NLP Final Project

Alexandre Bettencourt, Jing Dong, Jingyao Li, Xin Zheng

December 2020

1 Abstract

Understanding whether the customers like or dislike the products by analyzing their reviews becomes more and more critical to businesses in improving products and launching new products. In this paper, we aim to analyze the sentiments of Amazon food reviews up to October 2012 using supervised machine learning methods and deep learning methods, to develop a better understanding of customer opinion automatically. In supervised machine learning methods, we evaluate the performance of different features, including POS tags of unigrams and bigrams, lexicons and word embedding. Then, we train each feature with three different classifiers, Naive Bayes, Maximum Entropy and Support Vector Machine. In the deep learning method, we use BERT to train and test the reviews. As a result, we found that BERT generates the best accuracy scores, while different combinations of features and classifiers generates different performance in accuracy, but generally maximum Entropy outperforms Naive Bayes and SVM. For bag of word, unigrams with POS tags has better performance. For word embedding, word2vec has better performance.

2 Introduction

Our goal for this project is to use sentiment analysis in order to classify Amazon product reviews. The motivation for this project is as follows: we want to be able to determine whether or not a given review of a product is helpful. Here, a helpful review will be defined as one that has been deemed as helpful for the user base. We believe that engaging in such a project will result in a list of products of the highest quality, or in other words, the ones that worked the best for the community of users. As for the extension, the time spent by users in order to find a good recipe for a target dish or to find the best brand of some specific food can be diminished, and it can also be ensured that the users will have the best possible recipes or products recommended to them.

In order to engage in such a task, of course, a relevant data set is required (more specifically, a data set containing product reviews written by consumers). After debating whether or not to build our own data set of reviews, we decided to use the data set called "Amazon Fine Food Reviews" which contains over 500,000 reviews. Along with these reviews also came relevant data about them, such as the rating given to a product and a helpfulness score of a review, or in other words how helpful users thought a given review is. Such data presented convenient pre-processing choices for the large data set. For example, we deem any review presenting a rating lower than 3 being a negative review, and any review presenting a rating higher than 3 being a positive review. This essentially translated into convenient separation and storage of data, which turned out to be very helpful for training our models later on when performing sentiment analysis with different features.

3 Survey of Related Works

Developed from text categorization, sentiment analysis is more complex due to the large vocabulary to express positive or negative feelings. At first, continuing the method used in text categorization, most works in sentiment analysis were conducted at least partially knowledge-based. The traditional text categorization approach regards documents as bag of words and maps it to a word vector, then classifies each document using one of the three most-used algorithms, Naive Bayes, maximum entropy classification, or support vector machine. In 2002, Pang, Lee and Vaithyanathan introduced a fully unsupervised machine-learning method to classify the sentiment. They proposed three methods, Naïve Bayes, maximum entropy classification, and support vector machines. After examining the performance,they conclude that the three machine learning algorithms beat the performance of human-produced baseline systems, in which they classify texts based on the words of different sentiment provided by humans. However, when compared with traditional text-categorization methods, the three machine learning algorithms underperform. (Lee, Pang, and Vaithyanathan, 2002) In 2011, Xia et al. also utilized the three text categorization algorithms and compared the performances of using a single algorithm and feature, using ensembles of features, and using ensembles of both algorithms and features. They used pos-tag feature and word relation feature. They conclude that ensembles of algorithms and features are more effective than other combinations. (Xia, et al., 2011) As for the feature aspect, another method used in sentiment analysis is word embedding. Word embedding can be a better feature compared with atypical bag of word regard of the general better performance. Some researchers apply word2vec with random forest classifier as features to do sentiment analysis on Twitter data and achieves 0.81 accuracy (Deho et al., 2018).GLoVe is another popular word embedding algorithms and can be used in sentiment analysis (Pennington et al., 2014). However, the disadvantage of word embedding is, all tokens are transformed into vectors and it is not explainable for the classifier to directly point out which token has a higher importance. Besides bag of word and word embedding features,n lexicon-based features are also popular with sentiment analysis tasks. In 2017, Linh Vu used a lexicon-based method to do sentiment analysis for understanding what customers think about business products or services. Their method combines the method of Baccianella et al.[Baccianella et al., 2010], which utilizes SentiWordNet, with that of Liu et al. (LIU) [Liu et al., 2004, 2005] and a heuristic data pre-processing method for sentimentoriented filtering of words. Basically it includes three steps which are Lexicon loading, Text pre-processing, and Sentiment scoring. They did experiments on three dataset with their methods in three different versions compared with with SentiWN method and LIU method [Liu et al. 2004, 2005]. And it turns out that all three versions of their proposed method outperformed the SentiWN and LIU. Specifically, the hybrid version performed the best. In 2015, Hussam Hamdan, and Patrice Bellot also proposed a lexicon-based method which they build a new automatic sentiment lexicon using new metric called natural entropy from sentiment140 corpus. They rewrote the semantic orientation and they derived six features from the new lexicon to show that their metric outperforms the PMI metric which is often used for computing the semantic orientation as well.

Besides the traditional machine learning method, the start-of-art algorithm applied on sentiment analysis also includes deep learning methods such as Convolutional Neural Network, Recurrent Neural Network and Long Short Term Memory (Yang et al., 2016). In 2017, the Attention architecture was introduced to the NLP area. The transformer enables the input of "encoder-decoder" architecture to to pay attention to more important information of a sentence instead of the entire sentence which makes the computation more efficient. (Vaswani et al.,2017) After this breakthrough, some algorithms make extensions based on the attention network. One of them is the hierarchical attention networks called HAN (Yang et al., 2016). Not only focusing on the sentences levels, "it has a hierarchical structure that mirrors the hierarchical structure of documents" and also it applies two levels of attention mechanism on the word and sentence levels. In this way, the HAN has a better ability to pay attention to the more and less important context while creating the representation of the document (Yang et al., 2016).Another important technique they use is BERT– Bidirectional Encoder Representations from Transformers. Adding a layer of classifier, it is improves the performance of sentence classifications (Devlin, et al.,2018)Based on our deeper understanding of traditional machine learning models and features, we will also apply deep learning methods in our project.

4 Formulation

We formulate the problem as a binary classification problem, where we need to train a classifier to classify each review as positive or negative. In Amazon Fine Food dataset, all the data are stored as a csv file which contains helpfulness, ratings and reviews. Our task is to build a classifier using part of the data which are considered as helpful to customers, with the features extracted from reviews and labels extracted from ratings. And then, with the classifier, we test and validate the results using the rest of the data, by predicting the labels given reviews, and then compare the predicted labels with the actual labels. We use 5-folds cross validation to evaluate the performance of our classifiers. Performance of different classifiers are evaluated by the confusion matrices. Accuracy, F1-score, Precision score and recall score are computed. Then by comparing those scores, we are able to have a comprehensive evaluation of the performance of different classifiers. Using the classifier with the best performance, we are able to classify the customer reviews as positive or negative, to help the businesses in online fine food industry understand customer attitudes better.

We solve the problem using two different methods: supervised machine learning and deep learning. In supervised machine learning methods, we utilize three classifiers, Naïve Bayes with Gaussian, Maximum Entropy, and Support Vector Machine. We then formulate different approaches to extract features. The baseline model is unigram with bag of words. Then, we formulate our problem as bag of words using N-gram model combining with pos tags. Since we only consider words with the pos tag that can best convey sentiment, it will remove noise. Next, we formulate another feature using word embedding, in which word2vec and Glove are used. Finally, we formulate our problem as extracting a lexicon-based feature to train the classifier. In deep learning method, we use BERT, in which we use a multi-layer CNN classifier to improve the accuracy.

Overall, our problem is a classification problem, in which we use comprehensive methods, including bag-of-words cooccurance using unigram, unigram with pos, and bigram with pos, lexicon analysis, word embedding in supervised machine learning method, and BERT with CNN in deep learning method.

5 Approches

5.1 Overview

This section is about the overview of our project. Figure 1 is the overview of our system. First, we have a data processor that clean and tokenize the reviews and prepare for the feature extraction parts. To be more specific, we first filter out those unhelpful data which means the $\frac{\text{number of users who found the review helpful}}{\text{number of users indicate the reviews helpful or not}}$ is 0. Then we remove all contents inside $< >$, remove punctuation, and lower the tokens.In the meanwhile, we also label our data. The score higher than 3 means positive and the score lower than 3 means negative. It's important to notice that we eliminate the neutral data, which means when the score is 3. There are two reasons. First, neutral is useless to analyze the sentiment and also to dig customer's attitudes to a specific product. Additionally, the neutral data in general only takes 5% of the whole dataset. In this way, it also shows that customers think neutral data as useless.
Then in our system, it can be divided into two parts. One is the feature construction for the traditional machine learning models.We decided to apply unigram with POS, bigram with POS, lexicon-based features, word2vec and GLoVe to vectorize the reviews. Then those feature vectors will be fed into different three classifiers: Naive Bayes, Maximum Entropy, and Support Vector Machine. The second part is the deep learning part. Our system will first use Bert tokenizer and then directly transform the tokens into id which mapping to the embedding for that specific tokens and then the feature vectors are directly fed into a three-layers CNN to do the sentiment analysis. Finally, our system applies cross validation part to do the evaluation and delivers four scores: accuracy, f1 macro, precision macro and recall macro. We choose macro instead of weighted is because our data is relatively imbalanced. The rough ratio between the positive and negative is 4:1, thus the macro can better reflect the performance of the minority classes instead of favouring the majority class.

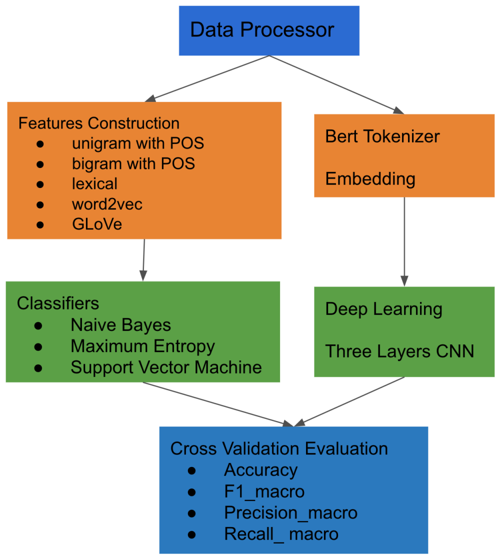

Figure 1: The Overview

5.2 Features

5.2.1 Unigram with POS

One of the features we used are unigrams. The main idea behind this feature is to determine whether or not having a certain set of unigrams as features can then translate into feature vectors that will be accurately classified by our models. So how can we get such a set of unigrams? The approach used in this project is as follows: for each review that remained after pre-processing, we store each word in a map along with its count. After this is done, we take an arbitrary number, say x and retrieve the x-most frequent words and use those as our features in order to build our feature vectors. However just doing this lead to average performance, as many frequent words within reviews did not necessarily tell a lot about the sentiment of a review, thus leading to useless features. For example, frequent words such as "chocolate" or "amazon" clearly aren't reflective of the sentiment of a review.

After observing this, the idea of using POS tags to get more accurate feature sets came up. Focusing on tokens with POS tags such as adjective and verbs (JJ, VB) showed itself to be a good way of overcoming the problem mentioned previously, as now our feature set would for the most part contain frequent adjectives and verbs such as "great", "nice", "love", etc.

Another factor that would make performance vary when it came to classification was the size of the feature set: if the size was too big, the feature set would become too sparse and contain words that still don't say much about sentiment, such as "eat" for example. On the other hand, if it was too small, many words that in fact are useful for classification would be left out. More specifically speaking, for our data set the feature set of size 20 (in other words, representing the 20 most frequent adjectives and verbs) performed the best while keeping a good range of useful words (performance on classification would decrease by around 3% for a feature set of size 200, for example).

5.2.2 Bigram with POS

In addition to analyzing unigram feature of certain pos tag, we also extract bigrams to involve the contexts of the word. A straightforward bigram method is to extract bigrams from the sentences of reviews. Then, we can use traditional bag of words method to produce a vectorized features. However, using this method will cause data sparsing and overfitting problem, because a bigram may contain words specific to the certain product, which thus reduce the matching frequency. Also, it might create overfitting problem since the training set and the test set both contain the reviews of the same product. Therefore, to avoid those problems, we utilizes bigrams along with pos tags to create a more general feature.

In 2010, Xia and Zong introduced the bigrams with pos method. (Xia and Zong, 2010) This is a slight variation version of their methods. When considering the sentiment of pos tags, we recognize that adjectives are most frequently used in sentiment expression. Adverbs are also useful when combining with adjectives or verbs. Verbs are useful in some cases such as "recommend". Nouns and pronouns are not as useful as the other three categories. And, we do not consider any other pos tags. Thus, to formulate a more general form of bigrams, we separate the pos tags into four clusters: J:{JJ, JJS, JJR}; R: {RB, RBS, RBR}; V: {VB, VBZ, VBD, VBN, VBG, VBP}; N: {NN, NNS, NNP, NNPS, PRP}. For each combinations of bigrams, as long as one of the words is in the "N" cluster, we use "N" to substitude the original word. For example, if we have a bigram "great taste", the word "taste" is substituted by "N". And, in the case of a bigram of adjectives followed by adverbs, we substitute the adverbs with the label "RB". Also, the bigrams with pos tag as "NN NN", "JJ VB" and bigrams containing other pos tags which are not in the clusters are removed from the features. In all other cases, both words are kept as original in order to get the context information.

5.2.3 Lexicon-based

For extracting lexical feature, we used VADER which is a lexicon and rule-based sentiment analysis tool. The reason we chose to use VADER is that VADER is good for analyzing sentiments on social media because it is specifically attuned to analyze sentiments expressed in social media. Therefore we consider VADER as an ideal sentiment-analysis tool for analyzing sentiment expressed on Amazon review session as well. Essentially, VADER will give each review four scores which are positive score, negative score, neutral score, and compound score. Compound score is the overall score for the text which is computed by summing the valence scores of each word in the lexicon, adjusted according to the rules, and then normalized to be between -1 which is the most extreme negative and +1 which is the most extreme positive. We consider the compound score as the most important metric and thus we are using the compound score as one of the feature that is lexical based. Besides, the positive, negative, and neutral scores are ratios for proportions of text that fall in each category. We are using positive score and negative score as another lexical based feature because we consider these are good metrics for multidimensional measures of sentiment for a given review. We do not consider the neutral score as we do not consider neutral as one of the sentiment cases.

5.2.4 word2vec

Since the unigram and bigram are applying bag of words, those methods are creating a a sparse feature vectors with high dimensionality. In this way, it's execution time is expensive. Additionally, it lacks notion of similarity. Then we try the word embedding. One of the most popular approach is word2vec.It calculates the co-occurrence within local context(neighbouring words) . We choose skip-gram algorithm since it is used for predicting the context and is good at rare words. We hope this algorithm can help us to catch some rare token that can represents sentiments.

5.2.5 GLoVe

Besides word embedding, we also decide to apply GLoVe. GloVe represents the Global Vectors for Word Representation.It develops a new global log-bilinear regression model combining the advantage of two popular models: the global matrix factorization and local context window. Thus, compared with word2vec, GLoVe, not focuses on the local text, but the global information. To construct the GLoVe, the first step is to create a word-word cooccurrence matrix. Each element X_{ij} in the matrix represents the co-occurrence times that token i and the token j in a specific size of context window. In this way, it eliminates those non-co-occurrence pair such that the word-word co-occurrence is denser.

5.3 Models

5.3.1 Naive Bayes

When it came to selecting models for our experiment, we immediately thought of using a Naive Bayes classifier as it is frequently used in the field of Natural Language Processing and thus would give us a good idea of how well certain features work in contrast to others. Being a conditionally probabilistic model, the Naive Bayes classifier will assign probabilities to sequences of features based on their likelihood to show up following previously observed features (or in other words, the history). Using such a model and observing the results produced from it will provide us a good insight as to how well a conditionally probabilistic approach will perform for our task of classifying reviews as emotionally positive or negative.

5.3.2 Maximum Entropy

Maximum Entropy is one of the widely used classifier algorithm in text classification. It is a generative model which estimates the conditional probability of features X, on the labels Y. Maximum Entropy utilizes the principle of maximum entropy, which is that the classification should reach the maximum entropy. Statistically, it means that the distribution is made to be as uniform as possible. Unlike Naïve Bayes which evaluates the independent probability, Maximum Entropy considers the optimization to find the weight which maximum the likelihood of the training data. Therefore, it can work well with connected features. Since Maximum Entropy is similar to Logistic Regression with multiclasses in statistics, in our implementation, we use Logistic Regression from the linear model in sklearn in Python to build our classifier.

5.3.3 Support Vector Machine

Support Vector Machine(SVM) is one of the model we chose to do experiment with since SVM is popular with classification problems usage and therefore we consider SVM as an essential model to include in our project. Basically, the support vector machine algorithm is trying to find a hyperplane in an N-dimensional space, where N stands for the number of features, that will classifies the data points distinctly. In our case, we are using the non-linear SVM which in other words SVM using a non-linear kernel. The benefit of using non-linear SVM is that it can capture more complex and accurate relationships between our dataset because the decision boundary does not have to be a straight line. With using the non-linear SVM, we do not have to do further transformations, although it is true that this computation does take more time as it is much more computationally intensive than the linear SVM. Our SVM is imported from Scikit-learn, which is the SVC class.

5.3.4 Deep Learning-CNN

To improve our accuracy, we also try out the deep learning method. We decide to combine BERT with CNN. BERT, representing Bidirectional Encoder Representations from Transformers, is a state-of-the-art neural network which generalizes an architectures for NLP tasks. BERT relies on Transfomers and is a stack of encoders. The core of BERT and the reason of why it is the stack of encoders is because, it first pretrains general purpose language representation models applying unannotated text to first find the general representations of the languages. There are two steps of BERT. The first step is pretraining to create a general representation of language. Then it does fine-tune on smaller task-specific datasets. Finally, in our case, we add three layers CNN to do the sentiment analysis. We choose to use CNN is because RNN is too time consuming and BERT already has good performance on preparation for sentiment analysis. so we do not choose to use RNN.

6 Experiments & Analysis

6.1 Data Set

We uses the data from amazon find food dataset and takes first 50,000 data for our use. After we filtering out the useful data, there are The first finding we find is that the data-set is imbalanced. After filtering out "useless" reviews, we find that the neutral dataset is around 5% of the overall dataset. The positive is the majority classes, which have 16757 data points, making up 73% of our dataset. The negative class has 4322, which are 22857 in total. The specific data we use is the score, the helpfulness Numerator which means the number of users who found the review helpful, the helpfulness denominator which means number of users who indicated whether they found the review helpful or not and the text, which are the reviews.

In the above bar plot, 1 stands for positive review and -1 stands for negative review. It shows the binary version of our data. We can see that the distribution between positive reviews and negative reviews is imbalanced. From the word cloud of reviews, we could also see that there are more positive words other than the negative ones.

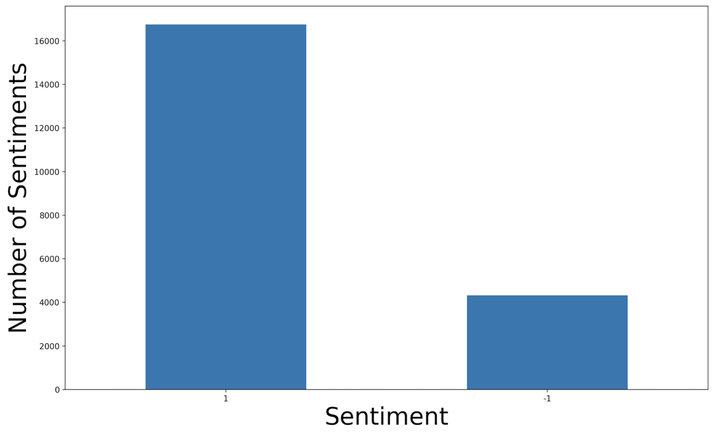

Figure 2: Distribution of our dataset

good
Great
Best
love
Delicious
taste
coffee
dog
Tasty
tea
flavor
Excellent
favorite
Yummy
Really
Chocolate
easy
chip
one
treat
healthy
buy
Great product
Wonderful
Amazing
Awesome
product
price
cookie
Perfect
snack
little
gluten free
bad
Yum
make
food
cup
stuff
cat
Organic
Nice
sweet
dogs love
OK

Figure 3: Reviews

6.2 Experiment Setup

For our experiments, we create a dataprocessor class, for each feature, we create a class and we also create a class for all classifiers. We also have an engineer which is our main method. During the implementation, we find that bigram with POS applying SVM runs out of memory on different computers. Thus, we will not experiment with SVM for bigram with POS features. Additionally, to make SVM executes faster, we apply BaggingClassifier. For Naive Bayes, we use Multinomial Naive Bayes for unigram with POS and bigram POS Gaussian Naive Bayes for the rest. For the deep learning part, we use Google Colab and work on the python notebook to increase our coding efficiency. The following tables show the details of implementations. In the column of experiment set up, there is information included hyper-parameters and other experiment decision so that our system can be easily replicated.

Feature Name	Libraries	Experiment set up
Unigram + POS	nltk	Eliminating irrelevant words; Bag of Words
Bigram +POS	nltk	Back "Nouns" to its POS tag; Bag of Words
Lexical + POS	vaderSentiment +nltk	vaderSentiment.vaderSentiment:SentimentIntensityAnalyzer
word2vec	gensim	Windowsize:3 Dimension: 200; Worker:3; skip-gram; Min-count:1
GLoVe	gensim	Pre-trained dataset: glove-twitter-200; Window size:200

Model Name	Libraries	Experiment set up
Naive Bayes	scikit-learn	Gaussian Navie Bayes
Maximum Entropy	scikit-learn	Logistic Regression; penalty:'l2'; C:1.0; max_iter:1000
Support Vector Machine	scikit-learn	BagClassifier kernel:'linear', probability:True class_weight:'balanced'; max_samples: 1.0/10 ;n_estimators:10
Bert	tensorflow + tensorflow_hub + bertfortf2 + sentencepiece	BERT_Layer: bert base;

CNN	tensorflow	Batch size :32; train_test_split: 0.2;
		embedding_dimension:128; cnn_filters:50;
		dropout_rate 0.1;number_of_layers:3;
		activation: relu; last_dense: sigmoid; number_epochs 5;
		loss:binary_crossentropy; optimizer:adam; metrics:accuracy

6.3 Results

In our experiment, we choose Gaussian Naive Bayes with unigram bag of word as our baseline model. From the result, it can be seen that it performs not well. Our features and models all beats the baseline model in accuracy and except for GLoVE with MaxEntropy, other models beat F1, precision and recall. In the result table on the next page, we only highlight the highest score for each metrics and other good performance feature and model. To evaluate the performance, we will first evaluate the deep learning method and supervised machine learning method. We found the BERT model with CNN classifier algorithm results in a 93% of accuracy, which significantly outperforms all different combination of features and classifiers in supervised machine learning method. Accuracy score is computed as the number of instances being correctly predicted over all instances. It gives a good intuition measure of the performance. However, when the cost of a miss-classification is large, it will not be a good indicator of the overall performance. For instance, if our classifier is used for predicting disease, the cost of miss-classification is high. In our example, the cost of classifying a small portion of the samples to the wrong class will not be significant. Thus, accuracy is a useful indicator in evaluating the overall performance.

Feature + Model	Accuracy	F1 macro	Precision macro	recall macro
Unigram + NB (baseline)	0.58 (+/- 0.02)	0.52 (+/- 0.01)	0.55 (+/- 0.01)	0.57 (+/- 0.01)
Unigram + POS+ NB	0.87 (+/- 0.00)	0.76 (+/- 0.01)	0.83 (+/- 0.01)	0.73 (+/- 0.01)
Unigram + POS+ MaxEntropy	**0.90 (+/- 0.00)**	**0.83 (+/- 0.00)**	**0.85 (+/- 0.00)**	0.81 (+/- 0.00)
bigram + POS + NB	0.85 (+/- 0.01)	0.70 (+/- 0.02)	0.82 (+/- 0.02)	0.66 (+/- 0.02)
bigram + POS + MaxEntropy	0.87 (+/- 0.01)	0.77 (+/- 0.02)	0.82 (+/- 0.02)	0.74 (+/- 0.02)
lexical compound + NB	0.83 (+/- 0.01)	0.69 (+/- 0.02)	0.75 (+/- 0.02)	0.67 (+/- 0.02)
lexical compound + MaxEntropy	0.83 (+/- 0.01)	0.68 (+/- 0.01)	0.77 (+/- 0.02)	0.65 (+/- 0.01)
lexical compound + SVM	0.81 (+/- 0.01)	0.71 (+/- 0.01)	0.71 (+/- 0.02)	0.70 (+/- 0.01)
lexical_neg_pos + NB	0.82 (+/- 0.01)	0.67 (+/- 0.02)	0.73 (+/- 0.03)	0.65 (+/- 0.02)
lexical_neg_pos + MaxEntropy	0.83 (+/- 0.01)	0.66 (+/- 0.02)	0.77 (+/- 0.02)	0.63 (+/- 0.01)
lexical_neg_pos + SVM	0.74 (+/- 0.03)	0.67 (+/- 0.02)	0.67 (+/- 0.02)	0.73 (+/- 0.02)
word2vec + NB	0.74 (+- 0.02)	0.68 (+/- 0.02)	0.67 (+/- 0.02)	0.74 (+/- 0.02)
word2vec +MaxEntropy	0.87 (+-0.00)	0.78 (+/- 0.01)	0.83 (+/- 0.00)	0.75 (+/- 0.01)
word2vec +SVM	0.82 (+/- 0.01)	0.76 (+/- 0.01)	0.74 (+/- 0.01)	**0.82 (+/- 0.01)**
GLoVe +NB	0.69 (+/- 0.01)	0.59 (+/- 0.00)	0.58 (+/- 0.00)	0.60 (+/- 0.00)
GLoVe +MaxEntropy	0.80 (+/- 0.01)	0.50 (+/- 0.01)	0.68 (+/- 0.07)	0.53 (+/- 0.01)
GLoVe + SVM	0.70 (+/- 0.01)	0.59 (+/- 0.01)	0.59 (+/- 0.01)	0.61 (+/- 0.01)
Bert+CNN	**0.9300**	–	–	–

However, since our data is imbalanced, there is more information needed in order to validate it as the method with the best performance.Next, we will evaluate the performance of the different combinations of features and classifiers in supervised machine learning methods. Since our data is imbalanced, F1 is an important indicator to find the balance between precision and recall. As we can observe from the result table, unigrams with pos using MaxEntropy produces the highest accuracy. And, unigrams with pos using NB, bigrams with pos tag classified using MaxEntropy and word2vec classified with MaxEntropy produce the second highest accuracy. The four of them also produce the four highest F1 macro scores. With the higher accuracy and F1 scores, they have a relatively better performance even with the imbalanced data. To further conclude they have the best performance, we also need to investigate in precision and recall scores, and the cost of them. As we can observe, those four classifiers has the four highest precision scores. And, unigram with pos tag using MaxEntropy has the second best recall macro scores, while the other three classifiers have a relatively high recall score. Since the recall scores are relatively high (above 0.5), and in our case the cost of classifying an actually positive/negative reviews to the wrong class is not high, we can conclude that overall, in supervised machine learning method, unigrams with pos tag using NB, unigrams with pos tag using MaxEntropy and bigrams with pos tag and word2vec features using MaxEntropy classifier generally have a better performance. Since MaxEntropy uses conditional probability, we do expect that it will work well on connected features, such as bigram and word embedding. Since the standard deviation is quite low (+/- 0.01), it also indicates that those two classifiers are robust. And, since it optimize the overall entropy, we expect that it will have a better performance in unigrams feature. Considering different features, word2vec and bag of words using unigram with pos tag have the better performance. Since unigrams of pos with MaxEntropy have the overall highest scores of f1, accuracy, precision, and second best score of recall, we conclude that unigrams with pos using MaxEntropy have the overall best performance in machine learning methods.

Another finding we have is that MaxEntropy generally outperforms than SVM and Naive Bayes. After researching, we realize that SVM does not rely on the distribution of the data while Logistic Regression relies on the distribution of the data. In our case, the positive class takes the majority so our classifiers tend to classify data as positive. This judgment can also be supported by the f1 macro, precision macro and recall macro. From those indicators, we can see that the majority has high accuracy but low f1, precision recall. Thus, we can see that imbalanced data makes our classifiers favor the main classes and make logistic regression outperforms the SVM.

Comparing with Naive Bayes, MaxEntropy generates a better performance since as a generative model, Naive Bayes computes the probability independently while MaxEntropy performs optimization of the overall entropy, with each feature conditional to each other.

Then, when evaluating the performance of rule-based features(lexical-based features), from our experiments we observed that lexical with the combination of negative and positive scores with SVM has the worst performance among the lexical-based features with classifiers. First of all, it has the lowest accuracy score among the lexical-based features combined with different classifiers, although other lexical features combined with classifiers shows accuracy scores that are at the average level. It indicates that the ratio of correctly predicted observation to the total observations is pretty low for lexical-based pos_neg feature combined with SVM specifically. However, as it is mentioned before since our data is imbalanced, only observing accuracy is not enough because accuracy is a great measure only when the data-set is symmetric. From our result table, we can see that its F1, Precision, and recall scores are at a average level which is consistent with every lexical-based feature combined with all three classifiers. Therefore, we could conclude that lexical-based feature has overall average performance while lexical-based pos_neg feature with SVM specifically has the lowest accuracy. Therefore, we could conclude that negative and positive score combined feature is not a good measurement compared to the compound score towards the sentiment when combined with SVM model. The low accuracy score that comes from neg_pos score using SVM is probably because there is no clear hyper-plane to segregate the two classes within the data, more specifically, when we are using both negative and positive score as features.

Another interesting finding is that word2vec outperforms GLoVe. Additionally, GLoVE has the worst f1, precision and recall scores. GLoVe with NB also has the lowest accuracy. There are two reasons. First, we used twitter corpus to pretrain GLoVe for computation efficiency. However, there is obvious different between twitter's sentiment with Amazon's food review. For the product reviews in our data , people tend of find the positive reviews more useful compared with negative ones. Additionally, the GLoVe takes the global information into consideration. However, in our cases, products are different and it makes more sense for our classifier to focus on neighbouring text compared with the global text. And if we on the global information, our classifier favours the positive more.

Comparing to the previous work, in 2011, Xia et al. performs classification with the feature of unigrams with pos tag using the three classifiers as we used in this project. (Xia et al., 2011)Since they performs classification on the reviews of all different kinds of product, such as movie, book, DVD, electronics, and Kitchen, we decide to compare the most relevant product which are kitchen products. Their results are, for Naive Bayes, 82.40% of accuracy, for MaxEntropy, 82.50% of accuracy, and for SVM, 82.10% accuracy. (Xia et al., 2011) Comparing to our result of 87% for NB, 90% for MaxEntropy, we have a better performance than theirs. One guess is that we have a much larger training set which contains $50,000$ data, whereas they only have $2,000$ data set. Thus, we will create a more comprehensive training set specific to fine food reviews. Further, in 2010, Xia and Zong introduces the bigram with pos-tag method as we introduced above in their research paper, which they refers to as GWR-Bi method. (Xia and Zong, 2010) In their research, they evaluate reviews of movies and e-products, using SVM and Naive Bayes. For comparison, we only consider NB classifier, which they get 85.45% accuracy of movies, and 67.50% accuracy of e-products. (Xia and Zong, 2010) Compared with our bigrams with pos using NB, we have an accuracy of 85%,which is similar to their work on movies. In general, our system with bigram and NB has a similar performance with the previous work and for the unigram outperforms previous work.

7 Conclusions

7.1 Major Findings

Overall, we have the following findings,

- The BERT with CNN model has the highest accuracy scores among supervised machine learning and deep learning methods.
- In supervised machine learning methods, unigrams with pos using NB, bigrams with pos using MaxEntropy generally have better performance overall. Unigrams with pos using MaxEntropy generates the best performance.
- Maximum Entropy out-performs other classifiers.
- Applying Multinomial Naive Bayes significantly improves the performance of bag of word features.
- For word embedding, word2vec outperforms GLoVe.
- Lexical methods generates a high accuracy score (above 0.5) but relatively lower f1, precision and recall scores and the reason can be the imbalanced dataset.
- Due to imbalanced dataset, we have relatively high accuracy but relatively low F1, recall and precision.

7.2 Weakness & Future Work

Though our system build different classifiers with comprehensive distinct features to find the best performance, there are some improvements that we can make to increase the performance.

First, since our data is imbalanced, which means that we have about 80% of positive reviews and the remaining 20% of negative reviews, it may result in the higher accuracy score of the classifiers. With the more positive data in both the training and the testing set, it is easier for the classifier to classify into the correct category due to the limited training on negative review and the limited amount samples in testing set of negative review. And, the misclassification of the minority class will be greater than the misclassification of the majority class. It means that the classifier will have a bias over the majority of the class, which in our case, is positive review. To solve this problem, in supervised machine learning approach, in which we use the three classifiers, we can use data level approach, algorithm-based approach, or a hybrid of both. For data level approach, we can down-sampling the data to balance the data before training. In algorithm-based approach, we can give different weights to different classes when classifying. Since reviews are sequential, in deep learning methods, we can use RNN instead of CNN to achieve a better performance.

Next, in this project, we only consider the performance of a single feature with different classifiers. We do not investigate into the cases where we can have ensembles of features, such as the mixture of unigram pos tag and bigram with postag. However, as many other research shows, using different ensembles of features with the classifiers can result in a better performance of the classifier. In the future, we can create ensembles of our specific set of features to investigate whether it can create a better performance.

Also, we are not considering the neutral sentiment as we think it is generally not that useful and applicable to our project. However, in the future, we could expand our sentiment class to include neutral and more sentiment such as relatively positive ,relatively negative and neutral to do the multi-class classification with considering a larger data-set. In this way, we can make our system much stronger.

8 Appendix

Roles

Xin: In this project, I worked on building the baseline model of unigram bag of words, with naive bayes classifier. And, I worked on building bigrams with pos tag bag of word feature and editing the unigrams with pos tag bag of word feature. Due to the memory excess, bigrams with pos tags are only trained with Naive Bayes and Maximum Entropy. In the final report, I worked on writing abstract, formulation, features – bigram with pos tag, and collaborated in results, findings and weaknesses and future works.

Jing: In this project, I worked on building the word2vec, GLoVe models, BERT with CNN. I also designed the classes(dataprocessor, features, classifiers and engine) and cleaned and reconstructed the codes to eliminate repetitive codes. In the final report, I am responsible for the overview of approach, word2vec, GLoVe features part, Deep Learning CNN part, experiment setup. I collaborated on the survey of related works, results analysis and conclusion. We collaborated on proofreading and experimenting.

Jingyao: In this project I worked on building the lexicon-based feature which are compound lexicon feature and neg_pos lexicon feature. I also worked with Alex to design the first draft of data-processor class. I collaborated on doing experiments and collecting results. In the final report, I generated the word cloud and the plot for the data-set, I did part of the data-set analysis under Experiment & Analysis section and the lexicon-based and the SVM part under Approach section. I also collaborated on writing the survey of related works, analysis of result and weaknesses & future work part. In the end, I did the overall proofreading and improved our report.

Alexandre: For the coding side of things, I implemented the unigram/POS tag feature for this project and experimented with different parameters for comparison purposes (such as using different POS tags, size of the bag-of-words, etc). When it comes to the final report, I took care of writing the introduction, Unigram with POS section as well as the Naive Bayes subsection under the Models section.

References

1. Baad, D. (2020, April 06). Sentiment Classification using Word Embeddings (Word2Vec). Retrieved December 13, 2020, from https://medium.com/swlh/sentiment-classification-using-word-embeddings-word2vec-aedf28fbb8ca

2. Deho, B. O., Agangiba, A. W., Aryeh, L. F., amp; Ansah, A. J. (2018). Sentiment Analysis with Word Embedding. 2018 IEEE 7th International Conference on Adaptive Science amp; Technology (ICAST). doi:10.1109/icastech.2018.8506717

3. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (cite arxiv:1810.04805Comment: 13 pages)

4. Hamdan, H., Bellot, P., amp; Bechet, F. (2015). Sentiment Lexicon-Based Features for Sentiment Analysis in Short Text. Research in Computing Science, 90(1), 217-226. doi:10.13053/rcs-90-1-17

5. Kim, R. (2018, February 13). Another Twitter sentiment analysis with Python - Part 10 (Neural Network with... Retrieved December 13, 2020, from https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-10-neural-network-with-a6441269aa3c

6. Malik, U. (n.d.). Text Classification with BERT Tokenizer and TF 2.0 in Python. Retrieved December 13, 2020, from https://stackabuse.com/text-classification-with-bert-tokenizer-and-tf-2-0-in-python/

7. Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - EMNLP '02. doi:10.3115/1118693.1118704

8. Pennington, J., Socher, R.,& Manning, C. (2014). Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). doi:10.3115/v1/d14-1162

9. Ribeiro, M., Singh, S., & Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations. doi:10.18653/v1/n16-

10. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems (p./pp. 5998–6008)

11. Vu, L., Le, T.(2017). A lexicon-based method for Sentiment Analysis using social network data, in: Proceedings of the International Conference on Information and Knowledge Engineering (IKE'17), Las Vegas, Nevada, USA. pp. 10–16.

12. Xia, R., Zong, C., & Li, S. (2011). Ensemble of feature sets and classification algorithms for sentiment classification. Information Sciences, 181(6), 1138-1152. doi:10.1016/j.ins.2010.11.023

13. Xia, R., Zong, C. (2010). Exploring the Use of Word Relation Features for Sentiment Classification. Coling 2010, Poster Volume, 1336-1344.

14. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical Attention Networks for Document Classification. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. doi:10.18653/v1/n16-1174