Machine Learning, NLP

CS-583 DMTM Project 2: Aspect based Sentiment Analysis (Reviews dataset. 2 nos.) [vsingh38@uic.edu]

# Abstract

The contemporary market is inundated with a myriad consumer products, goods, and services, offered by a plethora of companies, and even individuals. This proliferation of options, available to the users, presents a unique challenge that is also commonly known as the **‘Choice Paradox’**. It becomes increasingly difficult for the users to make the right, or even optimal, decision when buying a product, or choosing a service. It is beneficial therefore, for both the user, and the businesses, to be able to identify and promote the best products. For the user, this would help in making the right choice, and for the businesses, an increase in the profit, by focusing on products that are rated good by the consumers and improving those, that need improvement.

The Machine Learning system developed here, uses the reviews for products and services, to analyze and predict user sentiments. The system not only learns to predict the sentiment for a product, but also specific aspects of the product, or service offered. For example, voice quality of a phone. Or customer service of an airline.

# Introduction

In this final project for the Data Mining and Text Mining (CS 583) course at the University of Illinois, Chicago, the task is, given an **aspect term** (also called opinion target) in a sentence (user review), predict the **sentiment label** for the aspect term in the sentence. The data set used, consists of the reviews given by the users of consumer products and goods. For this task, two data files are provided, data 1, and data 2. The first consists purely of user reviews for electronic gadgets and software products, whereas, the second comprises of reviews for a mix of services and consumer goods.

Several data analysis, preprocessing and machine learning techniques are used, in the solution, to identify the patterns in the data, to train the system to predict the target label, previously referred to as the sentiment. The data contains the review text, product id, aspect of the product for the sentiment, and the sentiment label itself, which can be positive, negative, or neutral. **Natural Language Processing** is used to identify **parts-of-speech** and **aspect dependencies** of the aspect terms.

# Techniques (data pre-processing, features used, classification methods tried) & Evaluation (results from different methods)

# Phase 1

## Stage I

1. Loaded data using Pandas.read\_data()
2. Preprocessing
   1. Removed punctuations.
   2. Used SnowBall stemmer with stopwords: English.
   3. TF-IDF Vectorization (unigrams). Using bigrams (and even tri-grams) did not yield any substantial improvement in the performance.
3. Resolved issue with multiclass F1. Used f1-score(average=’weighted’).
4. Issue with precision\_recall\_curve() not supporting multiclass.
5. Used only ‘document’ and ‘label’ columns for now.
6. Used pyplot to write plotting library.
7. Initial run o/p

Processing 2203 samples with 5 attributes  
Evaluating classifiers

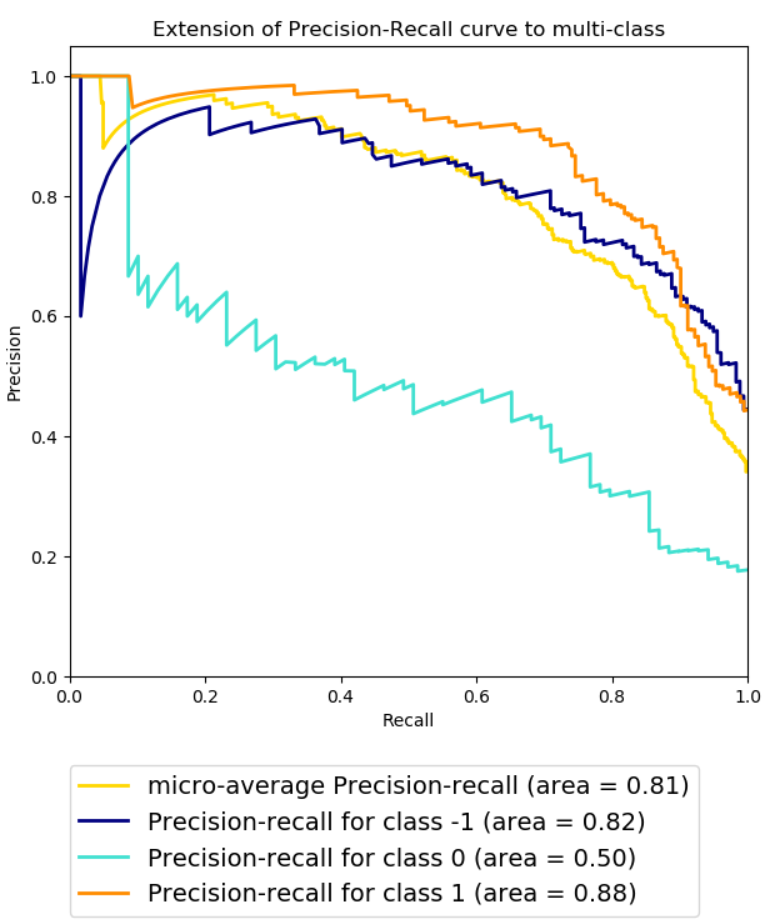
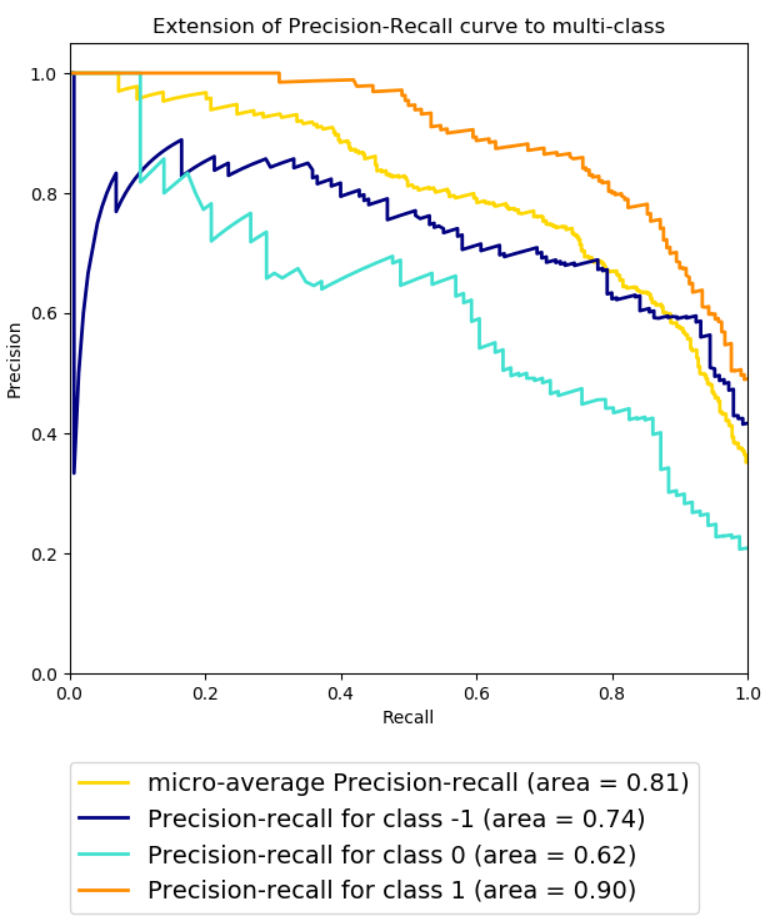
Plotting the results

Accuracy of clf: Linear SVC (F1 score=0.721) = 0.7188208616780045

Accuracy of clf: NuSVC (F1 score=0.729) = 0.7301587301587301  
Accuracy of clf: Ada Boost (F1 score=0.589) = 0.6009070294784581

## Stage II

1. Implemented multiclass Pr/Re
   1. Used label binarizer (required for multi-class Precision-Recall).
   2. Used OVR with LinearSVC (default params).
   3. Used Micro-Averaged PR
      1. Micro- and macro-averages (for whatever metric) will compute slightly different things, and thus their interpretation differs. A macro-average will compute the metric independently for each class and then take the average (hence treating all classes equally), whereas a micro-average will aggregate the contributions of all classes to compute the average metric. In a multi-class classification setup, micro-average is preferable if you suspect there might be class imbalance (i.e you may have many more examples of one class than of other classes).
      2. Ref: <https://datascience.stackexchange.com/questions/15989/micro-average-vs-macro-average-performance-in-a-multiclass-classification-settin/16001>
2. Cross-Validation
   1. Single (80-20) fold for now.
   2. TODO: 10-fold CV with GridSearch.
3. Uses average weighted F1, and overall accuracy.
4. Average precision score, micro-averaged over all classes: 0.81
5. Result Visualizations (data 1):

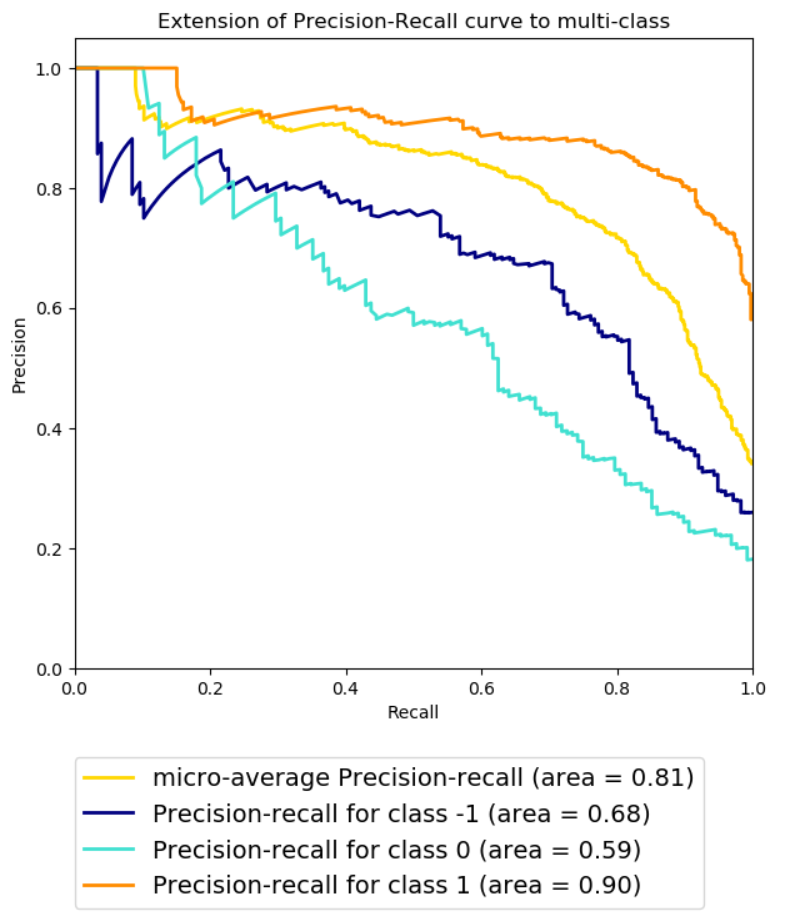
 

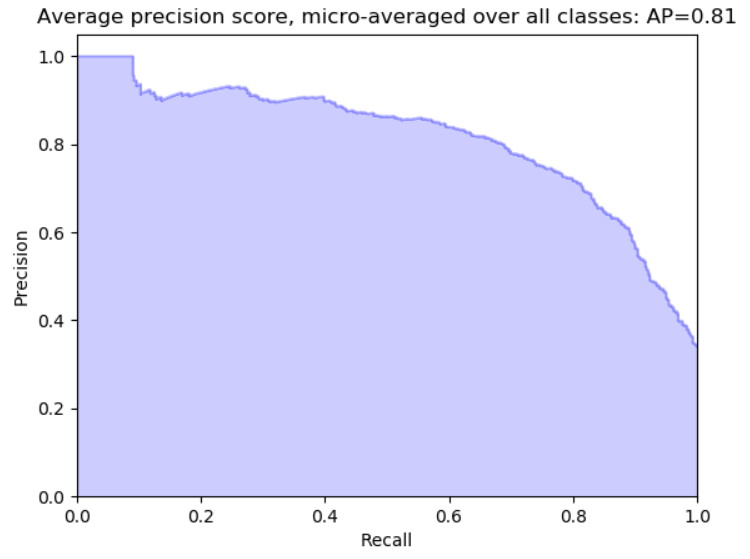
*With Stemmer A comparison w/o Stemmer*  
**OVR LinearSVC (F1 score=0.725, Accuracy=0.6644)**  **OVR LinearSVC (F1 score=0.719, Accuracy=0.6667)**

**Average precision score, micro-averaged over all classes: 0.81**

1. Results from data 2 (with stemming).  
   **OVR LinearSVC (F1 score=0.725, Accuracy=0.6907)**

**Recall (Positive) = 90%**





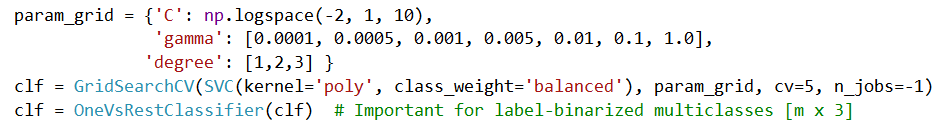
# Phase 2

1. Evaluation results using RBF SVM with GridSearch (Data 2):

**Processing 3602 samples with 5 attributes  
Evaluating classifiers  
SVC with RBF and GridSearchCV (F1 score=0.725, Accuracy=0.6782)  
Average precision score, micro-averaged over all classes: 0.72**

No visible improvement!

1. Using 5-Fold CV (GridSearchCV) with SVM poly kernel and OVR multiclass discrimination.

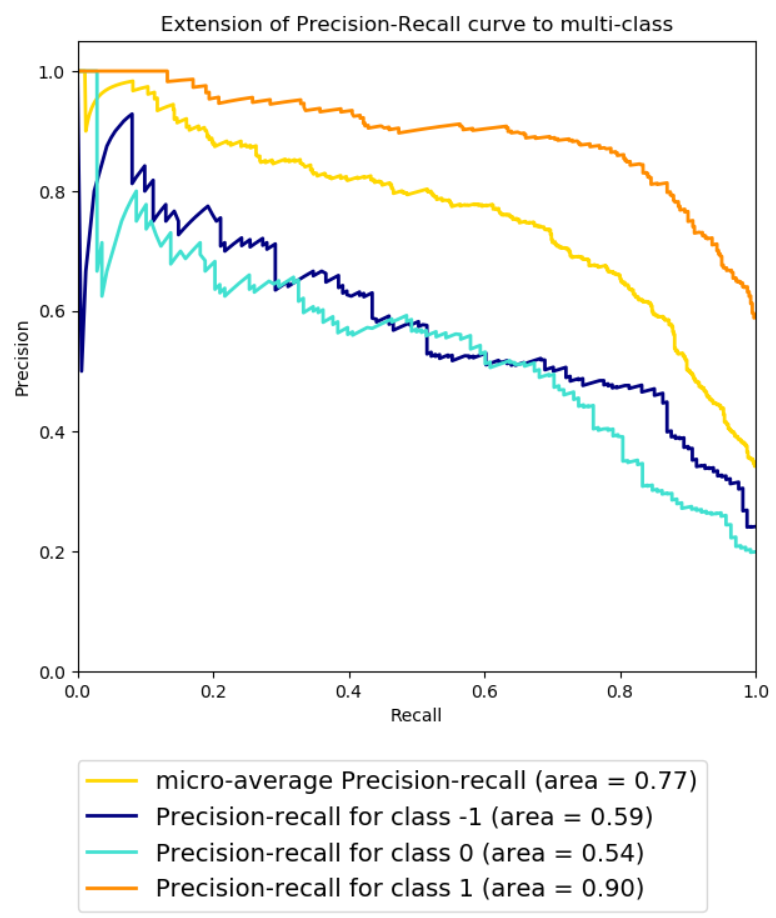


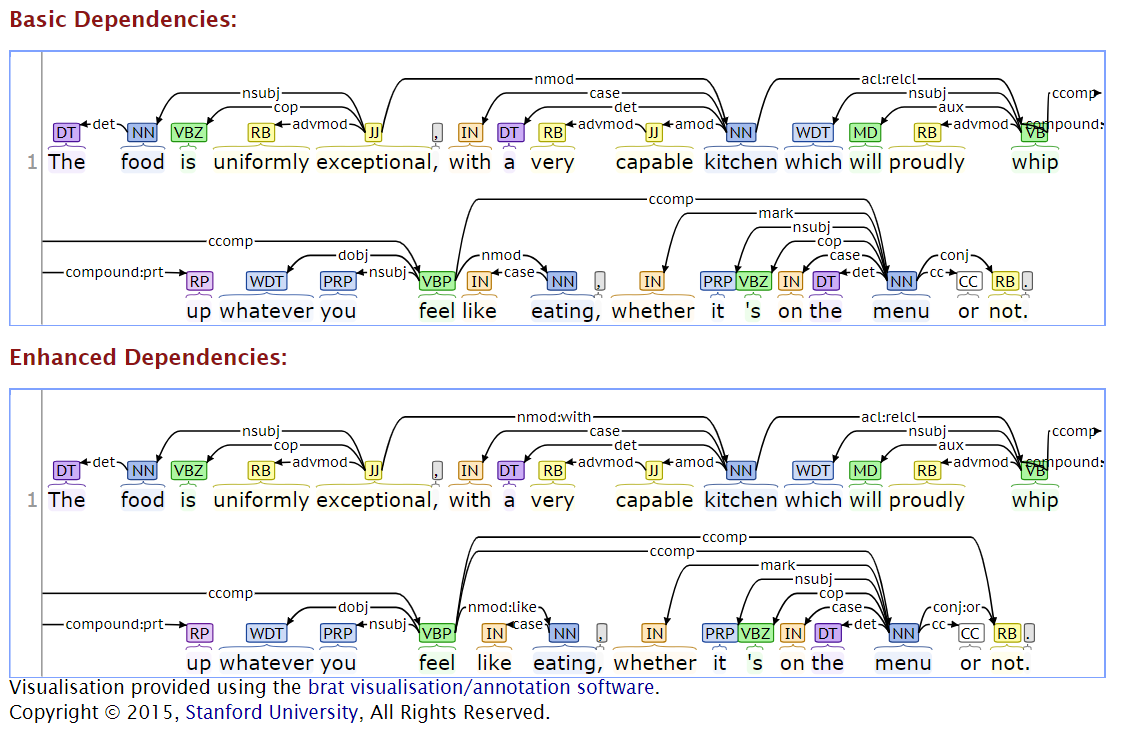
**Processing 3602 samples with 5 attributes**

**Evaluating classifiers**

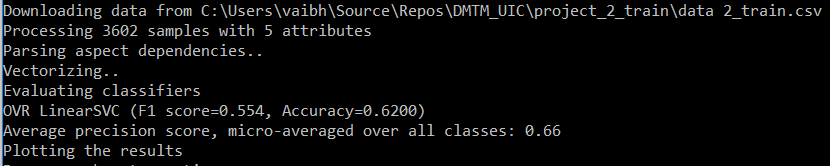
**poly SVM with OVR and GridSearchCV (F1 score=0.675, Accuracy=0.6366)**

**Average precision score, micro-averaged over all classes: 0.72**

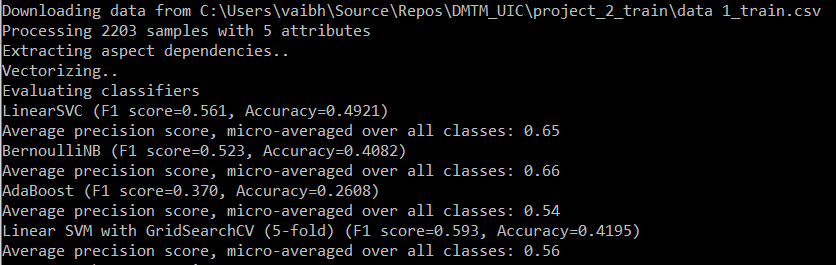
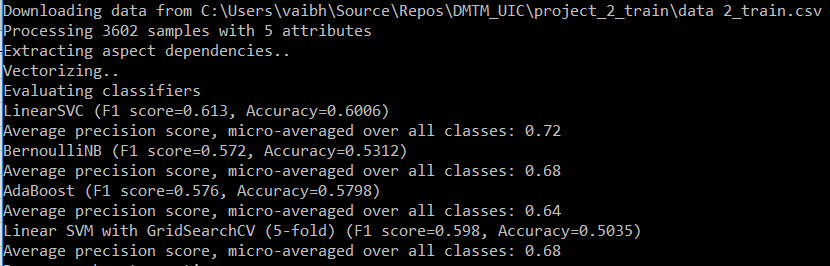
1. Using 5-Fold CV (GridSearchCV) with Linear SVM and OVR multiclass discrimination.  
   Linear SVM with OVR and GridSearchCV=5 (F1 score=0.717, Accuracy=0.6436)  
     
   
2. Using NLTK (Stanford CoreNLP parser)
   1. Parsed recursive dependencies using scoped dependency-type list.  
      Following visualization presents an overview of dependency parsing using one of the training examples.



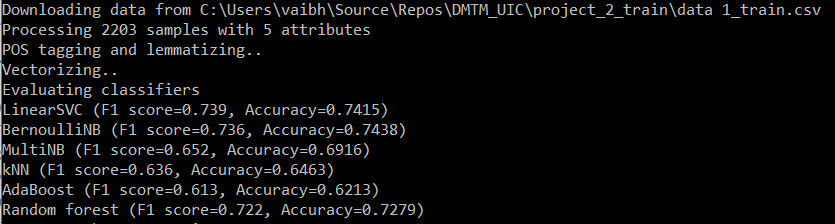
* 1. First o/p



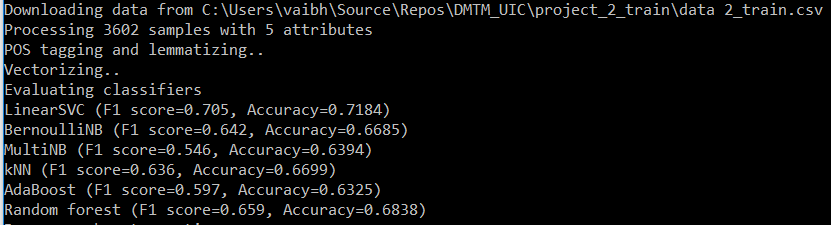
* 1. Further scoped to extract pos - adjectives and nouns. Excluded stop-words.  
     o/p (data 1 vs 2)

1. Using a different text processing strategy: **Document->Sentences->Tokens->POS->Lemmas**
   1. Used **NLTK POS-Tagger** and **Lemmatizer**
2. Results (data 1 vs 2):
   1. Data1: **Accuracy 74%, using LinearSVC** and BernoulliNB.



* 1. Data2: Accuracy 71%, using LinearSVC (C=5)



# Conclusion

There is inherent ambiguity in processing Natural Languages. Figures of speech like, metaphors, sarcasm, slangs, emoticons, etc. introduce interpretation, and POS identification complexities. Although, the current system does good at pre-processing and parsing the texts, it does not do a spectacular job at predicting the sentiments, with the highest accuracy at only around 75%, it still shows the approach towards solving the text analysis problems from a Machine Learning lens.

Further work on this project will continue, where the goal is to improve the features using other NLP techniques, like assigning aspect-proximity weights to important terms, identifying negative terms, using **WordToVec** embeddings to extract contextual information, and training **Neural Networks** for Deep Learning.

# References

1. <http://scikit-learn.org/stable/documentation.html>
2. <https://nlp.stanford.edu/software/lex-parser.html>
3. <https://www.nltk.org/>
4. <http://pandas.pydata.org/pandas-docs/stable/>
5. <https://docs.scipy.org/doc/>

---END---