ResumeBot – an Interactive Chatbot

Baranidhar Eswaramoorthy Manoharan, Raja Singh Ravi, Vigneshwar Sripad
Muma College of Business

Introduction

Chatbots can be defined as computer programs designed to simulate conversation with human users, and these conversations happen primarily over the internet. The main goal of a chatbot is optimization. Chatbots work in such a way that the person interacting with the chatbot will not realize that they are interacting with a virtual person. A chatbot can be programmed or designed with the same characteristics of the person for which it is being developed. In this project, we have developed the Chatbot for responding to recruiters and details which are present in the person's resume, so this can also be called a 'ResumeBot'. A resume in the form of a chatbot is a very good way to showcase our professional skills to recruiters. The Chatbot makes use of Natural Language Processing to identify what information the recruiter wants and respond accordingly. Chatbots mimic human behavior. They act as a conversational partner for the person on the other side. Other examples of Chatbots include Google Assistant and Alexa.

Another example of the chatbot is the Facebook Messenger, in which Chatbots are used by people who post ads on Facebook.

Problem Statement

Messaging has become a platform for everything like shopping, learning, dining, etc. People are spending more time messaging and less time reading mails or listening to voicemails.
If a recruiter needs any information or additional details about the job seeker, they must approach them through mail or phone call. Most of the time, they do not get immediate responses to process the application. Job portals like LinkedIn or Indeed, users have limitations to fill only the configured details. Even in resumes, it is not good to dump all the information that you have.

Solution

Our idea is to build a chatbot to help recruiters to know more about the job seeker. The ResumeBot works in such a way that the person interacting with the bot will not realize that they are interacting with a virtual person. This benefits job seekers to feed all the data about themselves and helps recruiters to the required information immediately.

Designing the data

The data set for a domain specific chatbot is subjective to the domain and its application usage. The reason for selecting job seeker profile as a specified domain is that – most of the profiles have similar structure and it is easy to add and update the data into the model. Also, the conversation between a job seeker and recruiter can get into several topics and multi-levels like education, projects, experiences, certifications, etc. related information. This kind of in-depth conversation is necessary to test out the contextualization of the framework. Here, the data set is confined to the English language because the python packages have all kind of NLP support for implementing and building the model. We have structured this data into the following model:

Tag: a unique name for each topic, the model covers
Sample Patterns: Examples of sample query of the topic. These are the possible pattern of questions asked by the recruiter.
Possible Responses: Sample responses of the topics. After identifying the topic, one of these responses will be randomly chosen as an answer to the query
Context: To identify the content of the topic.
Specifics: Specific details of the topic.
Datatype: The data type of responses for the topic.

Sample data set

A sample topic called “master” which has the details of job seeker’s master education details.

```
{
    "tag": "master",
    "patterns": ["Tell about his master's education",
                 "which year did he complete his master degree",
                 "When did he complete his master degree",
                 "what is his GPA in masters",
                 "where did he complete his master's",
                 "In which location did he do his master education",
                 "master degree details"],
    "responses":["He is pursuing his Master's Degree in Business Analytics and
                 Information Systems at the University of South Florida, Tampa, Florida.
                 His expected graduation date is May, 2020"],
    "specifics":[
            {
                "year":"His expected graduation date is May, 2020",
                "location":"He is doing is Master's in Tampa, Florida",
                "major":"His course major is Business Analytics and Information Systems",
                "university":"He is pursuing his degree from the University of South Florida",
                "gpa":"His GPA is 3.91 out of 4.0"
            }
        ],
    "context":"master",
    "type":"data"
},
```

This model which has been used to structure chat data has two main advantages. Firstly, if someone has existing data, it can mark every query to associate with a tag. In this way, pre-existent query data and response can be attached to the training dataset. Secondly, if someone has no prior dataset; anyone can create and add data in this format. There is no hassle to specify the flow of the conversation. All the inputs in these 6 input categories can be set by the person who wants to build the chatbot through this framework. Also, this does not require any special knowledge. We can see that there is no need to pre-set any conditional flow of the conversation. As every class in this structure of the dataset is discrete; if new data occurs, it can be easily entered as one of the existing tags or in a new tag without worrying about maintaining the relationship between the new data with other classes.

System design

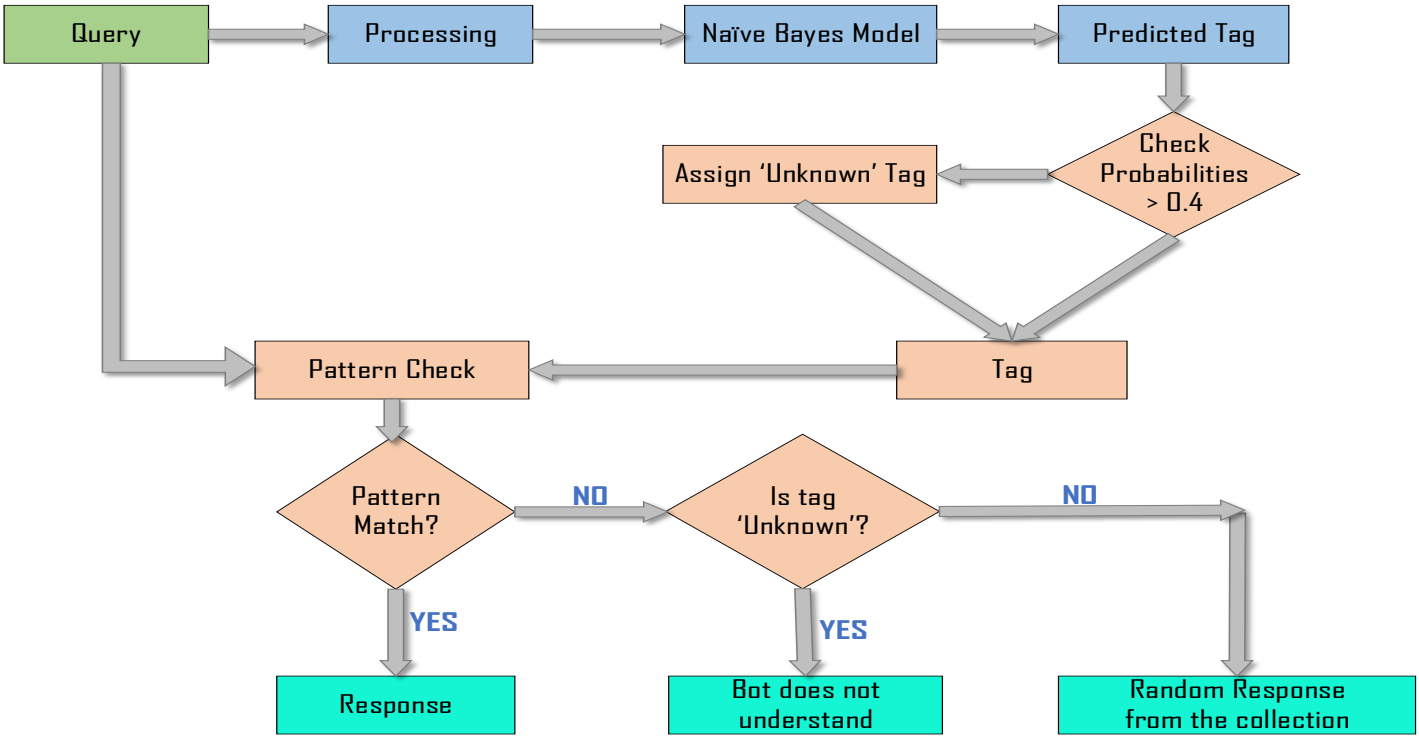
The ResumeBot is a domain-specific chatbot and we have a domain-specific data to train the chatbot. The data is pre-classified based on the tags(classes) and patterns (questions), and every class has some sample examples and some other data to track the flow of the conversation.

So, when a new query comes from a recruiter, we need to associate the query to our pre-existing classes based on the pattern that already exists in each class. Based on that topic identification; we give the user appropriate answers to their queries.

This looks like a text classification where the corpora are replaced with small sentences. Each class has only 3–10 sample sizes, and each sample contain only one sentence instead of a whole paragraph or document. Even the input sentence is one or two lines of sentences. This sparsity of data makes the model difficult to classify the data even with the well-established text classification techniques. Once the topic is identified, the appropriate response is picked from the dataset. When the query does not have any topic identity and if it matches with the “unknown” class, then responses are prepared based on NLP patterns.

Detailed Flow:

The below is the complete flow diagram of ResumeBot.



When a recruiter asks a new query, it undergoes the text processing steps like stop words removal, lemmatization, count vectorization, and TF – IDF vectorization. The query vector will pass through the trained Naïve Bayes model and it predicts the tag for the query. But there are chances like we get the class value with lesser probability. So, we check when the probability is greater than 40 %, then we accept the tag given by the model. Otherwise, we assign the tag value as 'unknown'.

In the next step, it goes for pattern check. If the actual query matches with any of the NLP patterns, the bot gives an answer and also sets the context. If the pattern does not match and the tag is not unknown, then the bot gets the random answer from the responses. If the pattern does not match and the tag is unknown, that means the bot does not understand the query asked by the recruiter.

Maintaining the context

The bot maintains the context by keeping the session in the database and also transfers the context value between client and server applications. We store the session and context details in the hidden field of the html page and this information keep on transferred between client and app servers.

To build this type of system we divided the entire flow into 3 major parts

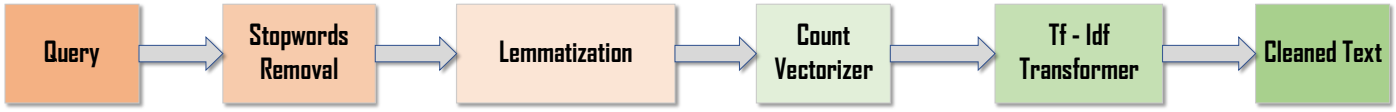
1) Pre-processing of patterns.

2) Writing NLP patterns

3) Building model

Pre-processing of patterns

The patterns are processed and converted into data that a computer can understand for building the model



1) Tokenization:

We have used the tokenizer because the Reviews there might be many smiley details of some phrases like 'YAAAY!!', 'Yappyyyy!!!',': P' etc. These phrases don't have any meaning and polarity cannot be determined with the use of these phrases. So, using tokenization for these phrases will not be considered by the model.

While doing the tokenization, the contraction words are taken care with the help of predefined rules. Also unnecessary punctuation symbols are also removed.

Example: "I won't replaced as I will not"

2) Stop Word Removal:

A stop word is a commonly used word that a pre-processing engine has been programmed to ignore so that we can save processing time and space. We imported the "stop words" package from NLTK library and customized the stop words list based on our requirement. We ignored the following set of words from the list. ['not', 'what', 'which', 'who', 'whom', 'do', 'does', 'did', 'when', 'where', 'why', 'how']. Along with the common words, punctuation marks and special characters are also removed for building topic identification model.

3) Part-Of-Speech (POS) Tagging:

NLTK pos tagging is used to tag each word with the appropriate Part-of-speech. This helps us to understand the contextual meaning of the sentences.

The words having list of POS tags (['CC','DT','EX','MD','PRP','IN','PRP$','RP','TO','UH']) are not considered while building the model for topic identification. But they are used while making NLP pattern matching because these POS tags are important to understand the language flow.

Example: The word "Book" can be used as a verb as well as a noun.

4) Lemmatization:

Normally lemmatization will return the existing word into the base or dictionary form of a word. We have used this pre-processing technique for the data as the patterns or queries can be given in the informal language and the model cannot predict the sentiment.

Example: Same word with different tenses have been brought down to its root level.

The words "projects", "took" are converted as "project","take".

5) Count Vectorization:

To Increase the sparseness in the vector matrix we have converted all the uppercase letters into lower case letters. This made the vector consider only unique words.

Example: The letter 'Good' can be written in the reviews as 'Good', 'good', 'GOOD' and many ways with upper and lower cases the vector matrix would consider every single case of good. After converting them into lower case, only 'good' is available, vector will consider the same.

6) TF-IDF Vectorization:

Term frequency and Inverse document frequency is a statistical measure used to evaluate how

important a word is to a document in a collection or corpus. We have used TF-IDF to give more weight to the words which are used less.

Consider a query "what are the projects you worked" is converted into "what project worked" after performing the pre-processing steps. As an end result, it will have an array of weights based on its occurrence.

Writing NLP Patterns

The topic identification model is not suitable for all types of questions. For the general questions like who are you? or Who is he?, mapping these into different classes are meaningless and it affects the model performance. In such scenarios, NLP patterns and Regex are helpful. POS tags are stored along with the words to identify the pattern of the queries. Once the correct pattern is matched, the response can be obtained.

POS tags are helpful while maintaining the context of the conversation. If the conversation is about the job seeker's bachelor education and when the recruiter asks the bot "When", the bot responds with an appropriate answer based on its previous context.

Building model

First, a deep learning neural network model is built with two hidden layers and an activation function of "softmax". The model is built using TensorFlow package and the training is done with 1000 iterations. But it does not work for us to build this bot, because the dataset is very small.

We tried to implement the same in other models like SVM and Recurrent Neural Networks. SVM is good for building text classification model but it is not a probabilistic model. As per our requirement, we need the probability scores for classes.
Recurrent Neural Networks are feedback networks which is not necessary for this application. Because, the conversation is not going to be discrete.

Finally, we end up with a simple naïve Bayes model and it worked well for this project. Also, it is easy for us to understand.

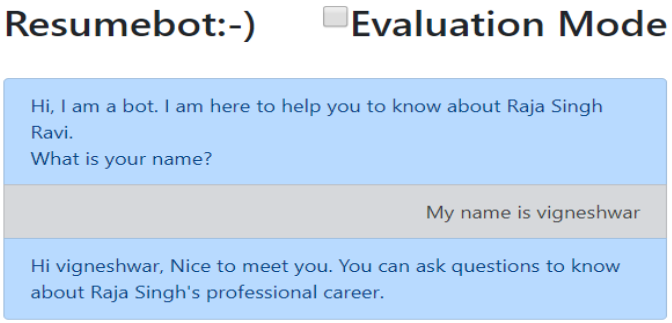
Use Cases

Building a complete chatbot is quite complex because we need to prepare the vocabularies and NLP libraries for that. So, in this ResumeBot application, we have implemented few cases and later we will try to complete the application and integrate with our portfolio along with an interview appointment module.

The following use cases are handled in this project;

1) Bot Introduction and basic salutation:

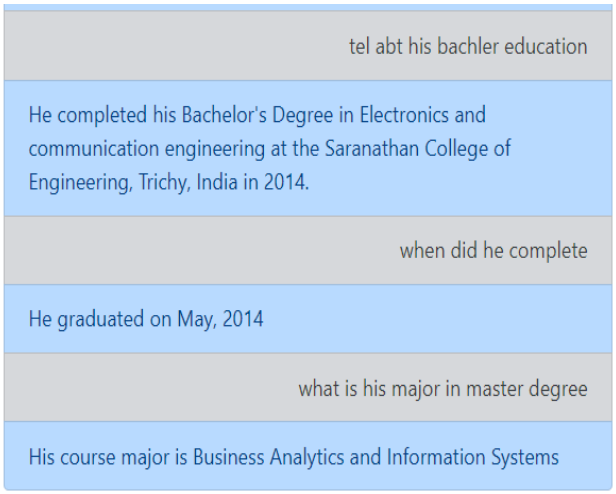
When the recruiter opens the ResumeBot chat window, it should be a blank window. The bot initiates the chat with an introduction and should tell its capabilities. On introduction, ResumeBot asks the recruiter's name and address him by his/her name. To implement this, we used NLTK's named-entity recognition technique which extracts the entity with the help of POS tagging based on its context in the sentence.
Along with this, we handled a few basic questions like who are you? How are you? using NLP patterns.



2) Maintaining context between conversations

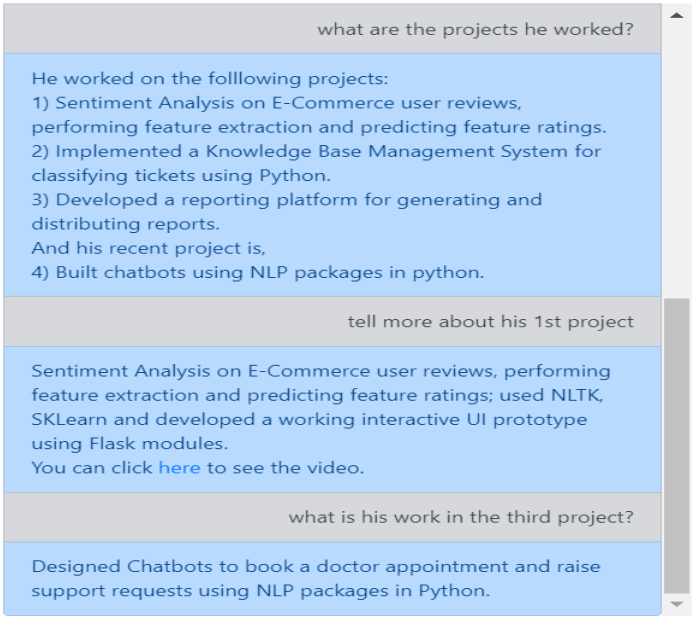
The context between sequential chats are necessary to maintain for a chatbot. In this ResumeBot, context of every chat is stored in database and also

passed between client and server side through cookies. When a recruiter first queries the bot about the job seeker's master's education and if he asks next where did he complete, then the bot tries to understand from the previous query that the recruiter is trying to ask where the job seeker complete his master's degree. We achieved this, with the help of NLP techniques.



3) Handling cardinal/ordinal cases

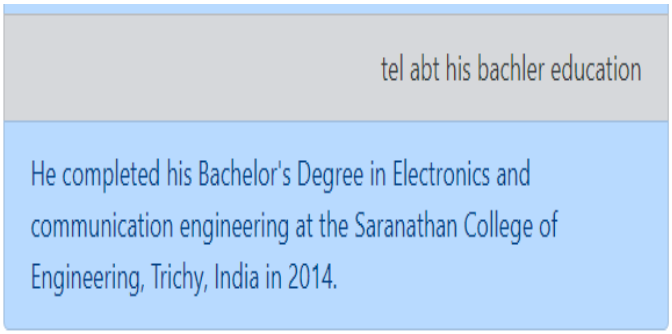
With the help of SpaCy package, the cardinal/ordinal tags are identified wherever required.
This maintains the context and also helps recruiters to know more about the list of things.



4) Handling typos and basic grammar:

To match with NLP patterns, the sentence should be grammatically correct and void of spelling mistakes. To handle that, we are using a python package called Gingerit. This helps to correct the words based on its syllabic sound and correct basic level of grammatical errors.

Sample Text: 'tel abt bachler education '
Output: ' Tell about bachelor education'



Challenges

1) Building vocabularies

We need to build word sets to handle different set of queries. For example, "master education" can also be queried as "graduate". Similarly, "Bachelor degree" can be queried as "undergraduate degree". Like these, words have synonyms and can be used differently based on the context. Resolving these issues require a lot of time and language specialists to build a possible set of words.
We tried using NLTK's Wordnet package which is a lexical database of English language. But, we require domain based knowledge to build those word sets and it is subjective to this application.

2) Multiple inputs

When a recruiter queries two or more questions in a single chat, then it is difficult for the bot to split, understand and answer the questions.
For Example, "When is he completing his master's degree and where is he doing?". Here, the recruiter likes to know the person's graduation date and the

location. Even though, the question points to a single class “Master Education” but the user expects two different answers.

3) Multiple outputs

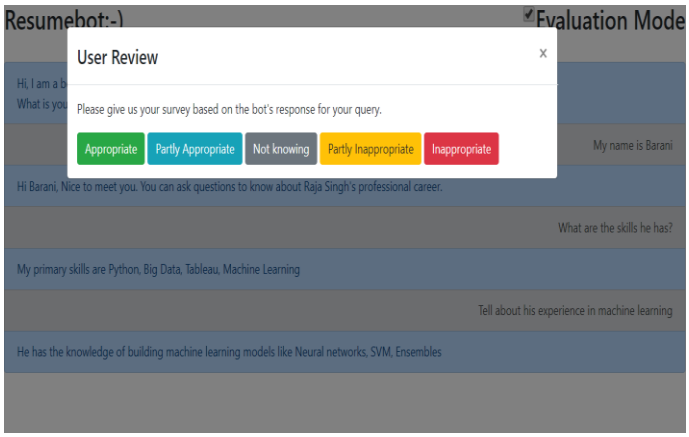
This bot responds quickly to the query asked by the recruiter and stores the context of the query. If the recruiter asked something wrong and if the person tries to correct himself in the next query, this causes a confusion. This bot is not designed to handle these situations.

Evaluation

ResumeBot is not only a text classification model, it also uses NLP techniques for pattern matching. In this domain specific chatbot, the samples for each class is 3-10 sentences. This makes it hard to split and test the model. The performance and quality of this bot is based on the user’s satisfaction and it cannot be measured using machine learning evaluation methods.
We evaluated this application by taking reviews from different users for each question and also saved the user’s query and bot’s response for our analysis.

We tested this application with 15 users and stored their survey along with the entire conversation in a database.

The user has to answer the survey at every time he gets an answer from the bot.



Based on the following units we evaluated this chatbot.

Label	Definition	Score
Appropriate	Concrete answer to the question	1
Partly appropriate	Generic response partly addressing the question	0.5
Not knowing	Neutral response that doesn't cause confusion	0
Partly inappropriate	Distant utterance to the question	-1
Inappropriate	Confusing answer	-2

From our analysis, the bot answers well for questions which are matched with the patterns list. We faced many scenarios where the questions are asked from the topic which is not present in our dataset. This helps to us retrain the model with new data.

Model Learning

We store all the conversations and surveys in a database.

Offline Learning – When analyzing the test surveys, we got new topics to add into our data. We manually analyzed the new queries and assigned a suitable topic for that. This is how we performed offline learning. This process happens on a periodic basis based on the application usage.

Online learning – is little tricky because, the bot should understand with whom it is talking before it starts learning.

Future Enhancements

This ResumeBot is not fully operational because we need to address the challenges mentioned before. Also building vocabularies and adding more patterns for each class improves the quality of the bot. Once we address those challenges, we are planning to add the appointment schedule module which helps to book interview appointments through Google calendar. This helps the recruiter to book an appointment with the profile owner for the next step. We are planning deploy this bot in Azure as a web service and use this in our personal portfolio.

REFERENCES

[1] Namita Mhatre, Karan Motani, Maitri Shah, Swati Mali, "Donna Interactive Chat-bot acting as a Personal Assistant" in International Journal of Computer Applications (0975 – 8887) Volume 140 – No.10, April 2016.

[2] Gayatri Nair, Soumya Johnson, V. Sathya, "Chatbot as a Personal Assistant" International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 20 (2018) pp. 14644-14649

[3] Anirudha Paul,Asiful Haque Latif,Foysal Amin Adnan &Rashedur M Rahman, "Focused domain contextual AI chatbot framework for resource poor languages"
https://tandfonline.com/doi/full/10.1080/24751839.2018.1558378?af=R&

WEB REFERENCES

[1] https://chatbotsmagazine.com/the-hidden-potential-of-chatbots-content-370e73eec50c

[2] https://chatbotslife.com/ultimate-guide-to-leveraging-nlp-machine-learning-for-you-chatbot-531ff2dd870c

EVALUATION REFERENCES

[1] https://arxiv.org/pdf/1603.08023.pdf

[2]https://books.google.com/books?hl=en&lr=&id=e01xDwAAQBAJ&oi=fnd&pg=PA30&dq=implementation+of+chatbot&ots=JFe5v3iZ2x&sig=eCvzAzOZxR0-zajkSO6AAXBTI6g#v=onepage&q=implementation%20of%20chatbot&f=false