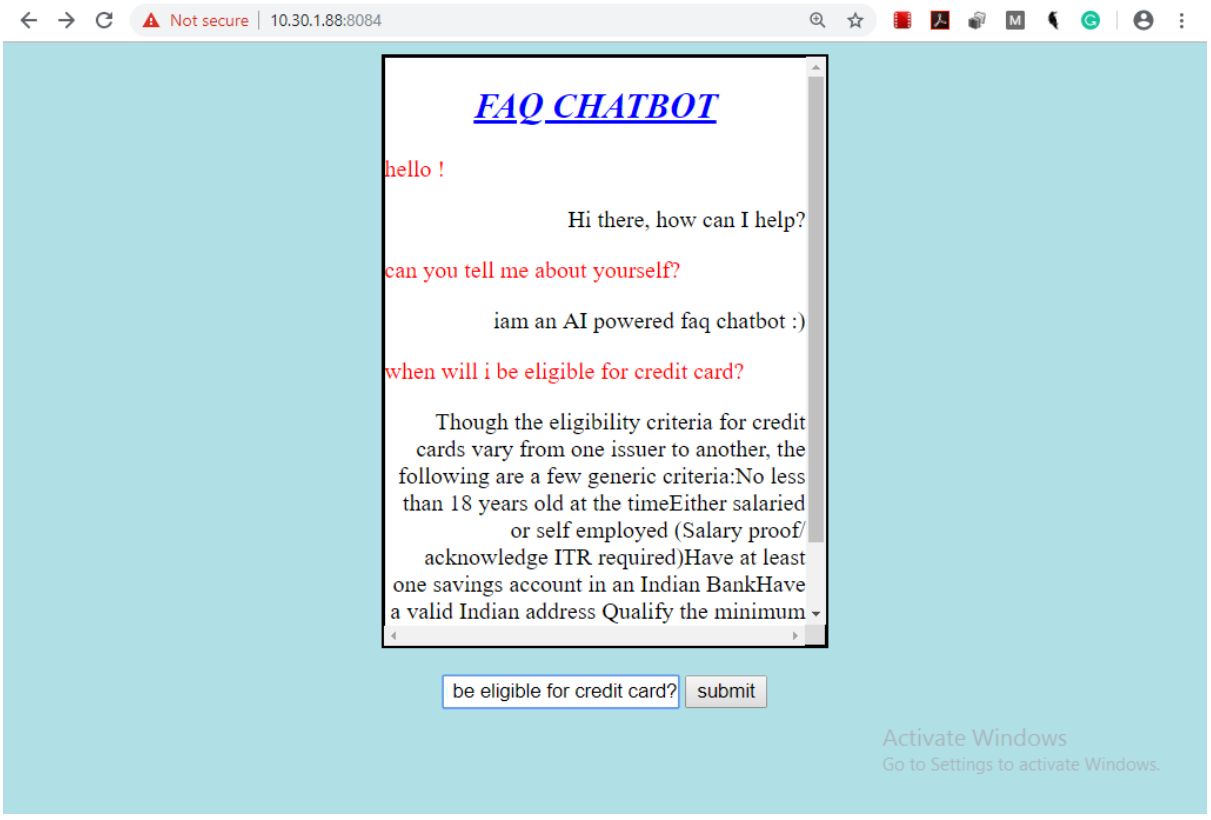
```
                if i['tag'] == results[0][0]:
                    if 'context_set' in i:
                       context[userID] = i['context_set']
                    if not 'context_filter' in i or \
                        (userID in context and 'context_filter' in i and i['context_filter'] == context[userID]):
                        return print(random.choice(i['responses']))
```

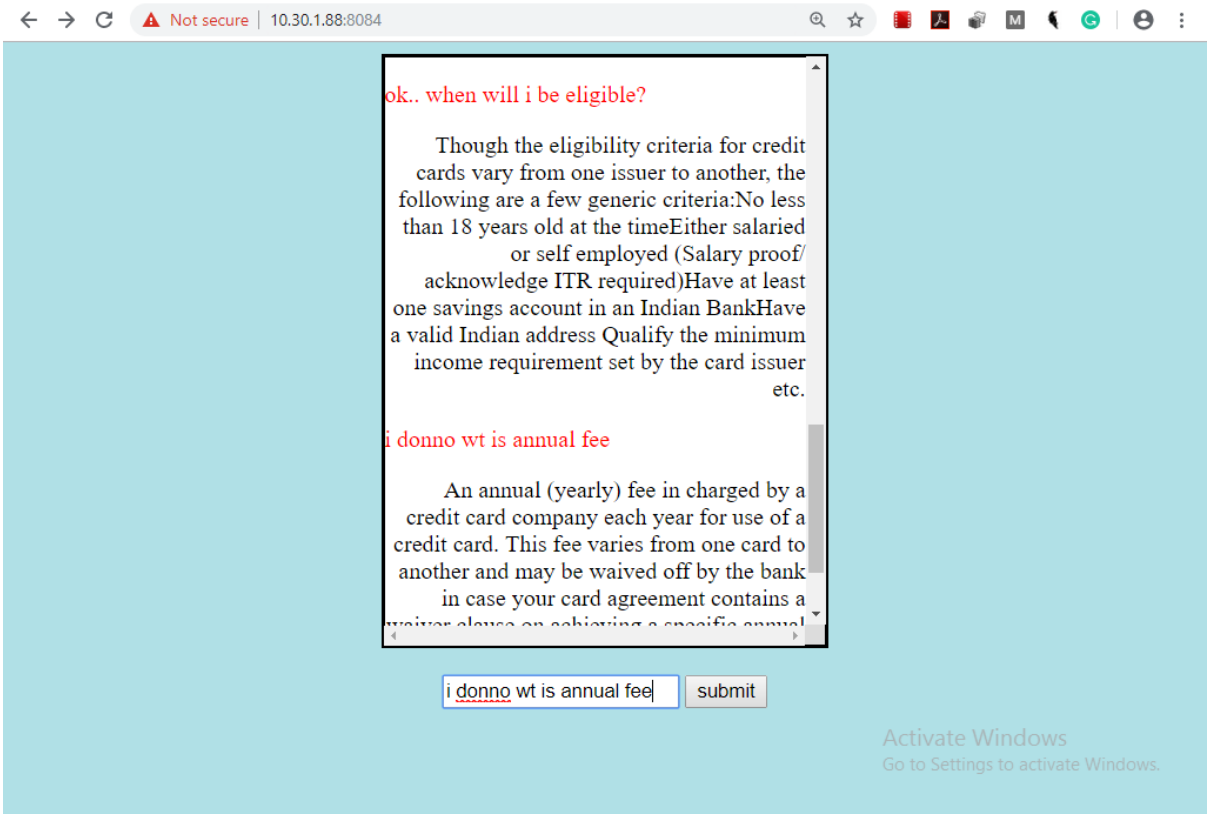
**Example of a Contextual response:**

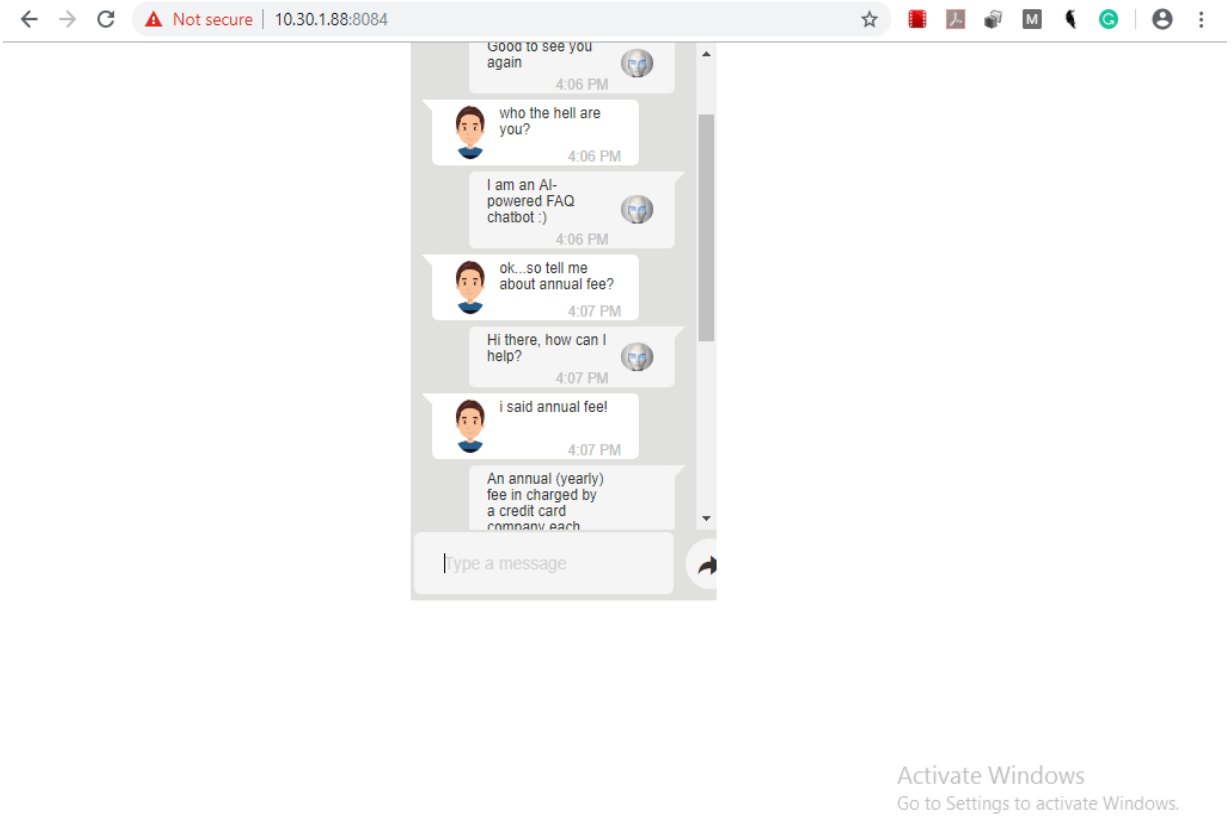



**We also constructed a basic frontend application using html, jquery and bootstrap to test our chatbot.**

Screen shots of chatbot in action*:*
*(initial stages)*




Not secure | 10.30.1.88:8084
ok.. when will i be eligible?
Though the eligibility criteria for credit cards vary from one issuer to another, the following are a few generic criteria:No less than 18 years old at the timeEither salaried or self employed (Salary proof/ acknowledge ITR required)Have at least one savings account in an Indian BankHave a valid Indian address Qualify the minimum income requirement set by the card issuer etc.
i donno wt is annual fee
An annual (yearly) fee in charged by a credit card company each year for use of a credit card. This fee varies from one card to another and may be waived off by the bank in case your card agreement contains a
i donno wt is annual fee
submit
Activate Windows
Go to Settings to activate Windows.


Not secure | 10.30.1.88:8084
Good to see you again
4:06 PM
who the hell are you?
4:06 PM
I am an AI-powered FAQ chatbot :)
4:06 PM
ok...so tell me about annual fee?
4:07 PM
Hi there, how can I help?
4:07 PM
i said annual fee!
4:07 PM
An annual (yearly) fee in charged by a credit card company each
Type a message
Activate Windows
Go to Settings to activate Windows.

# 1.TECHNICAL SPECIFICATION

## 1.1 Software Specifications and python libraries used in theimplementation

| Sr. No | Software | Version | Use case |
|---|---|---|---|
| 1. | Python | 3.4 | programming language. |
| 2. | Tensorflow | r1.13 | For the backend of the CNN classification. |
| 3. | Nltk | 3.2.5 | For the NLP tools like dependency parsing. |
| 4. | Anaconda | 1.6.2 | Platform for the IDE |

## Future implementations

This project has numerous functionalities yet to be explored in detail. Like building it as a working and deployable functionality for igcar with a front end and back end connectivity.

Other functionalities like including an automatic spell check feature could also improve the user experience.

## REFERENCES


- https://www.analyticsvidhya.com/blog/2018/01/faq-chatbots-the-future-of-information-searching/
- https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e
- https://github.com/yogeshhk/FAQChatbot
- https://chatbotsmagazine.com/contextual-chat-bots-with-tensorflow-4391749d0077
- https://www.guru99.com/what-is-tensorflow.html
- https://www.datacamp.com/community/tutorials/tensorflow-tutorial