Smart Phone Recommender

IST 664

Team Member:

Sile Hu

Xiaobin Ning

Henglong So

Ziyun Wang

Contents

1. Introduction

1.1 Research Problem

The main idea of our project is based on the knowledge of NLTK package from lecture to build a human-AI reacting interface leaning on the result of natural language process sentimental analysis. Given the training data, the algorithm is able to predict the sentiment score for specific brand and model.

Our outline of the project can be generally divided into 3 steps:

- Dataset interpreting and data cleansing.
- Sentiment analysis model build and trained based on the extracted and cleansed data.
- Recommendation system frame design, including chatbot interaction dialog followed with query generation, prediction and AI advising based on the result of sentimental analysis score.

1.2 Introduction to the Dataset

We are using the data from Amazon Reviews: Unlocked Mobile Phones. The data was originally proposed by PromptCloud, which contains over 400 thousand reviews of unlocked mobile phones sold on Amazon.com to find out insights with respect to reviews, ratings, price and their relationships.

The columns of data including Product Name(string, The name of the Product), Brand Name(string, Name of the parent company.), Price (number, Price of the product. Max: 2598, Min: 1.73, Mean: 226.86), Rating (number, Rating of the product ranging between 1-5), Reviews (string, Description of the user experience) and Review Votes (number, Number of people voted the review Min: 0, Max: 645, Mean: 1.50).

2. Data Visualization and Description

We did lots of visualization and filters to understand the data. The overall cleansed data was contained at file “data cleansing.ipynb”.

We first look at into the general information of the dataset. There are 65171 unique brand names.

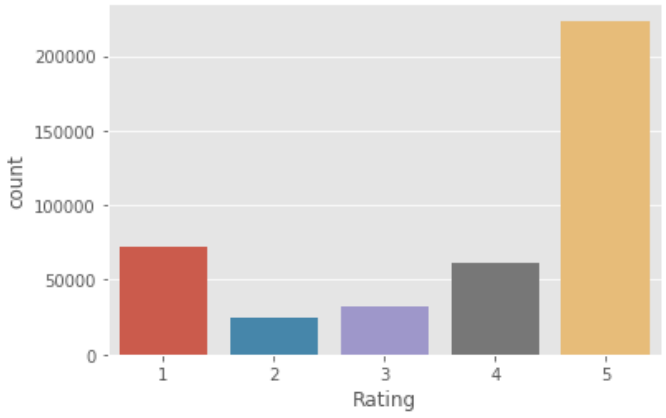
```
In [28]: num = df.isna().sum()
         num

Out[28]: Product Name        0
         Brand Name      65171
         Price            5933
         Rating              0
         Reviews            62
         Review Votes    12296
         dtype: int64
```

Then we look at the price variation. The mean, standard deviation, max, median, 25% ,75% and min are shown below:

```
In [16]: df['Price'].describe()

Out[16]: count    407907.000000
         mean        226.867155
         std         273.006259
         min           1.730000
         25%          79.990000
         50%         144.710000
         75%         269.990000
         max        2598.000000
         Name: Price, dtype: float64
```

Then we took a look at the number of samples for each rating, it appears that the majority of the products are satisfying.

```
In [14]: df = pd.read_csv('Amazon_Unlocked_Mobile.csv')

In [15]: #Labels Exploration
         sns.countplot(df['Rating'])

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2abf3588>
```

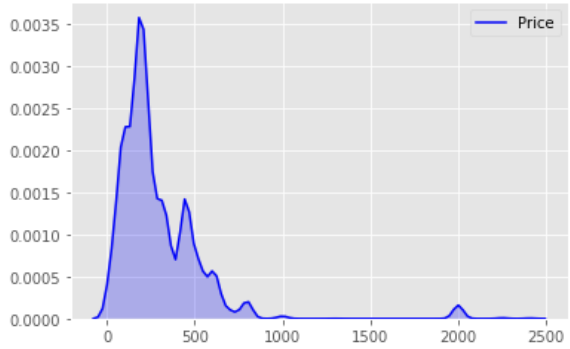


Due to the computing capability and time efficiency and the gravity of data, we only chose product under Samsung brand to be our data. Below are some basic data visualization.

The price variation of Samsung:

```
In [66]: #price for samsung
         sns.kdeplot(df2['Price'], shade=True, color="blue")
```

```
Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0x1a331bfc18>
```



The top 10 Samsang product:

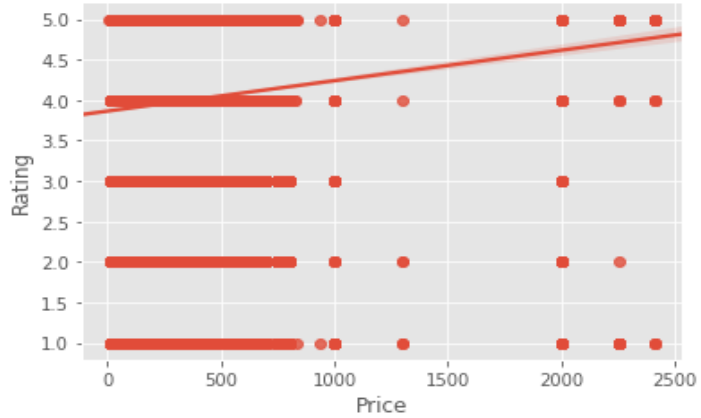
```
#top 10 samsung product
print(df2['Product Name'].value_counts().head(n=10))
```

```
Samsung Galaxy S Duos II S7582 DUAL SIM Factory Unlocked International Version - Black                                1084
Samsung Galaxy S Duos GT-S7562 GSM Unlocked Touchscreen 5MP Camera Smartphone White                                   1059
Samsung Galaxy S4 i9505 16GB LTE Unlocked International Version White                                                 1041
Samsung Galaxy S7 Edge G9350 32GB HK DUAL SIM Factory Unlocked GSM International Version no warranty (BLUE CORAL)     920
Samsung Galaxy Exhibit 4G (T-Mobile), t679                                                                            881
Samsung Galaxy S3 SGH-i747 4G LTE GSM Unlocked 16GB No Warranty (White)                                               873
Samsung Galaxy S5 Mini G800H Unlocked Cellphone, International Version, 16GB, White                                   865
Samsung S5830 Galaxy Ace - Unlocked Phone - Black                                                                     848
Samsung Galaxy Note I717 16GB 4G LTE GSM Android Phone - Carbon Blue (AT&T version)                                   844
Samsung Galaxy S4 SGH-I337 Unlocked GSM Smartphone w/ 13 MP Camera - Red (No Warranty)                                833
Name: Product Name, dtype: int64
```

Correlation between price and review:

```
#correlation price and review
sns.regplot(x='Price',y='Rating',data=df2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a332b16d8>
```



3. Sentiment Classification

3.1 Data Cleansing

Since the dataset contains 413840 rows and 65171 different brands, we choose to focus on Samsung which had 63038 reviews. The size of the review data for this specific brand is large enough for us to do the sentiment classification and gives the customer recommendations. After the first step, we got a new dataframe which contains only the reviews for Samsung. We also deleted all rows which contain NA value.

Then, we also removed the columns called “Rating” and “Review votes”. Because, the rating is for every single review not for each product. We are going to analyze the reviews from customers and generate a new “rating score” for each kind of product by ourselves. Finally, we got a clean dataset which had 64038 rows, 4 columns and 574 different products.

	Product Name	Brand Name	Price	Reviews
0	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	I feel so LUCKY to have found this used (phone...
1	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	nice phone, nice up grade from my pantach revu...
2	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	Very pleased
3	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	It works good but it goes slow sometimes but i...
4	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	Great phone to replace my lost phone. The only...

3.2 Features

We used the movie review sentences from the NLTK corpus to generate features. The movie review sentences are not labeled individually but can be retrieved by category. We created the list of documents where each document(sentence) is paired with its label. Since the documents were in order by label, we mixed them up for later separation into training and test sets. We needed to define the set of words that will be used for features. This is essentially all the words in the entire document collection, except that we will limit it to the 2000 most frequent words. Next, we built our cross-validation function.

```
def cross_validation(num_folds, featuresets):
    subset_size = len(featuresets)//num_folds
    accuracy_list = []
    # iterate over the folds
    for i in range(num_folds):
        test_this_round = featuresets[i*subset_size:][:subset_size]
        train_this_round = featuresets[:i*subset_size]+featuresets[(i+1)*subset_size:]
        # train using train_this_round
        classifier = nltk.NaiveBayesClassifier.train(train_this_round)
        # evaluate against test_this_round and save accuracy
        accuracy_this_round = nltk.classify.accuracy(classifier, test_this_round)
        print(i, accuracy_this_round)
        accuracy_list.append(accuracy_this_round)
        # find mean accuracy over all rounds
        print('mean accuracy', sum(accuracy_list) / num_folds)
```

After all the primary work, we generated BOW features, adding negation features, POS tag features and bigram features and compare the accuracy of these features.

3.2.1 BOW

We can define the features for each document, using just the words, sometimes called the BOW or unigram features. The feature label will be 'V_keyword' for each keyword (aka word) in the word_features set, and the value of the feature will be Boolean, according to whether the word is contained in that document.

```
def document_features(document, word_features):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['V_{}'.format(word)] = (word in document_words)
    return features
```

```
featuresets = [(document_features(d,word_features), c) for (d,c) in documents]
```

3.2.2 Adding Negation

Negation of opinions is an important part of opinion classification. We tired a simple strategy. We looked for negation words "not", "never" and "no" and negation that appeared in contractions of the form "doesn't".

One strategy with negation words is to negate the word following the negation word, while other strategies negate all words up to the next punctuation or use syntax to find the scope of the negation.

The form of some of the words is a verb followed by n't. Now in the Movie Review Corpus itself, the tokenization has these words all split into 3 words, e.g. "couldn", "'", and "t". (and I have a NOT_features definition for this case). But in this sentence_polarity corpus, the tokenization keeps these forms of negation as one word ending in "n't".

```
negationwords = ['no', 'not', 'never', 'none', 'nowhere', 'nothing', 'noone', 'rather',
'hardly', 'scarcely', 'rarely', 'seldom', 'neither', 'nor']
```

Start the feature set with all 2000 word features and 2000 Not word features set to false. If a negation occurs, add the following word as a Not word feature, and otherwise add it as a regular feature word.

```
def NOT_features(document, word_features, negationwords):
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = False
        features['contains(NOT{})'.format(word)] = False
    # go through document words in order
    for i in range(0, len(document)):
        word = document[i]
        if ((i + 1) < len(document)) and ((word in negationwords) or (word.endswith("n't"))):
            i += 1
            features['contains(NOT{})'.format(document[i])] = (document[i] in word_features)
        else:
            features['contains({})'.format(word)] = (word in word_features)
    return features
```

We create feature sets as before, using the NOT_features extraction function.

3.2.3 POS Tag

There are some classification tasks where part-of-speech tag features can have an effect. this is more likely for shorter units of classification, such as sentence level classification or shorter social media such as tweets. The most common way to use POS tagging information is to include counts of various types of word tags. Here is the definition of our new feature function, adding POS tag counts to the word features.

```
def POS_features(document,word_features):
    document_words = set(document)
    tagged_words = nltk.pos_tag(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    numNoun = 0
    numVerb = 0
    numAdj = 0
    numAdverb = 0
    for (word, tag) in tagged_words:
        if tag.startswith('N'): numNoun += 1
        if tag.startswith('V'): numVerb += 1
        if tag.startswith('J'): numAdj += 1
        if tag.startswith('R'): numAdverb += 1
    features['nouns'] = numNoun
    features['verbs'] = numVerb
    features['adjectives'] = numAdj
    features['adverbs'] = numAdverb
    return features
```

3.2.4 Bigram

when we worked on generating bigrams from documents, if we want to use highly frequent bigrams, we need to filter out special characters, which were very frequent in the bigrams, and also filter by frequency. The bigram pmi measure also required some filtering to get frequent and meaningful bigrams.

But there is another bigram association measure that is more often used to filter bigrams for classification features. This is the chi-squared measure, which is another measure of information gain, but which does its own frequency filtering. Another frequently used alternative is to just use frequency, which is the bigram measure raw_freq.

We started by importing the collocations package and creating a short cut variable name for the bigram association measures.

Then, we created a bigram collocation finder using the original movie review words, since the bigram finder must have the words in order. Note that our all_words_list has exactly this list.

We used the chi-squared measure to get bigrams that are informative features. Because we didn't need to get the scores of the bigrams, we used the nbest function which just returns the highest scoring bigrams, using the number specified.

The nbest function returned a list of significant bigrams in this corpus.

At last, we created a feature extraction function that had all the word features as before, but also had bigram features.

```
def bigram_document_features(document, word_features, bigram_features):
    document_words = set(document)
    document_bigrams = nltk.bigrams(document)
    features = {}
    for word in word_features:
        features['V_{}'.format(word)] = (word in document_words)
    for bigram in bigram_features:
        features['B_{}_{}'.format(bigram[0], bigram[1])] = (bigram in document_bigrams)
    return features
```

```
bigram_featuresets = [(bigram_document_features(d, word_features, bigram_features), c) for (d,c) in documents]
```

3.3 Classification

The model that we used to do the classification is Naïve Bayes. After creating those features, we run them with the classifier separately. First, we split each feature into training and test and run the classifier and acquire the accuracy. Then, we do the cross-validation and obtain the accuracy again for each feature. As we can see, the adding negation features brought the highest accuracy in the model. In this way, we chose to use it in order to get the sentiment score for each review.

```
train_set, test_set = NOT_featuresets[200:], NOT_featuresets[:200]
classifier1 = nltk.NaiveBayesClassifier.train(train_set)
print(nltk.classify.accuracy(classifier1, test_set))
classifier1.show_most_informative_features(30)
```

```
0.815
```

```
cross_validation(10, NOT_featuresets)
```

```
0 0.7795497185741088
mean accuracy 0.07795497185741088
1 0.7814258911819888
mean accuracy 0.15609756097560976
```

Features	Accuracy	Accuracy (after cross validation)
BOW	0.78	0.75
Adding Negation	0.815	0.78
POS Tag	0.774	0.75
Bigram	0.78	0.75

3.4 Sentiment Analysis

After importing the packages that we need, we pre-processed our data. For data preprocessing, we need to tokenize our text. For our dataset, the review text is stored in rows of a dataframe, so we can just extract them from the dataframe and store it into a list.

After data preprocessing, we used the model with highest accuracy to find out the results.

```
label = []
for i in ReviewList:
    texttokens = nltk.word_tokenize(i)
    textlower = [w.lower() for w in texttokens]
    inputfeatureset = NOT_features(textlower, word_features, negationwords)
    a = classifier1.classify(inputfeatureset)
    label.append(a)
```

```
def merge(list1, list2):
    merged_list = [(list1[i], list2[i]) for i in range(0, len(list1))]
    return merged_list
```

```
sentimentList = merge(ReviewList, label)
```

```
sentiment_neg = []
sentiment_pos = []
Sentiment = []
for i in sentimentList:
    if i[1] == 'neg':
        sentiment_neg.append(i[0])
        Sentiment.append(int(0))
    else:
        sentiment_pos.append(i[0])
        Sentiment.append(int(1))
```

Finally, we obtained 44943 negative reviews and 18095 positive reviews. We added the “sentiment score” after each review and got a new dataframe.

Index	Product Name	Brand Name	Price	Reviews	Sentiment
0	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	I feel so LUCKY to have found this used (phone...	0
1	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	nice phone, nice up grade from my pantach revu...	1
2	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	Very pleased	1
3	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	It works good but it goes slow sometimes but i...	0
4	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	Samsung	199.99	Great phone to replace my lost phone. The only...	1

3.5 Percentage of Positive Reviews

We want to understand the percentage of positive reviews for each product, because this is one of the most important information that the customers want to know before purchasing the items. Since we defined positive review as 1 and negative review as 0, the calculation of percentage of positive reviews is the same with the calculation of the mean value. It is easy for us to get a new dataframe which contains the product name and their percentage of positive reviews.

	Product Name	Positive Percentage
0	"CLEAR CLEAN ESN" Sprint EPIC 4G Galaxy SPH-D7...	0.270270
1	Cricket Samsung Galaxy Discover R740 Phone	0.185185
2	Galaxy s III mini SM-G730V Verizon Cell Phone ...	0.000000
3	Galaxy S5 G900A Factory Unlocked Android Smart...	0.297500
4	Galaxy S6 Active - Camo Blue (Unlocked)	0.000000

4. Chatbot

After we have done all the data cleaning and sentiment analysis, we are trying to maximin our effort by created a chatbot. In the world of machine learning and AI there are many kinds of chat bots. Some chat bots are virtual assistant, others are just there to talk to, some are customer support agents and for this project we will work with some business questions. It can communicate with the users and generated the response back to the user. This chatbot will also provide the sentiment analysis percentage score of each review once the user enters the product type they are looking for.

First, we need to create a .Json file that contains some intents, patterns, and responses to train our chatbot. We are doing with the JSON file is creating a bunch of messages that the user is likely to

type in and mapping them to a group of appropriate responses. The tags on each dictionary in the file indicates the group that each message belongs to.

```
{"intents": [
        {"tag": "greeting",
         "patterns": ["Hi", "How are you", "Is anyone there?", "Hello", "Good day", "Whats up"],
         "responses": ["Hello!", "Good to see you again!", "Hi there, how can I help?"],
         "context_set": ""
        },
        {"tag": "goodbye",
         "patterns": ["cya", "See you later", "Goodbye", "I am Leaving", "Have a Good day"],
         "responses": ["Sad to see you go :(", "Talk to you later", "Goodbye!"],
         "context_set": ""
        },
        {"tag": "age",
         "patterns": ["how old", "how old is tim", "what is your age", "how old are you", "age?"],
         "responses": ["I am 18 years old!", "18 years young!"],
         "context_set": ""
        },
        {"tag": "name",
         "patterns": ["what is your name", "what should I call you", "whats your name?"],
         "responses": ["You can call me Bot for IST 664.", "I'm Bot for IST 664! your personal assistant", "I'm Bot for IST 664 aka Tech With IST 664."],
         "context_set": ""
        },
        {"tag": "phone",
         "patterns": ["Id like to buy phone", "what's the best phone", "what do you reccommend for a phone?","do you have any phone to recommend","I want to buy a phone
         "responses": ["great! we can definately could help you with that. what phone are you lookin for ", "what kind of phone are you looking for"],
         "context_set": ""
        },
        {"tag": "brand",
         "patterns": ["samsung", "i am thinking about samsung", "what do you think about samsung"],
         "responses": ["Samsung is a great brand what is your price limit ", "we have a good samsung phones to recommend you. May i know your budget?"],
         "context_set": ""
        },
        {"tag": "recommend",
         "patterns": ["what is the best phone for samsung under 500", "give me the best phone under 500", "under 500"],
         "responses": ["We are a few suggesting for you: for under 500 Samsung SGH-A237 Blue No Contract AT&T Cell Phone is the best phone"],
         "context_set": ""
        },
```

Next, we are importing the package to create our chatbot. The package included: nltk, numpy, tflearn, tensorflow, random and json.

In [1]:
```
import nltk
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()
#nltk.download('punkt')
import numpy
import tflearn
import tensorflow
import random

import json
with open('intents.json') as file:
    data = json.load(file)
```

After we imported all the packages, now we need to extract the data we want from JSON file. We need all the pattterns and which class tag they belong to. We also want a list of all the unique words in our patterns. We will also performs a looping through our JSON data we want. For each pattern we will turn into a list of words using *nltk.word_tokenizer,* rather than having them as a strings. We will then add each pattern into our docs_x list and its associcated tag into the dosc_y list.

Extracting Data

In [2]:
```
words = []
labels = []
docs_x = []
docs_y = []
```

In [3]:
```
for intent in data['intents']:
    for pattern in intent['patterns']:
        wrds = nltk.word_tokenize(pattern)
        words.extend(wrds)
        docs_x.append(wrds)
        docs_y.append(intent["tag"])

    if intent['tag'] not in labels:
        labels.append(intent['tag'])
```

Word stemming: for this step, we are using word stemming to attemps to find the root of the words.

Word Stemming

```
In [4]: words = [stemmer.stem(w.lower()) for w in words if w != "?"]
        words = sorted(list(set(words)))

        labels = sorted(labels)
```

Bag of words: In this step, we are performing the bag of worlds. It will loaded the data and create a stemmed vocabulary. We need some way to represent our sentences with number and this is where a bag of words comes in. this is represent each sentence with a list length of the amount of words in our models vocabulary. Each position in the listz will represent a word from our vocabulary of the position in the list is a 1 then that will mean that the word exsits in our sentence, if it is a 0 then the word is nor present. We call this a bag og words because the order in which the words appear in the sentence is lost, we only know the presence of words in o[ur models vocabulary.

```
In [5]: training = []
        output = []

        out_empty = [0 for _ in range(len(labels))]

        for x, doc in enumerate(docs_x):
            bag = []

            wrds = [stemmer.stem(w.lower()) for w in doc]

            for w in words:
                if w in wrds:
                    bag.append(1)
                else:
                    bag.append(0)

            output_row = out_empty[:]
            output_row[labels.index(docs_y[x])] = 1

            training.append(bag)
            output.append(output_row)
```

```
In [6]: training = numpy.array(training)
        output = numpy.array(output)
```

Finally we will convert our training data output into numpy arrays.

Developing the model: Now that we have preprocessed all of our data we are ready to start creating and training a model. For our purposes we will use a fairly standard feed-forward neural network with two hidden layers. The goal of our network will be to look at a bag of words and give a class that they belong too (one of our tags from the JSON file). After the process we will train and saving the method.

In [7]:
```
tensorflow.reset_default_graph()

net = tflearn.input_data(shape=[None, len(training[0])])
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, len(output[0]), activation="softmax")
net = tflearn.regression(net)

model = tflearn.DNN(net)
```

```
WARNING:tensorflow:From /anaconda3/lib/python3.6/site-packages/tflearn/objectives.py:66: calling reduce_sum_v1 (from
tensorflow.python.ops.math_ops) with keep_dims is deprecated and will be removed in a future version.
Instructions for updating:
keep_dims is deprecated, use keepdims instead
WARNING:tensorflow:From /anaconda3/lib/python3.6/site-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from
tensorflow.python.ops.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
```

In [8]:
```
model.fit(training, output, n_epoch=1000, batch_size=8, show_metric=True)
model.save("model.tflearn")
```

```
Training Step: 4999  | total loss: 0.00692 | time: 0.010s
| Adam | epoch: 1000 | loss: 0.00692 - acc: 0.9999 -- iter: 32/38
Training Step: 5000  | total loss: 0.00672 | time: 0.011s
| Adam | epoch: 1000 | loss: 0.00672 - acc: 0.9999 -- iter: 38/38
--
INFO:tensorflow:/Users/henglong/Documents/IST 664 - NLP/project/model.tflearn is not in all_model_checkpoint_paths. M
anually adding it.
```

Prediction: last but not least, we are going to create a new function to predict our model. Ideally we want to generate a response to any sentence the user types in. to do this we need to remember our model doesn't not take string input, it takes a bag of words.

In [10]:
```
def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]

    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_words]

    for se in s_words:
        for i, w in enumerate(words):
            if w == se:
                bag[i] = 1

    return numpy.array(bag)


def chat():
    print("Start talking with the bot (type quit to stop)!")
    while True:
        inp = input("You: ")
        if inp.lower() == "quit":
            break

        results = model.predict([bag_of_words(inp, words)])
        results_index = numpy.argmax(results)
        tag = labels[results_index]

        for tg in data["intents"]:
            if tg['tag'] == tag:
                responses = tg['responses']

        print(random.choice(responses))

chat()
```

Final step, the following step is the demo of the chatbot that we have created we now understand that our chatbot is responded to the user input and giving the phone recommendation for the user.

```
        print(random.choice(responses))

chat()
```

```
Start talking with the bot (type quit to stop)!
You: what is your name
I'm Bot for IST 664 aka Tech With IST 664.
You: how old are you
I am 18 years old!
You: i want to buy a phone
what kind of phone are you looking for
You: samsung
Samsung is a great brand what is your price limit
You: under 500
We are a few suggesting for you: for under 500 Samsung SGH-A237 Blue No Contract AT&T Cell Phone is the best phone
You: i want to see score
With our sentiment analysis with all review for this phone is: 66% positive. the most positive words are: love, enjo
y, cool, work great.
You: i want to see the phone
here is the link for your pone that you are interesting in: https://www.amazon.com/s?k=Samsung+SGH-A237+Blue+No+Contr
act+AT%26T+Cell+Phone&ref=nb_sb_noss
You: bye
Good to see you again!
You: quit
```

Reference:

Amazon Unlocked Phone Reviews Dataset from
https://www.kaggle.com/PromptCloudHQ/amazon-reviews-unlocked-mobile-phones

Tech With Tim, 2019. Chatbot tutorial series: https://www.youtube.com/watch?v=wypVcNIH6D4

Edureka!, 2017. Azure tutorial series: https://www.youtube.com/watch?v=0bPJPiX89K0