Resume Chatbot

620 – Project Proposal

Introduction:

This document is a project proposal for 620 course. Document will explain about the features which will be built on the resume chatbot using NLP and deep learning concepts. It will also explain steps and the methods which will be used.

Motivation:

I have always wondered about the beauty of AI chatbot. Most of the companies are trying to integrate chatbot in some ways. This course has provided me good knowledge about the NLP. I want to use that knowledge and build an chatbot on top of my resume. Once developed, this can be deployed in my website to show future prospective interviewers and it will give a good experience to use NLP with deep learning.

Technology Stack and Data:

To build an intelligent chatbot, we need train the model with huge amount data and use deep learning for it.

Libraries:
Text Pre-processing: NLTK, Spacy, beautiful soup
Deep learning for creating neural net models: Keras, Tensorflow
NLP API(optional): rasa.ai
Other libraries as required

Data:
To perform this deep learning chatbot, we need to train the model with lots of data. Unfortunately my resume is not in format of QA(question answering type). So I need to generate lot of data with some existing data for this model.

JSON data with intent:
https://raw.githubusercontent.com/bvshyam/make_yourself_a_bot/master/workspace-watson.json

Web scrapping:

As the above link does not have all the data, we also might use web scrapping from below links which contains some managerial interview questions. Some answers need to be tweaked.

Links: https://www.paysa.com/blog/2017/05/23/33-common-software-engineering-interview-questions/

https://www.thebalance.com/top-interview-questions-and-best-answers-2061225

https://www.thebalance.com/top-job-interview-questions-2061228

Steps to build the chatbot:

Below are the list of steps which needs to be followed

1. Data import and cleaning:
    a. In this step we need to import all the necessary data from JSON and the web links.
    b. JSON data can be fetched using JSON package and web links data using beautiful soup.
    c. Chatbot works like "Garbage in and garbage out". So we need to correct all the answers and structure the output in JSON format.
2. Spell correction:
    a. Spell correction is one important step for correcting wrong input provided by the user.
    b. Use NLTK and its Glutenburg corpus to validate the spelling and change accordingly.
    c. Stacy library can also be used to correct the spelling.
3. Name entity recognition:
    a. Identify the input user details and find name-entity recognition. This will be useful to identify the person and his context.
4. Word Embedding:
    a. Once we perform all the preprocessing steps, we need to perform word embedding.
    b. To perform word embedding, the words needs to be tokenized and converted into word2idx(Word to index).
    c. From word2idx, we need to convert it to Word embeddings.
    d. Here we have two options,
        i. Either to use the word2vec approach using Keras Embedding functionality.
        ii. Use GloVe pre-trained word vectors and use it on the embedding layer of keras.
5. Creating Recurrent neural network(RNN using LSTM or GRU):
    a. After creating the word embedding for the JSON objects which has question and answers, we need create a neural network layer.
    b. We will train a LSTM or GRU neural network.
    c. The initial input of the neural network will be the word embedding.
    d. After that we can add more layers like Batch Normalization, Weight regularization, Dropout, etc.
    e. Many different layers can be added to the neural network.
    f. Also add some callback layers like EarlyStopping, Saving latest model, etc.
6. Fitting the model:
    a. After all the above steps, we can fit the model and run it for N epochs.
    b. More the number of epochs will yield better results. This can be executed in a GPU based cloud pc to get faster output.
7. Prediction or chatbot interaction:
    a. Once we fit the model, we can run the chatbot for real-time input.
    b. User or interviewer will interact with the chatbot to understand more about me.
8. Finally connect with the interviewer or user while exiting chatbot. Flask front end app can be deployed and users can interact with that front end screen.

Potential exploration:

1. Gathering data might be one big challenge which I will be having. Correct the input data will need more effort. If the input data is not enough, there might be some irregularities in output.
2. Apart from regular RNN, I can add it to Rasa.ai api which is specifically designed for chatbot. That needs more exploration on rasa.ai api.