Sentiment Analysis and Opinion Mining on YELP Restaurant Reviews

Final Report

CS 585, UMass Amherst, Fall 2017

Jie Song, Chen Pan, Ruisi Zhang

Abstract

This project aims to predict the rating for a restaurant based on the users' review texts alone. We compare and evaluate 4 sentiment analysis/machine learning algorithms (Naive Bayes, Logistic Regression, K-NN, and SVM) based on Word2Vec and Doc2Vec models. The experiment is conducted on the yelp challenge dataset Yelp Challenge Dataset 2017. We figure out that Doc2Vec can produce better predictions on clustering, while it might be computationally unrealistic to be applied in practical sentiment prediction. We also find that for sentiment analysis on the reviews, we can get better results by applying SVM.

1. Introduction

It is one of our crucial information-gathering behaviors to find out what other people think when we need to make a decision or choose from different options. Sentiment analysis and opinion mining is a type of Natural Language Processing that examine opinions, attitudes, or emotions from written languages such as comments, reviews or tweets. Nowadays, as the worldwide network grows so fast, many web sources such as forums, blogs, social networks, and content-sharing services help people share lots of useful information. In this project, we focus on the restaurant reviews and analyze the data from Yelp review texts. The aim of this project is to predict the rating for a restaurant based on the user's' review texts alone. We will compare and evaluate 4 sentiment analysis and machine learning algorithms - Naive Bayes, Logistic Regression, KNN, and SVM. All the three machine learning methods - Logistic Regression, KNN and SVM methods are based on Word2Vec and Doc2Vec, which are new study fields in sentiment analysis.

2. Literature Review

Nowadays, sentiment analysis becomes very beneficial for businesses, governments, and individuals. However, there are several challenges in sentiment analysis [10]: the sentiment of an opinion word may be changed (negative or positive) in different situations; people always express the same opinion in different ways, while most traditional text processing relies on the facts that merely small differences between texts would not affect the analysis results; it is easy for human to understand the combined different opinions in one sentence, but it is hard for computers to parse; it is hard to understand a short piece of text in the informal medium without other sources of information like images, videos or the previous context. Paper [7] also points out that both the sentiment review structure and sentiment techniques would affect the sentiment analysis result.

There are three main research fields in sentiment analysis: sentiment classification, feature-based sentiment classification and opinion summarization [10]. In this project, we only focus on the sentiment classification, which classifies entire documents (user's review texts) according to the opinions towards certain objects (restaurants). In papers [2], [7] and [10], the authors compare ten different algorithms (NLP or machine learning methods), such as Naive Bayes, Part of speech (POS), n-gram, Maximum Entropy, KNN, and SVM, etc., and presented the prediction accuracy for each method. However, none of the experiments use the dataset from Yelp, which is one reason that we choose the Yelp restaurant reviews as our datasets. According to their results, we will choose 3 algorithms with high accuracy performance as our experiment techniques - Naive Bayes, KNN, and SVM. Besides, we choose Logistic Regression as the contrast for machine learning classifiers. The sentiment analysis can be divided into three levels [9]: document level [11], sentence level [4], word/term level [3] or aspect level [12]. In this project, we would do the analysis in different levels according to different sentiment analysis techniques: word level for Naive Bayes and word level and document level (each review can be treated as a document) for Logistic Regression, KNN, and SVM.

Paper [5] is among the earlier research about sentiment analysis on restaurants reviews. They extract their corpus from Citysearch New York. This paper uses unigram model to capture sentence patterns, aiming to use the free-text of the reviews to improve the recommender systems. They show that using information from plain text is very helpful in predicting review

scores. Their methods combine machine learning, sentiment analysis techniques and natural language processing to classify sentences as positive, negative, neutral or conflict. In our work, we consider not just determining whether a review is positive or negative, but attempting to infer the numerical ratings, such as “two stars” or “five stars”. We can also extend the model to applying bigram, trigram or n-gram, hoping to overcome the limitation of the unigram model.

The word2Vec and related models are a group of models used to generate word embeddings. This application created by Mikolov et al. [8] at Google has attracted a great amount of attention in recent years. This model is a two-layer neural network that takes a large corpus of text as input and trained to reconstruct linguistic contexts of words – numeric vectors of several hundred dimensions. Word vectors that share common contexts in the corpus are located in close proximity to one another in the space. Compared to earlier sentiment algorithms, this method has the advantage of capturing the context of words, while in the meantime reducing the size of the data [6][8]. Doc2Vec is an extension inspired by and based on word2Vec. The vectors produced by doc2Vec not only represents words but the entire sentences and documents (in our example, reviews). Intuitively, doc2Vec has an advantage over word2Vec that it provides more context information. We would report our KNN and SVM classification results using doc2Vec to get the feature vector for each review in this progress report, and in the future work on word2Vec as a comparison.

In this project, we analyze 133198 reviews as our training set, and the vocabulary is 150505 words. For unigram, the total using space for all reviews is O(133198*150505*N) (N is the average number of the words in each review). And for bigram, trigram or n-gram, the required space is even more. Using Word2vec, the total using space for all reviews is reduced to O(133198*WV*N) (WV is the word dimensions by using Word2vec, ~100 - 500 dimensions). Inspired by Word2vec, we also decide to use Doc2vec - document-level sentiment analysis, - in this project to get the feature vector for each review. In this way, the running space significantly reduced to O(133198*WV) (WV is the word dimensions by using Doc2vec, ~100 - 500 dimensions).

3. Experiments and Methods

3.1 Approach

We plan to find the best model to predict the review stars for restaurants. The experiments can be divided into 3 steps. First, we will do texts processing, for which we will use standard Python libraries to remove capitalizations, stop words and punctuations. Second, we will do feature extraction to extract useful features from the review corpus and build a feature vector for each review. Particularly, in this project, we will extract features to vector representations by using Word2Vec and Doc2Vec. In this step, we will compare different extraction methods, like unigram, bigram or n-gram. Third, we will build several analysis models with prediction performance (prediction accuracy/MAE) comparison for evaluation. Paper [1] provides the similar analysis on Yelp reviews, however, we will use different algorithms and models to run the experiments.

3.2 Datasets

For this project, we use Yelp Challenge Dataset 2017, the competition round 10 (https://www.yelp.com/dataset/challenge, 2017). All the data are provided in JSON format. This data does not need human annotation since we have the users' star rating as our true label. The raw dataset is around 5.79 gigabytes. We will proceed two of the total six JSON files, which are review.json (3.82GB) and Business.json(132.3MB).

Our project aims to process the reviews which are relevant to restaurants. So at first, we filtered out all the restaurant information in the Business.json. During which, we also completed the basic data cleaning by deleting the irrelevant attributes in the file, like the restaurant's' location, open hours, etc. The total number of restaurants in this dataset is 51613, and the size of the cleaned restaurant dataset is 42.2 MB. In addition, the number of relevant reviews is 4736897, which are 1.93 GB. The attributes we have in the reserved files are average stars of the restaurants, review_id, restaurant_id, review text and star of the reviews.

During the implementation of the project, we notice that it occurs frequently that the memory we need for the algorithms exceeds the capacity of our laptop. We randomly picked 5 percent of the reviews from the dataset so as to explore more of the algorithms. And we split the sample dataset into train set and test set by the proportion of 70 - 30. Among all the training data, the numbers of reviews under 1-5 star are 16364, 13227, 19465, 36711 and 47431. We treat each review as a "document". After removing all the stop words and punctuations, the number of

tokens under each star are 1068598, 961183, 1354568, 2268917, 2321600. The total number of unique words in training corpus is 150505.

3.3 Scope

Yelp restaurant reviews contain detailed information about each reviewer's personal experience. The plain texts not only show their sentiment within the opinions but also reflect how satisfactory they are with the restaurant. According to paper [1], [2], [7] and [10], we will compare and evaluate 4 algorithms – Naive Bayes, Logistic Regression, SVM and KNN for yelp review star rating prediction. Our models extract different feature vectors from user review texts to develop star predictions based on vector representation for each word or each review. These predictions are then compared with actual ratings. We would like to see which model will achieve the best accuracy. And more specifically, we want to explore how the preprocessing of the dataset, or how performance metrics influence the performance of each model.

3.4 Software Systems

We implement our code in Python. NLTK will be used for text processing such as tokenization and lemmatization. We could draw useful information on word relations from NLTK. Numpy and Scipy will be used for numerical calculations and matplotlib will be used as a graphing tool. Machine Learning functions and utilities in Python from Scikit-Learn will also be immensely used for its built-in classifiers and model evaluation capabilities. Besides, we use Gensim for Word2Vec and Doc2Vec implementation, which is a robust open-source vector space modeling and topic modeling toolkit implemented in Python.

3.5 Algorithms

3.5.1 Baseline Model

There are three possible baseline predictions to obtain the ratings. One uses the average star rating of all reviews to predict every rating, regardless of who writes them, which restaurant it is targeting at, and when it is written, etc. This is the simplest prediction but may not be very accurate. Some of the authors may write good reviews generally, some may not. There are all kinds of restaurants, and the authors may tend to tailor their reviews according to different attributes of those restaurants. Therefore, there are another two baseline predictions, in which

one predicts using the average star rating by the specific user, and another predicts using the average star rating by the specific restaurant. We will perform our baseline model using the average star rating by the specific restaurant since it is easier to compute the average star ratings of each restaurant and it's comparable to the other prediction using the average star ratings of a particular Yelp user. The baseline prediction can then be compared with the proposed algorithms in terms of accuracy improvement.

3.5.2 Naive Bayes Classifier

Naive Bayes is a famous and popular method. It can do text classification with low storage requirements and the time consumption is also quite low. It is robust to irrelevant features and can perform well in domains with many equally important features. So, we take it as one of the algorithms to do the review classification. We used the Laplace smoothing to avoid zeros.

The Naive Bayes Function is as follows:

$$C_{MAP} = \arg\max_{c} \hat{P}(c) \prod_{i} \hat{P}(w_i|c)$$

$$\hat{P}(w_i|c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} count(w, c) + |V|}$$

3.5.3 Vector Representations

3.5.3.1 Word2Vec

The machine learning algorithms used in this project - Logistic Regression, KNN, SVM - require the input to be represented as feature vectors. The entire input should be transformed into a matrix, with reviews in the rows and the context information in the columns. Each feature vector represents one review, with similar reviews having similar feature vectors.

In order to complete this job, first, we use Word2Vec to convert each word from the corpus into a high dimensional vector. For this neural network inspired models, the key parameters of training command are: choice of training algorithm, default is continuous bag-of-word (CBOW) model; dimensionality of feature vectors, set to 100; maximum distance between the current and predicted word within a sentence, set to 1 corresponding to bigram, 2 corresponding to trigram, and 3 corresponding to 4-gram. In CBOW a word is used as the output

and its context as input, whereas in the Skip-gram model it is done the other way around. The advantages of CBOW model include the increased training speed, a higher quality of the representation for frequent words, etc. It fits our intention to use context information to generate high-quality feature vectors. We would also like to compare the results for different window size and see how it affects the accuracy of our prediction.

A well-known framework for learning the word vectors is shown in Figure 1 [14]. In this framework, a context of three words is used to predict the fourth word. The input words are mapped to columns of the feature matrix. The concatenation or sum of the vectors is then used as features to predict the output word. High-quality word vectors should be mapped into a vector space such that semantically similar words have similar vector representations.

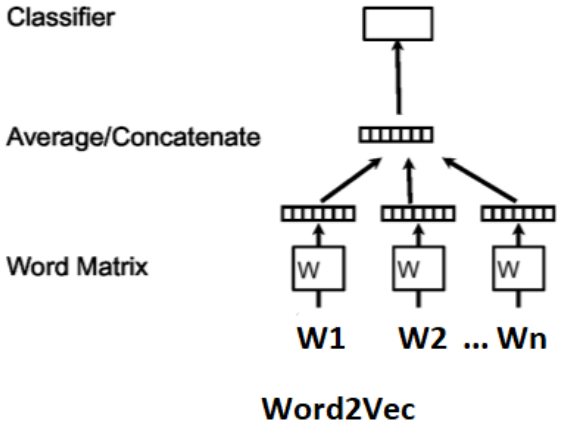

Figure 1. Framework for Word2Vec [14]

There are two ways to feed the Word2Vec model, aiming to generate high-quality feature vectors for each word. One is using only the review texts in training data, the other is using both training data and test data. It turns out that these two ways have little difference on the prediction of stars. Both ways generate comparably high-quality vectors. The reason may lie in the similar context in both training data and test data. We divide them to perform the analysis, but they have essentially the same review context. Therefore, I decide to feed only training data into Word2Vec model.

To get a sense of the quality of the word vectors generated, the 5 most similar words to ‘seafood’ using 4-gram are listed here with their cosine similarity respectively. These five words are: ‘lobster’ (0.625), ‘clams’ (0.613), ‘pasta’ (0.594), ‘crab’ (0.588), and ‘oyster’ (0.585). These words are indeed very similar to ‘seafood’, which shows that the word vector representations

capture some essence of the words semantically. On the contrary, ‘seafood’ and ‘spoon’ are more distant, with their cosine similarity being 0.165.

Our next job is to combine the word vectors and generate vectors for each review as a whole to facilitate the classification task. Two methods are compared. The first one is the simple average of each word vector composing a review. The other method is to compute a weighted average using TF-IDF (Term Frequency - Inverse Document Frequency) score. Each weight gives the importance of the word with respect to the corpus, which might be a better solution over the first one. However, the results show that there is little difference between these two methods. Therefore, only the simple average is adopted in presenting the results. After averaging, the vector representations of all reviews are ready to be fed into the classifiers.

3.5.3.2 Doc2Vec

In short, Doc2vec takes in a corpus and churns out vectors for each of those words. Doc2vec represents not only words (Word2vec), but also the entire sentences or documents. It produces a fixed-length vector for document representations and proceeds to run all standard classification algorithms, which makes it very powerful for sentiment analysis. In Doc2Vec framework (Figure 2), every word is mapped to a unique vector shared by all the corpus as Word2Vec does [14]. Meanwhile, every paragraph is also mapped to a unique vector, which is only shared by the words in the same document. The next word is predicted based on paragraph vector and word vectors (averaged or concatenated), and the weights of the model updates in each iteration until the model is converged. After being trained, the paragraph vectors can be used as features for the paragraph [14].

We use gensim, which has a readable implementation of Doc2Vec (and Word2Vec). The method is presented here following [13]:

1. Feeding Data to Doc2Vec. Each review should be format to: [['word1', 'word2', 'word3', 'lastword'], ['label1']].
2. Build Doc2Vec model by using the training dataset. Build different models by selecting different vector size (100-1000) and word window (1-10).
3. The vector representations for testing datasets can be obtained by calling infer_vector() function based on the model built by training data.
4. Evaluate the model by using different machine learning classifiers.

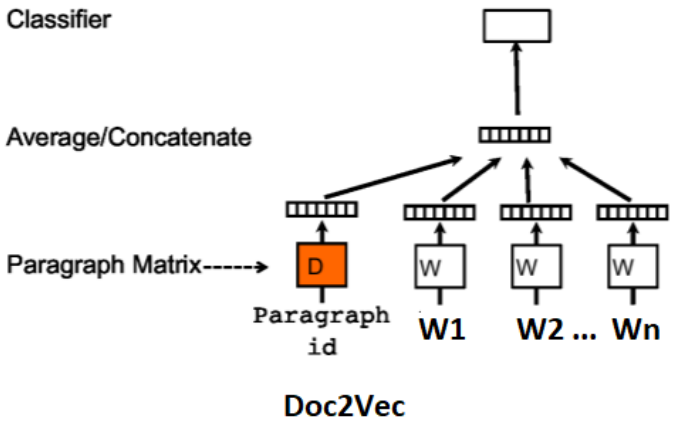

Figure 2. Framework for Doc2Vec [14]

3.5.4 Machine Learning Methods

3.5.4.1 Logistic Regression

Logistic regression is a probability discriminative classifier. Like all regression analyses, the logistic regression is a predictive analysis. It does the appropriate regression analysis to conduct when the dependent variable is binary. In the binary case, it directly models the decision boundary using a linear function. In our project, we have multi-classes. So we further defined the "binary" using "one versus rest" scheme. That means for each class, the samples of that class are defined as positive samples and all other samples as negatives. At classification time, LR is faster than Naive Bayes.

Logistic regression can also be extended to the multiclass case:

$$P(Y = c|\mathbf{x}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x} + b_c)}{\sum_{c' \in \mathcal{Y}} \exp(\mathbf{w}_{c'}^T \mathbf{x} + b_{c'})}$$

The classification function is:

$$f_{LR}(\mathbf{x}) = \arg\max_{c \in \mathcal{Y}} P(Y = c|\mathbf{x})$$

3.5.4.2 K-Nearest Neighbors (K-NN)

The K-NN classifier is a non-parametric classifier that simply stores the training data D and classifies each new instance x using a majority vote over its set of k nearest neighbors NK(x) computed using any distance function d: R^D x $R^D \longrightarrow R$. It maximizes the following function:

$$f_{KNN}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{i \in \mathcal{N}_K(\mathbf{x})} \mathbb{I}[y_i = y]$$

Use of K-NN requires choosing the distance function d and the number of neighbors k. In general, K-NN can work with any distance function d satisfying non-negativity $d(x, x') \geq 0$ and identity of indiscernibles $d(x, x) = 0$. Alternatively, K-NN can work with any similarity functions satisfying non-negativity $s(x, y) \geq 0$ that attains its maximum on indiscernibles $s(x, x) = \max_{x'} s(x, x')$. The output depends on whether K-NN is used for classification or regression: for K-NN classification, the object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors; while for K-NN regression, the output is the average of the values of its k nearest neighbors. In this project, we use K-NN classification for prediction outputs by using Scikit-Learn library, for which the inputs are Doc2vec vectors.

3.5.4.3 Support Vector Machine (SVM)

Support Vector Machine is a model that is very easy to train, as simple as naïve Bayes. It also has strengths that naïve Bayes and other models don't have. For example, SVM doesn't assume independence among words, which are often violated. It has a regularization parameter, which reduces overfitting of the model. It uses the kernel trick, which cleverly increases the model's capacity and allows non-linear decision rules. In addition, it doesn't have a lot of hyperparameters, like neural network.

We classify the review data into five stars, so we have five classes. After using the package doc2vec from gensim to perform document embedding, we obtain a dense feature vector of length 100 for each review. Then we perform this kernel hinge loss classification method using the functions from Scikit-Learn. The function we try to minimize is as follows:

$$w^* = \arg\min_{w} \sum_i L_{hinge}(y_i, \sum_j \alpha_j k(x_i, x_j)) + \lambda \alpha^T K \alpha$$

Where $L_{hinge}(y, f) = \max(0, 1\text{-}yf)$. It is a loss function that penalizes when it makes a mistake. For any two inputs x and x', basis expansion can be denoted by $\Phi(x)\Phi(x')$. The value of

the kernel k(x, x') is the dot product of x and x' after they have been transformed into some other feature space. A necessary condition for K is that it must be positive semi-definite.

There are some well-known kernels, including linear, polynomial, Gaussian RBF kernel, etc. We apply these three kernels and compare their results. We implement the SVM classifier in Scikit-Learn library for accuracy evaluation, based on Word2Vec and Doc2Vec vectors.

4. Results

4.1 Baseline Model

As stated above, the baseline is based on the user's stars given in the training set. We calculate the average stars of the reviews for each restaurant in the training set. The results are used as predictions for the reviews of the same restaurant. The accuracy is 23.054%. It is quite low, as expected.

4.2 Naive Bayes Classifier

We train Naive Bayes Classifier on the review training set. The classification accuracy we gain from the evaluation of the test set is 58.916%. Naive Bayes seems a very simple but effective way to do sentiment analysis. Compared to our baseline, it improves a lot.

4.3 Word2Vec

As we stated in the algorithms section, choice of training algorithm is continuous bag-of-word (CBOW) model; dimensionality of feature vectors is set to 100; maximum distance between the current and predicted word within a sentence is set to 1 corresponding to bigram, 2 corresponding to trigram, and 3 corresponding to 4-gram.

The training process of various machine learning models with hyperparameters (KNN and SVM) can be separated into several steps. First, the whole training data is randomly partitioned into training set (80%) and validation set (20%). Then fit each model to the training set and evaluate on validation set. Pick the model with the best score on validation data. Finally, retrain that model on all data, and return it.

The natural evaluation metric is accuracy, which shows the percentage of correctly labeled stars among all the reviews. However, there is some shortcoming in it. If the predicted label is not the same with gold label, this review is labeled incorrectly. For example, if two

reviews' gold label are both 5 stars, our incorrect prediction may predict it as 4 stars or 1 star. Accuracy will not take the difference in stars into account even if the former is labeled with only 1-star distance rather than 4 stars. Accuracy will not distinguish between these two cases. It is interesting and necessary to look into whether it's the former or latter, which would show a huge difference in the quality of our model predictions. Thus, we adopt a second evaluation metric of mean absolute error (MAE). It calculates the average star distance which would give a sense of how close our prediction is to the gold labels. Results are shown in the following subsections.

4.3.1 Logistic Regression

Results of logistic regression show that the accuracy grows higher when window size is increased. 4-gram may capture more context information than bigram and trigram. Generally, the MAEs are small and less than 1, which shows that our model prediction is quite reliable. There is no huge difference between gold labels and our prediction. It is reasonable that as accuracy increases, MAE decreases.

	Metric	Bigram	Trigram	4-gram
LogReg	Accuracy	0.54287	0.54805	0.55177
	MAE	0.64509	0.62651	0.61815
KNN	Accuracy	0.51497	0.51978	0.51601
	MAE	0.69532	0.68635	0.6915
SVM	Accuracy	0.66453	0.65625	0.65768
	MAE	0.44443	0.45135	0.44823

Table 1. Accuracy evaluation using Word2Vec and three classifiers

4.3.2 K-NN Classifier

During the training process, I train KNN classifier using different numbers of nearest neighbors and pick the best model. It is shown in Figure 3 that the best k for bigram is 140 with accuracy 49.741%, trigram is 110 with accuracy 49.88% and 4-gram is 160 with accuracy 50.327%. Generally, 4-gram performs the best with the highest accuracy.

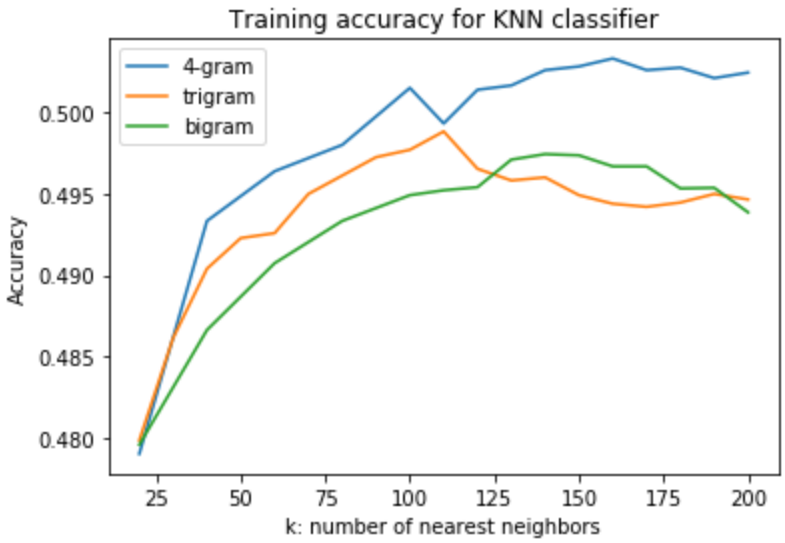

Figure 3. Training accuracy for KNN classifier

Then the models are retrained with the whole dataset. The test accuracies are similar to the validation accuracies.

4.3.3 SVM Classifier

The choice of kernels includes the (Gaussian) radial basis function (RBF) kernel, polynomial kernel, and linear kernel. The training accuracy for linear kernel is slightly lower than RBF (~55% vs. ~57%), and polynomial kernel is surprising low (~35%). After comparing all training accuracies, the best hyperparameter chosen is RBF kernel with C (Penalty parameter C of the error term) = 1 and gamma = 0.01.

When the whole dataset is retrained with RBF kernel, the accuracies increase a lot (~66%), compared to validation accuracies. Results are shown in Table 1. It may show that SVM model with RBF kernel generalizes very well for our data. Comparing all three machine learning classifiers, SVM is the most suitable for our dataset using Word2Vec. Accuracies are high and MAEs are low.

4.4 Doc2Vec

For Doc2Vec, we need to validate 2 parameters - the size of vector dimensions (100 - 1000) and word window (1-10) to get the best results. After first round evaluation, we choose the

size of vector dimensions = 100 as our optimal parameter setting. The selection for the word window is actually the selection of bigram, trigram … n-gram for the analysis. We cannot set window = 0, which is equivalent to unigram for data analysis.

4.4.1 Logistic Regression

Figure 4 shows the result of accuracy evaluation using Doc2Vec for Logistic Regression classifier, according to different word window size. The best accuracy is 37.569% when word window = 2, which uses a trigram model.

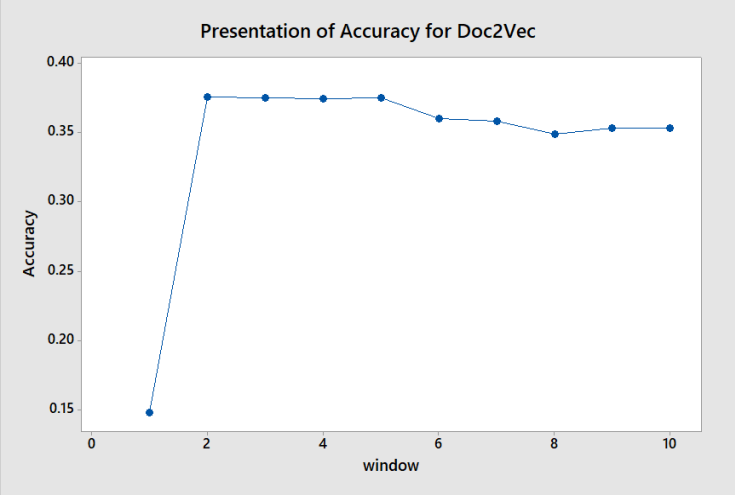

Figure 4. Accuracy evaluation of Doc2Vec using LogReg for Yelp reviews sentiment analysis

4.4.2 K-NN Classifier

As mentioned in Doc2Vec algorithms, we used two Doc2Vec models (vector dimension size = 100 and word window = 2, vector dimension size = 100 and word window = 3) as the document (reviews) vector models. For K-NN classifier, we need to find out the optimal k for best results. Different k value would affect the predictions, which is a hyperparameter we need to train. Figure 5 shows the results for accuracy evaluation of K-NN based on Doc2Vec according to different k values. The best accuracy is 37.884% when k = 3 and Doc2Vec model for trigram (vector dimension size = 100 and word window = 2).

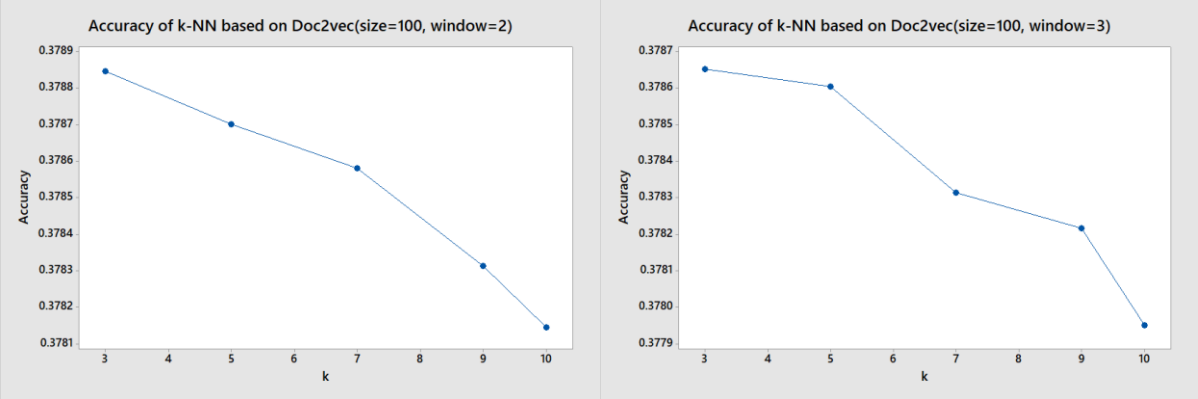

Figure 5. Accuracies of K-NN for Yelp restaurant reviews sentiment analysis

4.4.3 SVM Classifier

For SVM classifier, we also used the same two Doc2Vec models (vector dimension size = 100 and word window = 2, vector dimension size = 100 and word window = 3) as the document (reviews) vector models. As mentioned above, we choose linear, polynomial, Gaussian RBF kernel as the SVM model kernel methods. However, polynomial kernel does not seem to fit our dataset. The best accuracy is 37.911% when kernel = linear and Doc2Vec model for trigram (vector dimension size = 100 and word window = 2). The results for SVM classifier are as following:

Doc2Vec model: size =100, window =2
Doc2Vec SVM:
kernel = linear Accuracy: 0.379114614918
kernel = poly Accuracy: 0.0735476046089
kernel = RBF Accuracy: 0.377513644633

Doc2Vec model: size =100 window = 3
Doc2Vec SVM:
kernel = linear Accuracy: 0.378944815039
kernel = poly Accuracy: 0.0600121285628
kernel = RBF Accuracy: 0.377707701637

5. Discussion and Future Work

The first eight rows of Table 2 show the results of our project, comparing the accuracy obtained for 5 different algorithms - baseline, Naive Bayes, Logistic Regression, K-NN, and SVM. Word2Vec method shows very good performance. The highest accuracy for Word2Vec is about 66%, comparable to those obtained in previous research [1]. Consider that we do the 5-label (5-star ratings) classification analysis instead of the polarity analysis (labeled as positive or

negative), it is much harder to classify 5 classes rather than 2 classes. We are pretty satisfied with our results. Besides, it seems that Naive Bayes is actually a very simple but an effective and convenient method for sentiment analysis, which is not easily beaten by many complex methods.

Method	Unigram	Bigram	Trigram	4-gram
Baseline	0.23054	-	-	-
Naive Bayes	0.58916	-	-	-
Word2Vec + LogReg	-	0.54287	0.54805	0.55177
Word2Vec + K-NN	-	0.51497	0.51978	0.51601
Word2Vec + SVM	-	0.66453	0.65625	0.65768
Doc2Vec-train + LogReg	-	0.14802	0.37569	0.37511
Doc2Vec-train + K-NN	-	0.20150	0.37884	0.37865
Doc2Vec-train + SVM	-	0.24628	0.37911	0.37894
Doc2Vec-all + LogReg	-	0.93401	0.99896	0.99935
Doc2Vec-all + K-NN	-	0.99728	0.99973	0.99995
Doc2Vec-all + SVM	-	0.99356	0.99922	0.99997

Table 2. Accuracy Comparison of 8 methods

Regarding Doc2Vec, accuracies are much lower (highest is below 40%) and all the three machine learning methods yield similar results, although we expected that Doc2Vec would give higher accuracies and outperform Naive Bayes. The reason could be that the vector representations for testing data were obtained from the model built solely from training data, but not from the feature extractions of testing data themselves. And the vector representation for each document is a 100-dimension vector, which is too coarse for the feature extractions. Since Doc2Vec is an unsupervised method (the training process would not use the sentiment labels for each data), we built another Doc2Vec model by using training and testing data together. This model is expected to capture much more context information and give more accurate and high-quality feature vectors. The results are shown in the last three rows in Table 2, where "Doc2Vec-all" means training the Doc2Vec model using both training and testing data and "Doc2Vec-train" means training the Doc2Vec model using only training data. The results are incredibly better than others, which almost reach 100% correctness. However, it seems to be not feasible in the real world, where we want to build a model that can well predict brand new and untouched data. According to the results, we can say that the Doc2Vec may not be good for "prediction" (especially for new data), but it should be a very good method to "clustering" data. One of the

biggest advantages of Doc2Vec is to compare the similarity of different documents. Previous research reported that Doc2Vec worked well and gave a 32% relative improvement in terms of error rate for document distance analysis, compared to the bag-of-words or bag-of-n-grams models or even more advanced methods (such as Recursive Neural Network) [14]. Thus, Doc2Vec would not be a priority method for our project because of resource-consuming.

In the future, we want to apply supervised learning algorithms that are more suitable for high dimensional feature vectors. Theoretically, KNN tends to work well in low dimensions but does not scale well to high dimensions. It is shown in our results that KNN predictor perform worse than SVM. We would like to try, for example, Neural Network, and make comparisons. It is also interesting to incorporate other methods to extract features in the reviews and see if there is any improvement.

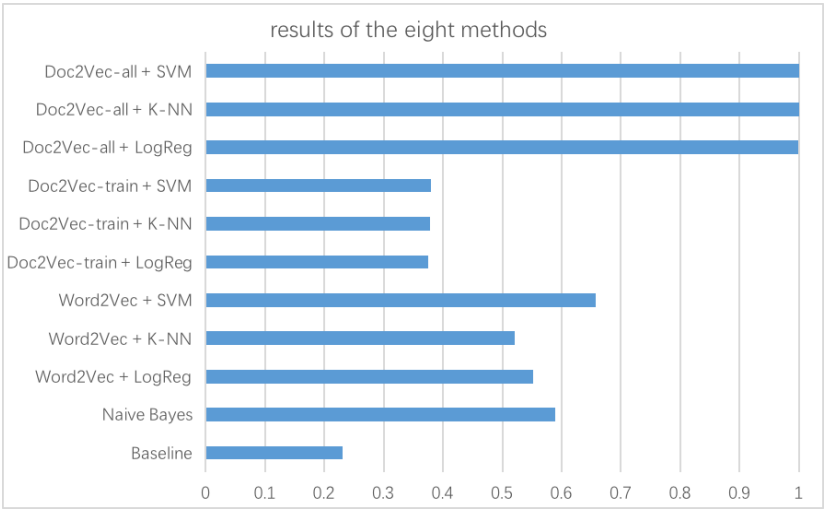

Figure 6. Accuracies of the eight methods on Yelp restaurant reviews sentiment analysis

6. Conclusion

In this project, we implemented 4 different methods - Naive Bayes, Logistic Regression, K-NN, and SVM as well as baseline for sentiment analysis on Yelp restaurant reviews. The last three methods are machine learning methods, which were based on vector models - Word2Vec or Doc2Vec. According to our results, Word2Vec method shows the best performance and SVM was the best classifier. Besides, 4-gram is the better language model than unigram, bigram, and trigram for the sentiment analysis. Although Doc2Vec is not very practical for sentiment prediction, we can use it to do the document clustering.

References

[1] N. Asghar, "Yelp Dataset Challenge: Review Rating Prediction," CoRR, abs/1605.05362, 2016.

[2] M. D. Devika, C. Sunitha and A. Ganesh, "Sentiment analysis: A comparative study on different approaches", *Procedia Computer Science,* vol. 87, pp. 44–49, 2016.

[3] N. Engonopoulos, A. Lazaridou, G. Paliouras and K. Chandrinos, "ELS: a word-level method for entity-level sentiment analysis," in *WIMS'11 Proceedings of the International Conference on Web Intelligence,Mining and Semantics,* 2011.

[4] N. Farra, E. Challita, R. Assi and H. Hajj, "Sentence-level and document-level sentiment mining for arabic texts," in *Proceeding IEEE International Conference on Data Mining Workshops,* 2010.

[5] G. Ganu, N. Elhadad and A. Marian, "Beyond the Stars: Improving Rating Predictions using Review Text Content," in *Proceedings of the 12th International Workshop on the Web and Databases,* pp. 1–6, 2009.

[6] Y. Goldberg and O. Levy, "word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method." *arXiv preprint arXiv:1402.3722,* 2014.

[7] D.-M.E.D.M. Hussein, "A survey on sentiment analysis challenges," *Journal of King Saud University – Engineering Sciences,* http://dx.doi.org/10.1016/j.jksues.2016.04.002, 2016.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," In *International Conference on Learning Representations: Workshops Track,* arxiv.org/abs/1301.3781, 2013.

[9] B. Thomas, "What consumers think about brands on social media, and what businesses need to do about it," Report, Keep Social Honest, 2013.

[10] G. Vinodhini and RM. Chandrasekaran, "Sentiment analysis and opinion mining: a survey," *International Journal of Advanced Research in Computer Science and Software Engineering,* Vol. 2, no. 6, pp. 282-292, 2012.

[11] A. Yessenalina, Y. Yue and C. Cardie, "Multi-level structured models for document-level sentiment classification." in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing,* MIT, Massachusetts, Association for Computational Linguistics, USA, pp. 1046–1056, 2010.

[12] H. C. Zhou and F. Song, "Aspect-level sentiment analysis based on a generalized probabilistic topic and syntax model," in *Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference,* Association for the Advancement of Artificial Intelligence, 2015.

[13] Sentiment Analysis using Doc2Vec [Internet]. 2017. Available from: https://github.com/linanqiu/word2vec-sentiments/blob/master/word2vec-sentiment.ipynb

[14] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In International Conference on Machine Learning, 2014.