**Project Title: Opinion Mining of Amazon Fine Food (“Cookie”) Review**

**Team Member:**

Kai Chung, Ying (Online Student)

**Project Description: Goals/ Motivation/ Importance**

This classification project employs the techniques of the opinion mining to analyze the Fine Food Reviews available in Kaggle.com. The concept of this project is to build a Review monitoring system to manage the accuracy of the posted review on any review-based marketplace. As we know, nowadays customers depend on reviews heavily to pick their targets (i.e. products, restaurants or other business). So that the accuracy of reviews is one of essential factors impacting on their success. Ecommerce giant Amazon has already built a system to measure the accuracy of reviews in order to provide customers and business owner to trade in a fair ground. Our targets are as the followings:

* With the given dataset, would the designed grammar patterns capture the imperative part of the reviews?
* Would our technique / models be able to predict the sentiment class (I.e. Positive=1 and Negative=0) correctly for the extracted review segments?
* Which classifiers/ models be able to efficiently determine the highest accuracy of this classification experiment?

**Proposed Methodology and Techniques**

This project is entirely done by Python 3.6 version and the code is implemented in Python notebook environment.

**Step1.** This project starts with downloading the original dataset from Kaggle.com. Prior to the core analysis, preliminary setup steps were first accomplished (i.e. Importing libraries, cleaning missing values, Correcting data type, Tokenizing Sentences and Words etc.). The imported libraries are shown below:

* import gensim
* from gensim.models import Word2Vec, Phrases
* from gensim.models.phrases import Phraser
* from nltk.corpus import PlaintextCorpusReader
* import nltk
* import pandas as pd
* import numpy as np
* import time
* from nltk.tokenize import sent\_tokenize, word\_tokenize
* from collections import Counter

**Step2:** In order to get focus on extracting related content from the review, this experiment will only focus on one type of product which is “cookie”. So that the word2Vec function is exploited to extract features which are related to “Cookies”, “cookies”, “cookie” or “Cookie” in this case. Features will be adjusted depending on the final Accuracy of the classification, number of segments be able to extracted and other metrics factors.

Code:

#Train the Word2Vec Model with 100 features

model = Word2Vec(sent\_train)

#Initialize the list of feature

feature\_word\_lst= []

for word in ["cookies","Cookies", "cookie", "Cookie"]: #Review Topics

word\_lst = list(model.most\_similar(word, topn=100)) #Find top 100 most similar words

for i in range(len(word\_lst)):

#print(word\_lst[i][0])

feature\_word\_lst.append(word\_lst[i][0]) #Save similar to a list: feature\_lst

#Obtain the feature word list

feature\_word\_lst = list(set(feature\_word\_lst)) #Get the Union of all the feature words

**Step3.)** Regular Expression Pattern will then be constructed accordingly. With the completion of Pattern structure, then parsing the reviews by exploiting the NLTK Regex Chunk parser to extract segments are associated with the features from the Training set of data. The followings are the designed patterns to capture the review segments.

#regex grammar

grammar = r"""

JN:{<RB.\*|JJ.\*>+<.>\*}

NP:{<EX|DT|PRP\$>?<RB>\*<JN|VB.\*>+<NN.\*>+<.>\*}

cla1a:{<DT|PRP>?<NN.\*>+<VB.\*>+<JN>+<JJ.\*>\*<.>\*}

cla1b:{<cla1a>+<,|CC>+<JN|NN.\*>+<JN|NN.\*>\*<.>\*}

{<cla1a>+<,|CC>+<cla1a>+<.>\*}

cla2a:{<DT>?<NN.\*|PRP.\*>+<IN|VB.\*>+<DT>\*<NP|NN.\*>+<NP|NN.\*|JN>\*<.>\*}

cla2b:{<DT>?<cla2a>+<,|CC>+<DT>\*<JN|NN.\*|NP>+<.>\*}

{<DT>?<cla2a>+<,|CC>+<cla2a>+<.>\*}

cla3: {<DT>?<NN.\*>+<NP>+<JN>\*<.>\*<IN>\*<NP>\*}

cla4: {<cla1a>+<VB.\*>+<JN>\*<.>\*}

cla4b: {<cla4>+<PRP.\*>+<NN.\*>+<TO>\*<VB.\*>\*<.>\*}

cla5: {<NN.\*|PRP.\*><NP>+<IN>+<NP>+<.>\*}

{<NN.\*|PRP.\*><NP>+<IN>+<NP>\*<.>\*}

cla6: {<PRP>\*<MD>\*<JN>\*<VB.\*>+<DT>+<TO>+<NP>+<.>\*}

cla7: {<WP>+<PRP>\*<MD>+<JN>\*<VB.\*>+<PRP>+<JN>+<.>\*}

cla8: {<PRP>?<MD>\*<JN>\*<VB.\*>+<DT>+<TO>+<PRP\$>+<NN.\*>+<.>\*}

cla9: {<DT>\*<NN>+<MD>\*<VB.\*>+<DT>+<NN>+<JN>+<IN>+<CD>\*<NN.\*>\*<.>\*}

cla10: {<PRP>+<VB.\*>+<JN>+<NP>+<IN>+<NP+>\*<.>\*}

cla11: {<CC>+<DT>\*<NN.\*>?<JN>+<VB.\*>+<IN>+<PRP.\*>\*<NN>\*<.>\*}

cla12: {<CC>?<VB.\*>+<IN>+<PRP.\*>+<.>\*}

cla13: {<CC>?<PRP.\*>+<VB.\*>+<JN>+<IN>+<DT|VB.\*>\*<JJ.\*|JN|NN.\*>\*<.>\*}

cla14:{<CC>?<PRP.\*>\*<MD>\*<JN>\*<VB.\*><TO|IN>+<VB.\*|JN>+<.>\*}

cla15: {<CC>?<NN.\*|PRP>+<VB.\*>+<VB.\*|JN>+<PRP.\*|DT|WDT|NN>\*<VB.\*>\*<.>\*}

cla15b: {<cla15>+<CC>+<JN|JJ>+<.>\*}

cla15c: {<cla15>+<TO>+<VB.\*>+}

cla16:{<CC>?<DT>+<VB.\*>\*<JN|JJ.\*|NP>+<JN|VB.\*>\*<.>\*}

cla16b:{<cla16><,|CC>+<NN.\*|NP>\*<.>\*}

cla16c:{<cla16b>+<PRP.\*>+<JN>\*<VB.\*>+<.>\*}

cla17: {<NP>+<WDT|WP.|WRB>\*<PRP.\*>\*<VB.\*>+<NP|JN>\*<.>\*}

cla17b: {<cla17>+<TO|VBG>+<VB.\*>\*<JN>\*<.>\*}

cla17c:{<cla17>+<PRP.\*>+<JN | JJ>\*}

cla18:{<DT>?<CD>+<IN>\*<JN|NN.\*>\*}

cla18a:{<cla18>+<,|CC>+<cla18>+<.>\*}

cla19: {<PRP.\*>\*<JN>\*<WP>?<MD>\*<VB.\*>+<DT|PRP.\*>\*<NP|NN.\*|PRP.\*>+<.>\*}

cla19b: {<cla18>+<cla19>+<.>\*}

cla20: {<cla15>+<cla12>+<.>\*}

cla21: {<PRP>\*<MD>\*<JN>+<TO>\*<VB>+<DT>\*<cla16|cla18>\*}

cla22: {<VB.\*>+<cla2a>+<WRB|WP|WDT>\*<PRP>\*<VB.\*>\*<.>\*}

cla23:{<cla2a>+<WDT|WP>+<cla19>+}

cla24: {<NNP>+<JN>+<IN>+<DT>+<PRP>\*<:|.>\*}

cla24b:{<cla24>+<cla17>+}

cla25: {<NN.\*>+<TO|PRP.\*>\*<NN.\*>+<,|IN|CC>\*<NN.\*>\*<IN|TO>\*<VB.\*>+<PRP.\*|DT>\*<:|.>\*}

cla25b:{<cla25>+<,|CC>\*<JN|NP>+<.|:>\*}

cla26: {<PRP.\*>+<NN.\*>+<,|IN|CC>\*<NN.\*>\*<VB.\*>+<DT|PRP.\*>\*<JN>\*<:|.>\*}

cla27: {<cla15>+<IN>\*<WRB|WDT|WP.\*>\*<JN>+<DT|NN.\*>\*<VB.\*>+<:|.>\*}

cla28:{<NP|NNP>+<,|CC>\*<NP|JN>+}

cla29: {<WP>+<JN>\*<PRP.\*>+<MD>+<VB.\*>\*<:|.>\*}

cla30: {<WP>+<JN>\*<VB.\*>+<cla21>+<:|.>\*}

cla31: {<NN.\*>+<,|CC|IN>\*<JN|JJ|NN.\*>\*<,|CC|IN>\*<NN.\*>\*<:|.>\*}

cla32: {<NN.\*|PRP.\*>\*<MD>+<VB.\*>\*<PRP.\*|DT>+<JN>\*<.|:>\*}

cla33:{<cla19>+<DT|JN | RP>\*<.|:>\*}

cla34:{<PRP.\*>+<cla18>+<cla3>\*<cla19>+<JN>\*<.|:>\*}

cla35:{<DT>\*<JJS>+<,|CC>\*<JJS>\*<NN.\*><PRP.\*|DT>+<JN>\*<.|:>\*}

cla36:{<cla2a>+<IN>\*<DT>\*<NN.\*|NP|cla31>+<.|:>\*}

cla37:{<DT|PRP.\*>\*<cla15>+<CC|IN|,>\*<cla15>+<.|:>\*}

cla38:{<DT>+<MD>+<VB.\*>+<IN|DT>\*<cla31>+<.|:>\*}

cla39:{<NN.\*|PRP.\*>+<VB.\*>\*<PRP.\*>+<JN>\*<.|:>\*}

cla40:{<NN.\*|PRP.\*>\*<MD>\*<VB.\*>+<JN|cla16>+<.|:>\*}

cla41:{<cla33>+<TO|IN>+<cla2a>+<.|:>\*}

cla42:{<cla31>+<TO|IN>+<VB.\*>+<TO|IN>+<NP>+<.|:>\*}

cla43:{<DT>\*<cla31>+<cla40>+<TO|IN>+<DT|PRP.\*>\*<cla31>\*<.|:>\*}

cla44:{<WP>+<cla40>+<IN>\*<DT|PRP.\*>+<NP|NN.\*>+<.|:>\*}

cla44b:{<WP>+<cla40>+<IN>+<cla18|DT|NN.\*|cla31>+<.|:>\*}

cla45:{<cla2b>+<cla40>+<.|:>\*}

cla46:{<cla13>+<NP>+<.|:>\*}

cla47:{<WRB>+<JN|VB.\*>+<CC|IN>\*<JN|VB.\*>\*<DT|PRP.\*>\*<VB.\*>+<.|:>\*}

cla48:{<cla17>+<IN>+<cla2a>+<.|:>\*}

cla49:{<cla19>+<IN>+<PRP.\*>\*<cla18|cla31>+<.|:>\*}

cla49b: {<cla49>+<cla3>+<.|:>\*}

cla50: {<PRP.\*>?<cla12>+<cla31>+<.|:>\*}

cla51:{<CC>\*<DT|PRP.\*>\*<cla31>+<cla19+>+<JN>\*<.|:>\*}

cla52:{<cla2a>+<MD>\*<VB.\*>+<WRB>\*<JN>+<DT>\*<PRP.\*|cla31>\*<VB.\*>\*<.|:>\*}

cla53:{<cla31>+<cla39|VG.\*>\*<DT|cla31>\*<cla33>+<.|:>\*}

**The following examples in Table 1 are part of segments are expected to be extracted by the above constructed patterns with using the nltk function nltk.RegexpParser(grammar):**

*Table1:*

|  |  |
| --- | --- |
| cla1a | #sent ="the cookies are not very good enough"  #sent ="the cookies are not very good"  #sent = "this cookie was surprisingly yummy." |
| cla1b: | #sent = "cookies was tasty and soft"  #sent = " the cookies is pretty beautiful and the cookies is pretty soft"  #sent = " the cookies is big and the raisins is very soft" |
| Cla2a | #sent="I love the food so much"  #sent = "I participated in a product review" |
| cla2b | #sent = "She likes this big cookie and soft cookie"  #sent = "She likes these cookies and the smell" |
| cla3 | #sent = "the cookies are way too small"  #sent = "The cookies are soft, chewy, yummy,"  #sent = "These cookies have perfect flavor, excellent moist/dry combination" |
| cla4 | #sent="the cookie did not smell good" |
| 4b | #sent="please don't waste your money " |
| cla5 | #sent = "I love cookies with many rasins" |
| cla7 | #sent= "who would buy it again??" |
| cla13 | #sent = "It goes down hill from there."  #sent = "They're are so soft, perfect for anytime" |
| cla14 | #sent = "I can't wait to try "  #sent="can't wait to buy more" |
| cla15 | #sent="It was great!!!"  #sent = "i love eating it"  #sent = "They're are so soft and perfect" |
| cla15b | #sent="They are so soft and tasty!" |
| cla15c | #sent = "it is great to buy it" |
| cla16 | #sent = "This is my great best experience!"  #sent = "This is the high point."  #sent = "This was the most delightful cookie i ever tasted." |
| cla16b | #sent = "the best and freshest cookie" |
| cla16c | #sent = "the best and freshest cookie I ever had" |
| cla#17 | #sent = "this is bad food which hurt your healthy body"  #sent = "The high quality ingredients are a crowd pleaser." |
| cla17c | #sent = "Oatmeal cookies are my favorite" |
| cla19 | #sent="what's the point?"  #sent="you'll love this snack ....."  #sent = "I highly recommend this cookie" |
| cla21 | #sent="I would never try this one again." |
| cla22 | #sent="throw them in the garbage which they belong" |
| cla23 | #sent="I participated in a product review that included a sample" |
| cla26 | #sent="My kids and i loved these." |
| cla27 | #sent="I was surprised how soft the cookie was."  #sent="I was immediately surprised by how soft the cookie was " |
| cla#28 | #sent = “Delicious and nutritious …..”  #sent = “good price and delicious” |
| cla29 | #sent="what else you can say!!!" |
| cla30 | #sent="what else is there to say...." |
| cla31 | #sent = "Soft, chewy and tasty."  #sent = "Highly recommend!"  #sent = "yummy and delicious." |
| cla33 | #sent = "we ate them all...!!!" |
| cla34 | #sent="My three year old kid loved it so much !!!" |
| cla36 | #sent = "I am a huge fan of the Quaker Chewy cookies!" |
| cla37 | #sent="The moist cookies are satisfying and they are not too sweet." |
| cla40 | #sent ="I will buy some more" |
| cla41 | #sent = "I recommend these to anyone with a sweet tooth" |
| cla42 | #sent = "Great to take as a quick breakfast " |
| cla43 | #sent = "This cookie would be perfect for breakfast"  #sent = "This cookie would be perfect for my family..." |
| cla44 | #sent = "What could be better than that?!" |
| cla44b | #sent = “What could be better than this one ?!”  #sent = “What could be better than this product ?!” |
| cla46 | #sent = “they are more healthy than most other products.”  #sent=”they are more healther than other food” |
| cla47 | #sent = "how soft and fresh it tasted"  #sent="How good it tasted?" |
| cla49 | #sent="I even shared it with my brother"  #sent="I shared it with my child" |
| cla50 | #sent = "They taste like my grandmas homeade cookies.." |
| cla51 | #sent = "the kids love them!"  #sent ="My kid loved it so much " |
| cla52 | #sent = "I guess the younger folks will like how sweet the cookie is." |
| cla53 | #sent="Thank you my mom buying me this!!!" |

**Step4.** With all the parsed review segments for every record, 2-step content filtering is performed before mining Features for building the classifiers. This step would be able to filter out segments without sentiment meaning which might not be appropriate to express accurate Positive or Negative sentiment.

* **Filtering out all the segments without tagging “Adverb” or “Adjective” or “Verb”**
* **Filtering out Segments not including Feature words which were calculated by using Word2Vec in Step 2**

**Step5.** There are TWO Phases of features implementation for the classification. The followings are the Features are compiled from the extracted segments for the classification process. The Phase-1 twenty-eight Features in Table2 were first implemented with the classifiers algorithms as our baseline models. With that, the Phase-2 thirty Features in Table 3 will then be added on top of original dataset and passed to the classifiers for metrics re-evaluation and models comparisons. By doing that, we would be able to distinguish how the two set of features influence the accuracy of the models.

*Table2:*

|  |  |  |
| --- | --- | --- |
| **Phase 1** | | |
| **#** | **Features** | **Description** |
| 1 | repeat\_word\_cnt | Count of word "repeat\_word in consecutive order" |
| 2 | no\_cnt | Count of word "no" |
| 3 | never\_cnt | Count of word "never" |
| 4 | quite\_cnt | Count of word "quite" |
| 5 | but\_cnt | Count of word "but" |
| 6 | very\_cnt | Count of word "very" |
| 7 | more\_cnt | Count of word "more" |
| 8 | br\_cnt | Count of word "br" |
| 9 | else\_cnt | Count of word "else" |
| 10 | must\_cnt | Count of word "must" |
| 11 | many\_cnt | Count of word "many" |
| 12 | up\_cnt | Count of word "up" |
| 13 | so\_cnt | Count of word "so" |
| 14 | all\_cnt | Count of word "all" |
| 15 | dots\_cnt | Count of word "..." |
| 16 | exclamation\_mark\_cnt | Count of word "exclamation\_mark" |
| 17 | question\_mark\_cnt | Count of word "question\_mark" |
| 18 | not\_cnt | Count of word "not" |
| 19 | er\_cnt | Count of word ending with "er" |
| 20 | est\_cnt | Count of word ending with "est" |
| 21 | ous\_cnt | Count of word ending with "ous" |
| 22 | ly\_cnt | Count of word ending with "ly" |
| 23 | dy\_cnt | Count of word ending with "dy" |
| 24 | ful\_cnt | Count of word ending with "ful" |
| 25 | ed\_cnt | Count of word ending with "ed" |
| 26 | less\_cnt | Count of word ending with "less" |
| 27 | Polarity Score 1 | Evaluated by swn.senti\_synset() function from the first synset element |
| 28 | Polarity Score 2 | Averaging the Polarity Scores evaluated by swn.senti\_synset() function from ALL synset elements |
|  | *Table3:* |  |
| **Phase 2** | | |
| **#** | **Features** | **Description** |
| 29 | CC\_cnt | Count of word tag "CC" |
| 30 | CD\_cnt | Count of word tag "CD" |
| 31 | DT\_cnt | Count of word tag "DT" |
| 32 | IN\_cnt | Count of word tag "IN" |
| 33 | JJ\_cnt | Count of word tag "JJ" |
| 34 | JJR\_cnt | Count of word tag "JJR" |
| 35 | JJS\_cnt | Count of word tag "JJS" |
| 36 | MD\_cnt | Count of word tag "MD" |
| 37 | NN\_cnt | Count of word tag "NN" |
| 38 | NNS\_cnt | Count of word tag "NNS" |
| 39 | NNP\_cnt | Count of word tag "NNP" |
| 40 | NNPS\_cnt | Count of word tag "NNPS" |
| 41 | POS\_cnt | Count of word tag "POS" |
| 42 | PDT\_cnt | Count of word tag "PDT" |
| 43 | PRP\_cnt | Count of word tag "PRP" |
| 44 | PRP\_dol\_cnt | Count of word tag "PRP$" |
| 45 | RB\_cnt | Count of word tag "RB" |
| 46 | RBR\_cnt | Count of word tag "RBR" |
| 47 | RBS\_cnt | Count of word tag "RBS" |
| 48 | TO\_cnt | Count of word tag "TO" |
| 49 | VB\_cnt | Count of word tag "VB" |
| 50 | VBD\_cnt | Count of word tag "VBD" |
| 51 | VBG\_cnt | Count of word tag "VBG" |
| 52 | VBN\_cnt | Count of word tag "VBN" |
| 53 | VBP\_cnt | Count of word tag "VBP" |
| 54 | VBZ\_cnt | Count of word tag "VBZ" |
| 55 | WDT\_cnt | Count of word tag "WDT" |
| 56 | WP\_cnt | Count of word tag "WP" |
| 57 | WP\_dol\_cnt | Count of word tag "WP$" |
| 58 | WRB\_cnt | Count of word tag "WRB" |

**Step6: Phase I Classification**

In the Phase I classification process, FIVE classifiers with 28 features were attempted to be implemented to compare their accuracy metrics. The five classifiers are GaussianNB, BernoulliNB, Gradient Boosting, SVM Linear, K-Nearest Neighbors and Random Forest. The capability to classify Positive sentiment reviews is our key motivation of this review classification experiment. So that we would focus more on the Overall Accuracy, Recall (i.e. Sensitivity) and Precision measurement metrics. As we can see the Table 4 below, the Accuracy score of Testing set among the classifiers are very closed to each other (i.e. 69%-77%) except the Gaussian Naïve Bayes algorithm (i.e. <60%). And when we come down to the Sensitivity score, the scores of the classifiers (i.e. 80%-88%) are very closed to each other except the GaussianNB (i.e 58.5%). The Recall result is very much consistent with the Overall Accuracy Result. However, the Precision result aren’t showing any huge lacking deficiency between classifiers, their scores are ranged in 73%-83%. As we would not be achieving any optimal accuracy, so that further analysis is encouraged to optimize the classification accuracy.

*Table4:*



**Step7: Phase II Classification**

In the Phase II classification process, FIVE classifiers with 30 Extra features (i.e. total features = 59) were attempted to be implemented to compare their accuracy metrics. The extra features which are mainly consisted of the count of Tagging in the segments are. With the new added features, Accuracy results are able to be improved or boosted. As we see the table 5 below, the classifiers Gradient Boosting, SVM Linear and Random Forest Classifiers are able to yield a higher overall Accuracy result up to 74-80% along with better performance on Sensitivity which is up about 10% averagely ranged in 82%-87% although there is NO obvious improvement on the Precision Score (Further discussion in the later steps). Among these good performed classifiers, we are going to dive in one step (i.e. Step 8) further to check which model would be the best distinctive classifiers.

*Table5:*

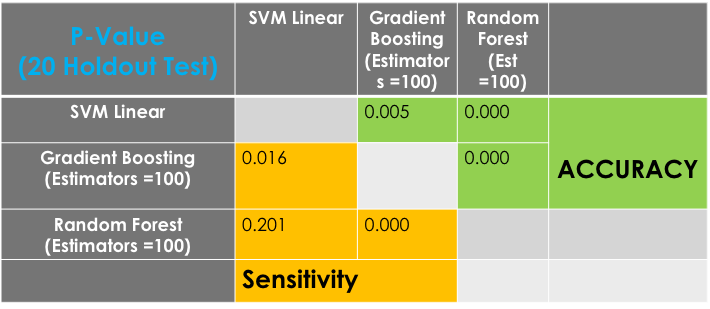


**Step8: Classification Model Comparison**

In this step, twenty-round of holdout test and T-test performed on the three classification models (i.e. SVM Linear, Gradient Boosting and Random Forest). The result in Table 6 below shows that **SVM-Linear model and Random Forest** both obtain the best Recall score ~85% with P-value higher than 5%. Similarly, **Random Forest** is the best model for the Precision (i.e. ~83%) and Overall Accuracy (i.e. ~80%) result among all three classifiers.

*Table6:*



*Table7:*

**Step9: False Positive adjustment**

As we focus to optimize the accuracy of True Positive rate for this experiment, so that the following two approaches were implemented in order to test whether the Recall and Precision score would be able to improved. Both of the approaches are to shift from class 1 to 0 if either of the conditions is met. Condition 1.) The first condition is that the Target labels shift from 1 to 0 if the Polarity score of segments are less than a preset value (e.g. -1.6. The preset values were iterated through +2 to -2 in order to demonstrate the change of Accuracy through the iterations. 2.) The second condition is to check the similarity of every word in the segments whether they are in the set of Positive sentiment words. If not so, the target label shift from 1 to 0. The set of Positive sentiment words were generated by Word2Vec function with a set of standard Positive sentiment words (i.e. ["great", "soft", "taste", "recommend", "good", "like", "liked" ,"love", "loved", "tasty"] ). Unfortunately, the False Positive rate were not shrink during these two-adjustment process. Opposingly, the True Positive rate were declined. For this reason, we would call the Positive Adjustment procedure failed in this experiment.

1. Relabel target class based on the value of Polarity Score
   * If Polarity Score < preset value:

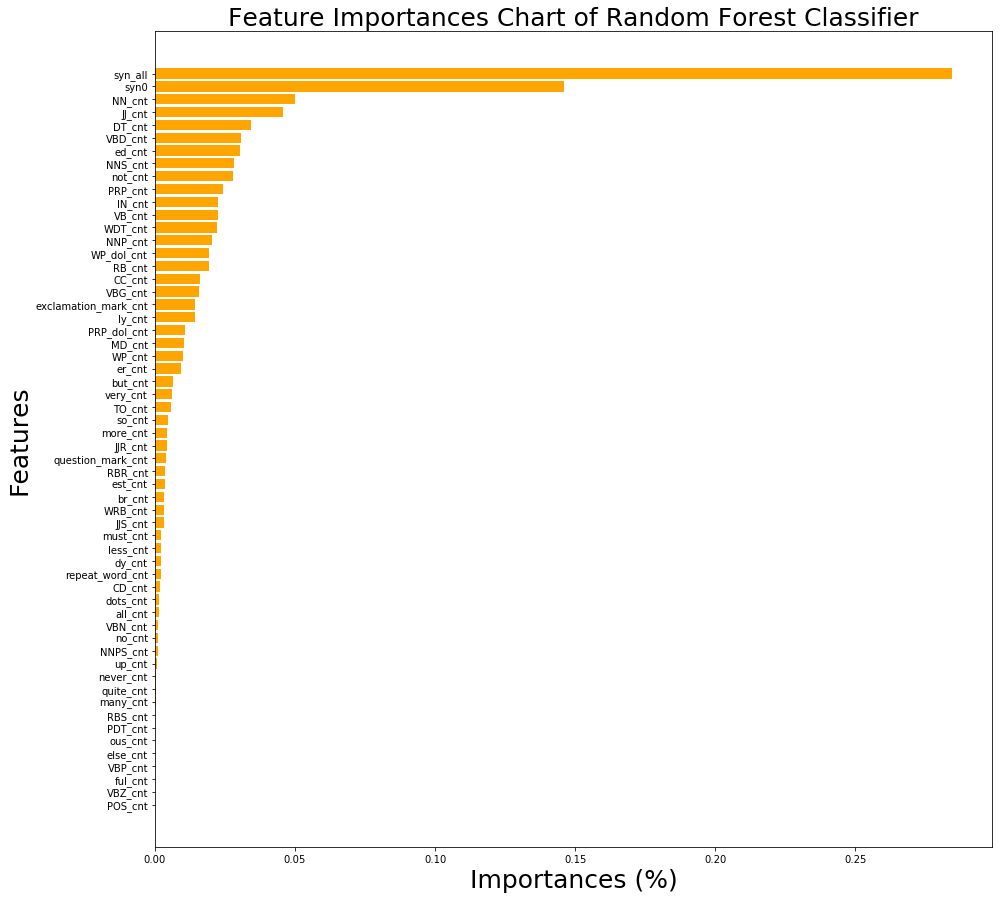
Result: Target value 🡪 0

1. Relabel target class based on the similarity of positive sentiment value
   * Check the similarity of words in every segments with a set of positive sentiment words (found by Word2Vec Function)
   * If any word in segments NOT in [list of positive sentiment word]

Result: Target value 🡪 0

**Summary / Future Improvement**

As far as the result goes, the review sentiment experiment achieves a considerably acceptable classification result. With ~1320 instances training set and ~680 instances testing set, the Random Forest algorithm offers overall Accuracy ~80% along with ~85% Recall score and ~83% Precision score. The parameters for this best model includes 100 estimators, 10 max features, 2 min. sample split, 2 min. sample leaf and “gini” criterion. The following plot shows the distribution of importance level of “Features”. We could observe that the Polarity scores take the top 2 ranked features and most of the followed runner-up features are the POS Tagging (e.g. NN, JJ, DT etc.). By following these facts, this distribution makes sense to the overall result because these added Part-Of-Speech Tagging features really boost the accuracy in the second phase analysis.



Although we’ve obtained an above average classification result, I would propose a number of ways to improve the model. The classification analysis is a learning process from the properties of features in the dataset which yields the probability of class labels of each instance, so that we all understand the quality of features of dataset is one of crucial parts of building a classifier. According to this concept, I would propose the followings in order to improve the Accuracy of the classification process. 1.) We should use as many as instances to train the model in the learning process which means that more manually marked instances we should make during the training stage, then we should have more solid ground truth embedded in the model to yield more accurate result. 2.) Based on the point 1, we might be able to obtain more relevant features from dataset which is another way to improve the performance of the classifiers. Besides, failure of False Positive adjustment implies that we might need to re-construct the system for False Positive adjustment purpose. 3.) We could measure the similarity of Positive sentiment word in the segment with the set of standard Positive sentiment words in Bigram or Trigrams format instead of only using Unigram. With this method, we might be able to measure the probability of phases in the segment more accurate in order to shrink the False Positive rate. Grammar pattern design and segment extraction are the major part of this project. As we can see the example of extracted segments, some of them are not completed segment, Opposingly they are fragments which are not able to express the whole meaning of segments / review etc. In this case, the probability of the classifier yields the correct class labels would be a lot lower. For this reason, 4.) I think the other action of improvement is that we should keep improving the quality of Tagging patterns in order to extract the most meaning segments out of the original paragraph/ reviews. However, I would recommend to take closer look of the original text before designing the patterns because reviews in different product categories would have different styles of writing grammars / patterns. So that the more we understand the text patterns, the better the grammar patterns we could design.

By the meaning of Improving the quality of grammar patterns, we could see the table 8 below

the extracted segments with more representable meaning which probably makes our model returning more predictable results. Opposingly, this table 9 shows the Extracted segments that are fragmented during parsing process. They might not mean nothing if we just take the meaning of one segment; They however might show the whole meaning if we combine them together. Thus, extracting a whole meaning segment is one of the vital factors making us success in this type of project.

*Table8:*

**Examples of Extracted Patterns with completed meaning:**

|  |  |
| --- | --- |
| **1.** | [('we', 'PRP'), ('already', 'RB'), ('buy', 'VBP'), ('the', 'DT'), ('chewy', 'NN'), ('oatmeal', 'NN'), ('bars', 'NNS'), (',', ','), ('adding', 'VBG'), ('these', 'DT'), ('soft', 'JJ'), ('baked', 'VBD'), ('oatmeal', 'JJ'), ('raisin', 'NN'), ('cookies', 'NNS'), ('to', 'TO'), ('our', 'PRP$'), ('list', 'NN'), ('of', 'IN'), ('quaket', 'NNP'), ('products', 'NNS'), ('!', '.')] |
| **2.** | [('i', 'PRP'), ('found', 'VBD'), ('these', 'DT'), ('cookies', 'NNS'), ('really', 'RB'), ('moist', 'JJ'), ('and', 'CC'), ('yummy', 'NN'), ('.', '.')] |
| **3.** | [('i', 'PRP'), ('highly', 'RB'), ('recommend', 'VBP'), ('this', 'DT'), ('cookie', 'NN'), ('...', ':')] |
| **4.** | [('it', 'PRP'), ('kind', 'NN'), ('of', 'IN'), ('had', 'VBD'), ('a', 'DT'), ('salty', 'NN'), ('after', 'IN'), ('taste', 'NN'), ('.', '.')] |
| **5.** | [('lots', 'NNS'), ('of', 'IN'), ('oats', 'NNS'), (',', ','), ('brown', 'JJ'), ('sugar', 'NN'), (',', ','), ('cinnamon', 'NN'), (',', ','), ('and', 'CC'), ('not', 'RB'), ('too', 'RB'), ('many', 'JJ'), ('raisins', 'NNS'), ('.', '.')] |
| **6.** | [('lots', 'NNS'), ('of', 'IN'), ('oats', 'NNS'), (',', ','), ('brown', 'JJ'), ('sugar', 'NN'), (',', ','), ('cinnamon', 'NN'), (',', ','), ('and', 'CC'), ('not', 'RB'), ('too', 'RB'), ('many', 'JJ'), ('raisins', 'NNS'), ('.', '.')] |
| **7.** | [('they', 'PRP'), ('do', 'VBP'), ('not', 'RB'), ('taste', 'VB'), ('fake', 'VB'), (',', ','), ('they', 'PRP'), ('are', 'VBP'), ('not', 'RB'), ('overly', 'RB'), ('sweet', 'JJ'), (',', ',')] |
| **8.** | [('i', 'PRP'), ("'m", 'VBP'), ('not', 'RB'), ('a', 'DT'), ('big', 'JJ'), ('fan', 'NN'), ('of', 'IN'), ('the', 'DT'), ('oatmeal', 'JJ'), ('raisin', 'NN'), ('cookies', 'NNS')] |

*Table9:*

**Extracted segments that are fragmented during parsing process:**

|  |  |
| --- | --- |
| **1.** | [('a', 'DT'), ('delicious', 'JJ'), ('little', 'JJ'), ('snack', 'NN'), (',', ',')]  [('can', 'MD'), ('also', 'RB'), ('substitute', 'VB')]  [('here', 'RB'), ("'s", 'VBZ'), ('a', 'DT'), ('breakdown', 'NN')] |
| **2.** | [('my', 'PRP$'), ('least', 'JJS'), ('favorite', 'JJ'), ('part', 'NN'), ('was', 'VBD')]  [('how', 'WRB'), ('crumbly', 'RB'), ('it', 'PRP'), ('was', 'VBD'), ('.', '.')] |
| **3.** | [('it', 'PRP'), ('has', 'VBZ'), ('oatmeal', 'VBN'), ('is', 'VBZ'), ('a', 'DT')]  [('plus', 'JJ'), (',', ','), ('meaning', 'VBG'), ('my', 'PRP$')]  [('kids', 'NNS'), ('are', 'VBP'), ('putting', 'VBG'), ('something', 'NN'), ('healthy', 'JJ'), ('in', 'IN')] |
| **4.** | [('i', 'PRP'), ('have', 'VBP'), ('absolutely', 'RB'), ('fallen', 'VBN')]  [('love', 'NN'), ('with', 'IN'), ('the', 'DT'), ('soft', 'JJ'), ('baked', 'JJ'), ('cookies', 'NNS'), ('!', '.')] |
| **5.** | [('i', 'NNS'), ('normally', 'RB'), ('do', 'VBP'), ('not', 'RB'), ('like', 'VB'), ('oatmeal', 'JJ'), ('cookies', 'NNS'), (',', ',')]  [('but', 'CC'), ('these', 'DT'), ('were', 'VBD'), ('sooo', 'JJ'), ('soft', 'JJ'), ('and', 'CC'), ('chewy', 'NN'), ('.', '.')] |