**ChatBot Framework for ordering products and checking out:**

An example:

Bot: Hi There! What do you want to purchase from our store?

Customer: I need a phone **or** I want to buy a phone **or** phone etc.

Bot:ok, Phone (Extraction of the products from text using NLP and DATABASE)

Bot: Here are the suggestions for phones

Bot: Phone1, phone2,phone3...phone10. (**Option to click on phone) *Waits for sometime*** Bot: Do you want to look at other phones too?(**Yes/No**) (Waits for no, Suggests more for yes)

Customer: *Clicks on a phone1*.

Bot: The Original price of the phone is, X1, However, after discounts, it cost, X2.

Bot: The price of the phone would be X2

Bot: Do you want to Add this item to cart? (**Yes/No**) (*If yes*)

Bot: Item 'Phone1' Added to cart

Bot: Do you want to do more shopping? (**Yes/No**)

(*If yes*) Bot: What do you want to purchase?

(*If no*) Bot: So, the items in cart are, 'Item1','Item2'...etc

Bot: -1) Do you want to checkout for payment? (**Button**)

-2) Do you want to look at the details of the phone1?(**Button**)

*on Clicking button for 2), redirects to the page for phone*

*on clicking button for 1),*

*(If checkout option is chosen, checkout framework Starts)*

Bot: How do you want to pay for it?

- COD(**Button**)

(*On clicking COD go to Address module*)

- Wallet(**Button**)

Wallet balance: AAAA

+Add money to wallet

(*On clicking wallet, check wallet balance and go to address module*)

- Card

saved cards:

-card1(**Button**)

-card2(**Button**)..

+Add card

(*On clicking card1, card payment procedure and go to address module*)

(Address Module)

Bot: Where should we send your order?

Saved Addresses:

- Address1

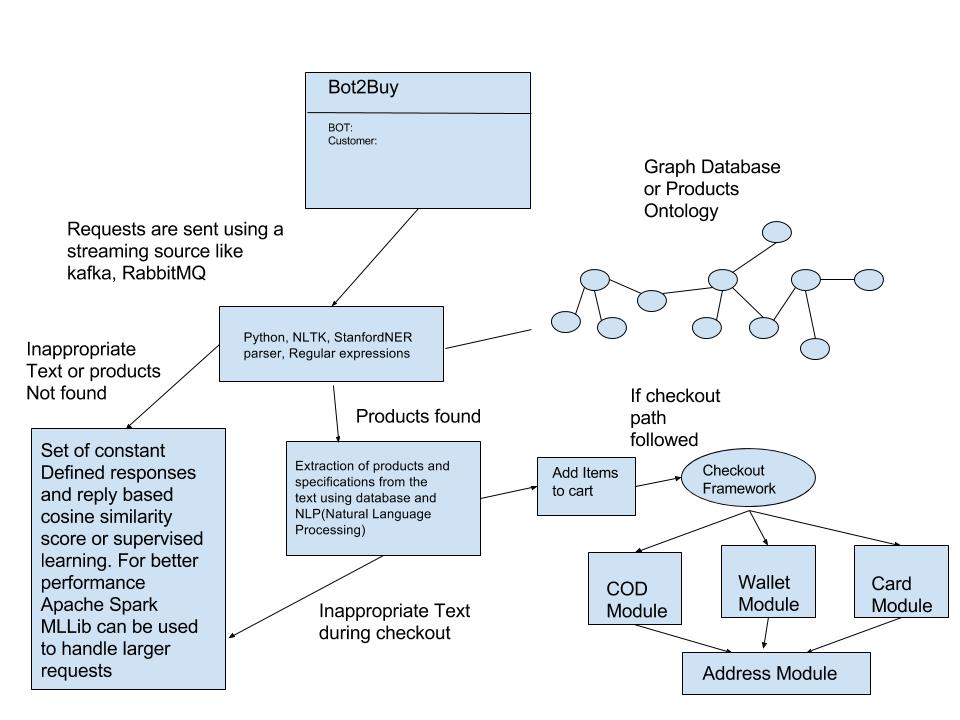
-Address2

+Add new address

Bot: Please Confirm the order. (**Confirm Button**).

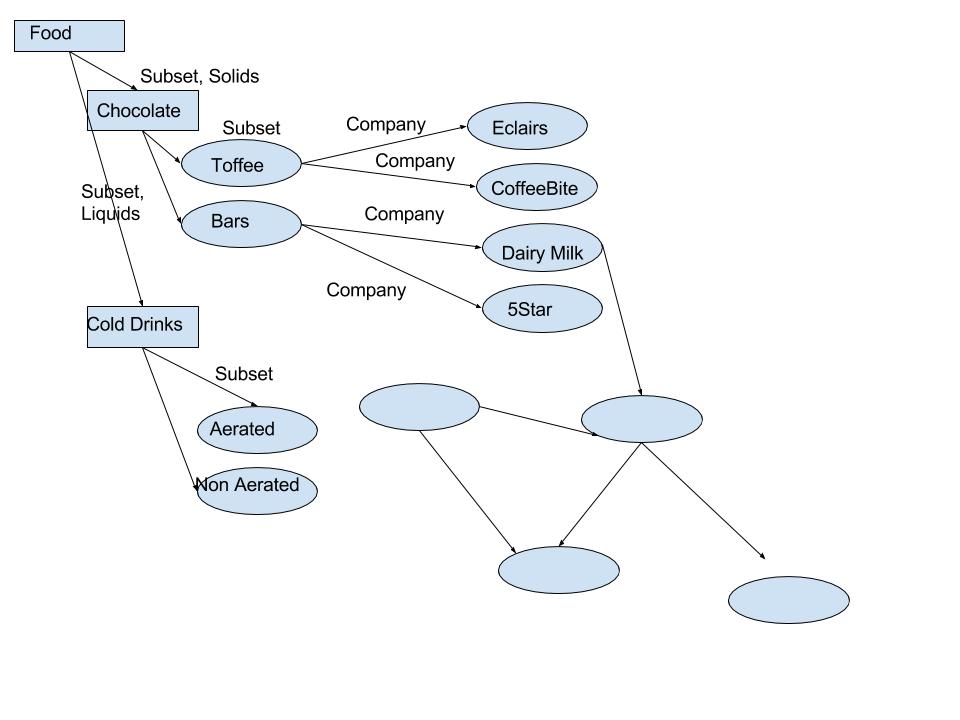
Bot: Thanks for Shopping

**Achitecture diagram:**



Database: We can use products ontology. This means, we can put all the products of the e-commerce site in the database similar to graph database. We can consider ‘PROTEGE’ for this. The advantage with protégé is we can define products and link with other products defining the relation between the products. Similarly, we can also define the relations among various relations in the database. The advantage for this over the other database structures is we can perform scoring for the text better and this will help in recommending products too. We need not worry about creating another recommendation engine separately to recommend products.

So, how does protégé looks like?



This is a small example of how protégé looks like. We can also have options to define what the product is, price etc. Relations can also be defined. Eg: Subset can be defined in a text box saying “Part of a larger group”, Chocolate can be defined as “Food item in the form of paste of solid block form” etc. The advantage is, we can find the product and also related products using the text information which we define in protégé. This data which we store is called as ontology. This ontology can be stored in Apache Jena in RDF (Resource description framework) form. We can use SparQL(query language) to get data from the ontology.

Next part:

Now that our database is ready, when the user poses a query for an item, how do we extract?

First, we’ll use natural language processing techniques to extract all the nouns from text. These nouns are then checked if they match with the items in the database. If that is not the case, we’ll have a scoring module using cosine similarity or any other similarity measure such that if there is any relevance with text, we’ll suggest products based on score. Eg: If someone types, I want to buy a playing item to pass the time. Based on the scores, all the playing items and time pass items will be suggested based on description. Suppose if a Rubik’s cube description in protégé is like, “A puzzle where player tries twist and turn to arrange colors in such a way that each side of the cube has same color. It is a great game for passing the time and sharpening the brain.” This has both time pass and playing item like the user query. Since there is no product found in the query, the system suggests based on description analysis. Based on most related item based on description, we can further use items in close relation with rubik cube like other puzzle items.

If the user is passing time with BOT and not asking questions anything related to buying items, we can use seq2seq modelling using keras, tensorflow backend where train system with various questions and answers or otherwise called as encoders and decoders. Once the machine is trained with enough information, we can use this like normal chat.

In case something offensive is given by user, we can create humorous response along with ignorance module which consist of responses for unrelated queries.

After all the process, the checkout process for payment gateway would be the same as in the website except that, we would be providing buttons in the chat so that user can just click on the buttons to move for the payment. This will be rule based.

For the text analysis and tensorflow part, we could use “Python”.

Summary:

To summarize the architecture, As soon as a query is posed to bot, the text analysis, NLP process analyses from the query to see if the user is asking a question related to buying or not. If yes, is there a product in the text or not that is in the database. If yes, we’ll suggest the product along with the related products based on various recommendation methods. If the user is interested in buying, items are added to the cart and the process restarts. Once the user is done, payment gateway starts and based on checkout process will be completed. During checkout process, the user is only supposed to use buttons. If the user queries anything in between the checkout process, the process starts again. Suppose, if the query is to buy items and the product names are not specifically mentioned, based on the description in the query, similarity scores for the description is used to get the product as explained in rubik cube example. If based on the analysis if the query is not related to buying, we move on to the general texting module where we use seq2seq modelling to respond general queries. However, each time, when the user queries, we’ll first classify if there is anything related to buying the product or not to grab the chance to recommend products.