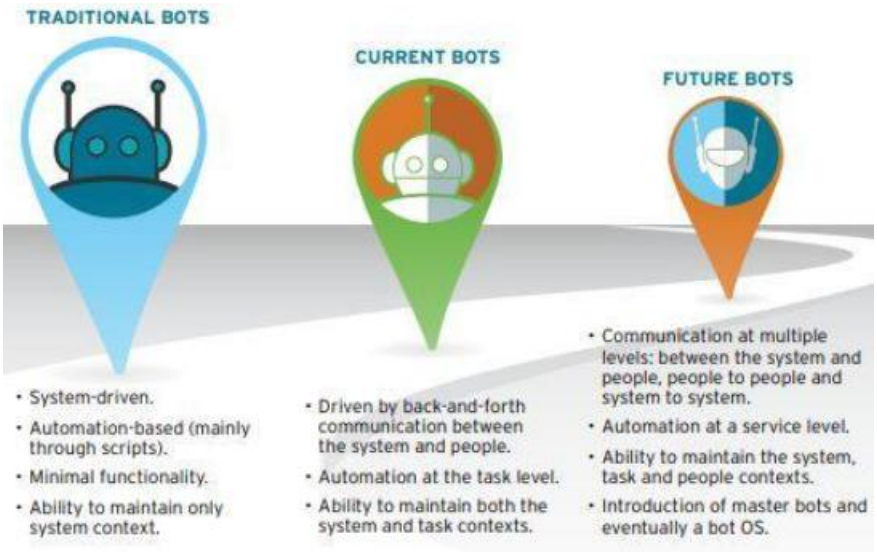
智能聊天机器人项目报告

一、 背景知识

聊天机器人是人造的以智慧为动力的软件，如 Siri，Alexa，谷歌助理等，它们存在于设备中，应用程序，网站或其他网络，试图衡量消费者的需要，然后帮助他们执行一个特定任务，如商业交易，酒店预订，表单提交等等。今天几乎每个公司都部署聊天机器人与用户交流。公司使用聊天机器人的一些方式是：提供航班信息；连接客户和他们的财务；作为客户支持等。

聊天机器人的历史可以追溯到 1966 年，当时韦森鲍姆发明了一种名为“伊丽莎”(ELIZA)的电脑程序。它仅仅从 200 行代码中模仿一个心理治疗师的言语。

聊天机器人的变革历程：



二、 聊天机器人的实现

文本数据的主要问题是它都是文本格式(字符串)。然而，机器学习算法需要某种数值特征向量来完成任务。因此，在我们开始任何 NLP 项目之前，我们都需对其进行预处理。基本文本预处理包括：

将整个文本转换为大写或小写，这样算法就不会将大小写的相同单词视为不同的单词

词语切分：指将普通文本字符串转换为符号列表的过程。也就是我们真正想要的词。句子分词器可用于查找句子列表，单词分词器可用于查找字符串形式的单词列表。

NLTK 数据包包括一个用于英语的预训练 Punkt 分词器。

去除噪声，即所有不是标准数字或字母的东西。

删除停止词。有时，一些在帮助选择符合用户需要的文档方面似乎没有什么价值的常见单词被完全排除在词汇表之外。这些单词叫做停止词。

词干提取：词干提取是将词尾变化词(有时是派生词)还原为词干、词根或词根形式(通常是书面形式)的过程。例如，如果我们要提取下列词：“Stems”，“Stemming”，“Stemmed”，“and Stemtization”，结果将是一个词

“stem”。

词形还原：词干提取的一个细微变体是词形还原 。它们之间的主要区别在于，词干提取可以创建不存在的词，而词元是实际的词。所以你的词根，也就是你最终得到的词，在字典里通常是查不到的，但词元你是可以查到的。词形还原的例子如：“run”是“running”或“ran”等词的基本形式，或者“better”和“good”是同一个词元，因此它们被认为是相同的。

在初始预处理阶段之后，我们需要将文本转换为有意义的数字向量(或数组)。单词袋是描述文档中单词出现情况的文本表示。它包括两个东西:

一个已知词汇表。

•一个对已知词存在的量度。

为什么它被称为一个单词袋？这是因为关于文档中单词的顺序或结构的任何信息都会被丢弃，模型只关心已知单词是否出现在文档中，而不关心它们在文档中的位置。

单词袋的直观感受是，如果文档的内容相似，那么文档就相似。此外，我们还可以从文档的内容中了解一些文档的含义。

例如，如果我们的字典包含单词{Learning, is, the, not, great}，并且我们想向量化文本“Learning is great”，我们将有以下向量:(1,1,0,0,1)。

单词袋方法的一个问题是，频繁出现的单词开始在文档中占据主导地位(例如，得分更高)，但可能并没有包含太多的“有信息内容”。此外，它将给予较长的文档更多的权重。

一种方法是根据单词在所有文档中出现的频率重新调整单词的频率，以便对“the”等在所有文档中也经常出现的单词适当降低权重。这种评分方法称为检索词频率-逆文档频率，简称 TF-IDF，其中:

检索词频率: 是当前文档中单词出现频率的得分。

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

逆文档频率：是这个词在文档中罕见度的得分。

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.

余弦相似度：

TF-IDF 是一种在向量空间中得到两个实值向量的文本变换。然后我们可以通过取点积然后除以它们的范数乘积来得到任意一对向量的余弦相似度。接着以此得到向量夹角的余弦值。余弦相似度是两个非零向量之间相似度的度量。利用这个公式，我们可以求出任意两个文档 d1 和 d2 之间的相似性。

Cosine Similarity (d1, d2) = Dot product(d1, d2) / ||d1|| * ||d2||

我们现在真正地开始聊天机器人的编写：（代码部分由粗体标出）

导入必备库：

```
import nltk
import numpy as np
import random
import string # to process standard python strings
```

我们将使用聊天机器人的 Wikipedia 页面作为我们的语料库。从页面复制内容并将其放入名为“chatbot.txt”的文本文件中。然而，你可以使用你选择的任何语料库。

读入数据：

我们将阅读 corpus.txt 文件，并将整个语料库转换为句子列表和单词列表，以便进行进一步的预处理。

```
f=open('chatbot.txt','r',errors = 'ignore')
raw=f.read()
raw=raw.lower()# converts to lowercase
nltk.download('punkt') # first-time use only
nltk.download('wordnet') # first-time use only
sent_tokens = nltk.sent_tokenize(raw)# converts to list of sentences
word_tokens = nltk.word_tokenize(raw)# converts to list of words
```

预处理原始文本：

我们将定义一个名为 LemTokens 的函数，它将接受符号作为输入并返回规范化符号。

```
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)))
```

关键字匹配：

接下来，我们将通过机器人定义一个问候函数，即如果用户的输入是问候语，机器人将返回相应的回复。ELIZA 使用一个简单的关键字匹配问候。我们将在这里使用相同的概念。

```
GREETING_INPUTS = ("hello", "hi", "greetings", "sup", "what's up","hey",)

GREETING_RESPONSES = ["hi", "hey", "*nods*", "hi there", "hello", "I am glad! You are talking to me"]

def greeting(sentence):
```

```
    for word in sentence.split():
        if word.lower() in GREETING_INPUTS:
            return random.choice(GREETING_RESPONSES)
```

生成回复：

为了让我们的机器人为输入问题生成回复，这里将使用文档相似性的概念。因此，我们首先需要导入必要的模块。

从 scikit learn 库中，导入 TFidf 矢量化器，将一组原始文档转换为 TF-IDF 特征矩阵。

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

同时，从 scikit learn 库中导入 cosine similarity 模块：

```
from sklearn.metrics.pairwise import cosine_similarity
```

这将用于查找用户输入的单词与语料库中的单词之间的相似性。

我们定义了一个回复函数，该函数搜索用户的表达，搜索一个或多个已知的关键字，并返回几个可能的回复之一。如果没有找到与任何关键字匹配的输入，它将返回一个响应:“对不起!”我不明白你的意思″

```
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
```

最后，我们将根据用户的输入来决定机器人在开始和结束对话时说的话。

```
flag=True
print("ROBO: My name is Robo. I will answer your queries about
```

```
Chatbots. If you want to exit, type Bye!")

    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..")
```

互动结果展示：

```
ROBO: My name is Robo. I will answer your queries about Chatbots. If you want to exit, type Bye!
hi
ROBO: I am glad! You are talking to me
```