NOVATEUR PUBLICATIONS
INTERNATIONAL JOURNAL OF INNOVATIONS IN ENGINEERING RESEARCH AND TECHNOLOGY [IJIERT]
ISSN: 2394-3696 Website: ijiert.org
VOLUME 8, ISSUE 6, June. -2021

CUSTOMER SUPPORT CHATBOT USING NATURAL LANGUAGE PROCESSING

Richa Ranveera
UG Student,Dept of Computer Science and Engineering,
New Government Engineering college ,Raipur,Chattisgarh,India
Email-Id:richaranveera141@gmail.com

Akanksha Kesharwani
UG Student,Dept of Computer science and Engineering,
New Government Engineering College ,Raipur,Chattisgarh,India
Email-Id:keshakanksha16@gmail.com

Srishti Kumari
UG Student,Dept of Computer Science and Engineering,
New Government Engineering College ,Raipur,Chattisgarh,India
Email-Id:srishtisk142000@gmail.com

ABSTRACT
Customer support has become one of the most important communication tools used by companies to provide before and after-sale services to customers. The immediate response to all customer query is very necessary for company success. In this paper we focus on, providing a Chatbot that will see to all the queries user have and will provide a solution or answer to that.Usually companies have many people involved for dealing customer queries but this is time-consuming and tedious job to be done. For solving these problems, Chatbot was created for use. Generally, the frequently asked customer questions and corresponding answers and description about the company details are stored in a text file. So in this model, it will take the customer's question as input, preprocessing them using some Natural Language Processing techniques that include Tokenization, Lemmatization, and stemming, find the cosine similarity between the user query and provides a score for each answer,the more suitable and perfect answer with more cosine similarity score will be considered as the answer for the given question. The Chatbot helps to provide high accuracy by proving the correct and satisfying answer to the customer's question for a company.

Keywords: Natural Language Processing; Tokenization; Lemmatisation; Stemming; Cosine Similarity; Term Frequency-Inverse Document Frequency;

INTRODUCTION
A chatbot is described as one of the most advanced and promising expressions of interaction between humans and machines. From a technological point of view, a chatbot represents the natural evolution of a Question-Answering system using Natural Language Processing (NLP). There are lots of queries that the customer have when the have to purchase any product ,to deal with all of that chatbot is developed which is a computer program or an artificial intelligence which conducts a conversation via auditory or textual methods. The importance of these applications appears when no technicians manage the customer service office due to the end of working time or their presence outside the office. The repeatedly asked question's answer and valuable details about the company will be stored in a corpus text file and Each sentence will be considered as a document on its own. when a user input his query and then we use tf-idf and cosine similarity to compare them both. Each answers will be provided with a score based on the user given question. The higher score answer with higher similarity will be preferred. We get the most similar sentence from the dataset that is most similar to the query and generate it as answer.

SCHEMATIC MODEL

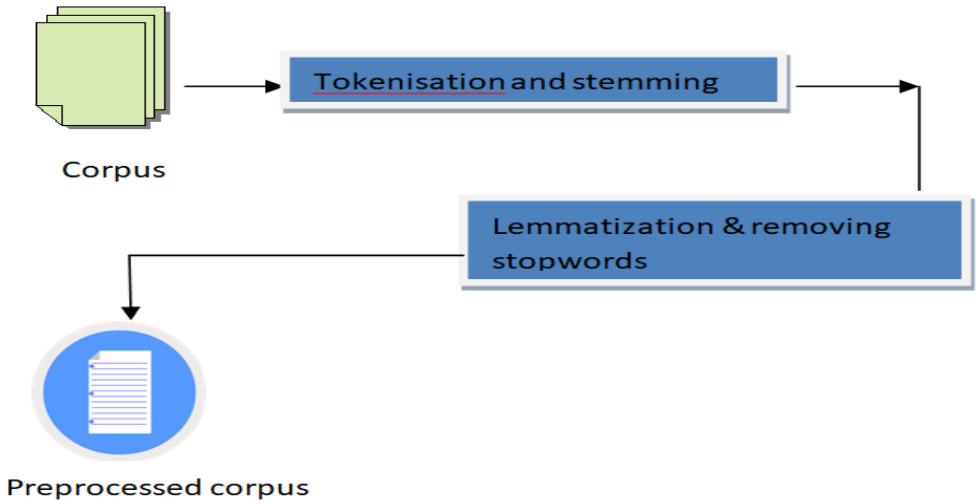

Fig.1 Preprocessing the corpus text file

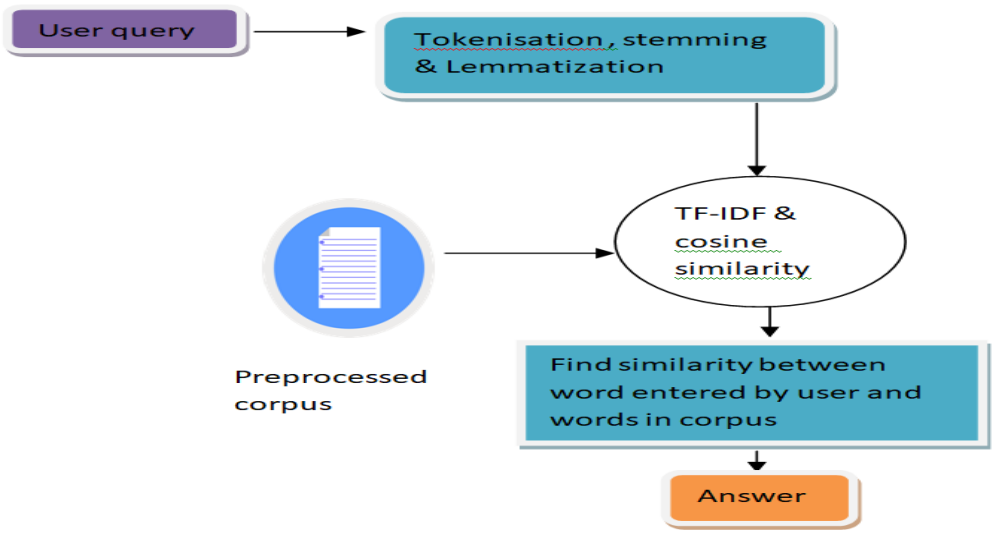

Fig .2 Model for generating response from user query

Pseudocode for the proposed Chatbot model

1.Start

PREPROCESSING:

2. FUNCTION initialize():

a.. Open Corpus dataset (chatbot .txt)in read mode and assign data to read variable

b. data=data.lower ().Store the sentence tokens in sen_tok and store the word in word token and lemmatize the token

3. FUNCTION LemTokens():

a. Lemmatize the token generated b. END

4. FUNCTION stemTok ():

a. Stemming all the tokens b. END

5. FUNCTION Greets(sentence):

a. Create GREET_INP=array (“hello”,”hii”,”hey”,”greetings”,”whatsup”,”hi there”).

b. Create GREET_RESPONSES = array["hi", "hey", "hi there", "hello”]

c. FOR each words in the sentence: i. IF word lower is in GREET_INP:

d. RETURN random choice in : GREET_RESPONSES

d. END

6. FUNCTION response(user_responses): // generating Response of user query
a. Add sen_tok and user_responses
b. Make A variable Tf-idVec to store the vector & STORE vector of sen_tok in Tf-id Vector
c. Find the cosine similarity of the user query & word in corpus and store into vals
d sort values in vals and flatten it and STORE it in FLAT // Cosine Similarity
e. storing the index value of high score and SCORE= FLAT [-2]
f. IF SCORE is equal to 0:
i. Response="I am sorry! I don't understand you" and RETURN Response
ELSE: i. Response= sen_tok[index] and. RETURN Response END IF
7. FUNCTION Chatbot():
a. CALL PREPROCESSING : - i. GET user_responses and lower it
. ii. IF user_responses!="Quit":
1. IF user_responses= "Thank You" :Then Flag=false and print "you are welcome"
iii. ELSE:
1. IF user_responses= Greets(): Call Greets
2. ELSE: a. CALL response(user_responses).

c. END chatbot
d. END.

METHOD AND TECHNIQUE

TEXT PREPROCESSING

First task is to do preprocessing on our data.The text data that we have needs to be preprocessed as we have all data in text format (strings)and our Machine learning algorithms need numerical feature vector in order to perform the task

- Text pre-processing includes: Converting entire text into lowercase & Removing Stop words. : some extremely common words are of little value These words are called stop words & Removing Noise i.e everything that is not in stanadard number or letter.
- Tokenization: Tokenization is a process of converting the normal text strings into a list of tokens. & Stemming: Stemming is the process of reducing derived words to their stem, base or rootfor ex : "given", "giving", the result would be a single word "give".
- Lemmatization: Lemmatization give importance to context and convert the words to its lemma. Examples "better" and "good" are in the same lemma so considered the same.
- In this Pre-processing technique, each words in sentence is separated and stored in separate list for words. This pre-processing technique is applied to both the user query and the corpus of the data present It makes processing faster.

FEATURE ENGINEERING

After the initial preprocessing phase, we have to transform text into a meaningful vector (or array) of numbers.. After the query and all the documents are pre-processed into two separate lists, they needed to be compared to get the scores. So this is basically done using Cosine Similarity and TF-IDF approach.

TF-IDF Approach : Tf-idf weight is a transformation which is applied on to text to get vectors in vector space. This approach to scoring is to rescale the frequency of words by how much they come in document so that scores for frequent words like "the" gets penalized. Tf-IDF convert the collection of raw document to a matrix of Tf-idf features.

Term Frequency: It is a scoring of the frequency of the word in the current document.

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

Inverse Document Frequency: is a scoring of how rare the word is across documents.

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

Cosine Similarity : To generate a response from the Chatbot for input questions, the concept Cosine similarity will be helpful .It is a measure that tells us the similarity between two non-zero vectors. The

similarity between any two documents m1 and m2 is Cosine Similarity (m1, m2) = Dot product(m1, m2) / ||m1|| * ||m2||
,where m1,m2 are two non zero vectors.

GENERATING RESPONSE

```
In [9]: from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.metrics.pairwise import cosine_similarity

In [10]: def response(user_response):
             robo_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

In [11]: flag=True
         print("ROBO: My name is Robo. I will answer your queries about Chatbots. If you want to exit, type Bye!")

         ROBO: My name is Robo. I will answer your queries about Chatbots. If you want to exit, type Bye!
```

Fig .3: code for response

For generating a response from our bot for input questions, the concept of document similarity will be. Used. Cosine similarity will be helpful to find the similarity between words entered by the user and the words in the corpus.In this Cosine Similarity, the score will be provided for each vector based on the occurrence of how much token present in the question.The token in the question occurs in the answer more, than that answer will be provided with a high score. After providing every answer a score, the answer with that has higher score will be displayed as an answer to the corresponding query . So ,In this model we basically use Tf-Idf and cosine similiarity to compare user query and word in corpus and get the most similar sentence from the dataset that is most similar to the query and output the response.

```
In [*]: while(flag==True):
            user_response = input()
            user_response=user_response.lower()
            if(user_response!='bye'):
                if(user_response=='thanks' or user_response=='thank you' ):
                    flag=False
                    print("ROBO: You are welcome..")
                else:
                    if(greeting(user_response)!=None):
                        print("ROBO: "+greeting(user_response))
                    else:
                        sent_tokens.append(user_response)
                        word_tokens=word_tokens+nltk.word_tokenize(user_response)
                        final_words=list(set(word_tokens))
                        print("ROBO: ",end="")
                        print(response(user_response))
                        sent_tokens.remove(user_response)
            else:
                flag=False
                print("ROBO: Bye! take care..")

        hi
        ROBO: *nods*
        hello
        ROBO: hi
        can u help me?
        ROBO: thus, for example, online help systems can usefully employ chatbot techniques to identify the area of help that users req
        uire, potentially providing a "friendlier" interface than a more formal search or menu system.

In [ ]:
```

Fig4. Final Chatbot output

CONCLUSION

In this project, we build customer support chatbot for helping the companies to have 24 hours of automated responses. After analyzing the dataset and understanding the importance to have automated responses to customers and companies. If the questions are unable to answer, then newer questions can be added in the dataset and increase their accuracy. This chatbot model will help to improve customer satisfaction by answering all user query.

REFERENCES

1) Niranjan Dandekar,suyog Ghodey "Implemnetation of a chatbot using natural language processing" ISBN:978-93-86171-88-7
2) X. Chen, H. Xie, F. Wang, Z. Liu, J. Xu, and T. Hao, "Natural Language Processing in Medical Research: A Bibliometric Analysis,"
3) X. Schmitt, S. Kubler, J. Robert, M. Papadakis and Y. LeTraon,A Replicable Comparison Study of NER Software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate,2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS}
4) Naveen S, "AUTOMATED CHATBOT IMPLEMENTED USING NATURAL LANGUAGE PROCESSING" e-ISSN: 2582-5208.