Data Mining 2018

Assignment 2

Predicting Deceptive Opinion Spam with Text Mining Algorithms

Authors:

Olaf Kampers (4255194) Andreas Sakapetis (6315305)

Course coordinator: Ad Feelders

Abstract

The goal of this paper was to keep track of different classifiers and their performance in text mining. More specifically, four different machine learning algorithms are used to identify deceptive opinion spam: Naive Bayes, Regularized Logistic Regression, Classification Trees and Random Forests. The experiment design is based on the cross-industry standard process for data mining ([2] Chapman et al., 2000). The results obtained in this research suggest that Naive Bayes performs slightly better than the other classifiers. Including bigram features did not improve the classification results.

Keywords: *Machine Learning, Classifiers, Naive Bayes, Regularized Logistic Regression, Classification trees, Random forests*

1 Introduction

This report describes the effort to discriminate between truthful and deceptive reviews using classifiers for text mining, carried out during the Data Mining 2018 course at Utrecht University. The following four classification algorithms are used for identifying these deceptive reviews:

- Naive Bayes (generative linear classifier)
- Regularized Logistic Regression (discriminative linear classifier)
- Classification Trees (flexible classifier)
- Random Forests (flexible classifier)

In the section *Motivation* the incentives for the attempt to identify deceptive reviews are explained. Secondly, the section *Data Description* elaborates on the dataset provided for this report. Thirdly, the section *Experiment Approach* summarizes the approach that was selected and contains information about the different parameters and other details that were used to produce the results. The section *Results* is a presentation of the results from the experiments and their analysis. Finally, the section *Conclusions* contains conclusions based on the results.

1.1 Motivation

During the Data Mining 2018 course, taking place at the University of Utrecht, theory about text classification and several relevant classification algorithms was discussed. However, in order to fully comprehend the various techniques, it is necessary to practically implement and test them, so as to gain hands-on experience. A variety of examples were introduced to the students, however the application of the various classifiers in text mining will bring more insight to the way that those classifiers function.

Reading [5] Ott et al. (2011) as well as [6] Ott et al. (2013), the efforts to identify deceptive opinion spam by the authors, revealed an interesting challenge to the students.

Deceptive opinion spam makes it incredibly hard for customers to get a truthful impression about a hotel or restaurant, so it is a very relevant problem in this digital society. As already mentioned, this report aims to discriminate between truthful and deceptive reviews using text mining classifiers. The authors of this report will primarily focus on identifying deceptive reviews on a text corpus that consists of truthful and deceptive hotel reviews of 20 hotels in Chicago ([5] Ott et al., 2011) and subsequently will identify the top 5 words indicative to either a truthful review or a deceptive review.

2 Data Description

The dataset which is used by the classifiers for identifying deceptive reviews, consists of genuine and fake reviews for 20 hotels in Chicago. The genuine reviews have been collected from several popular online review communities, while the fake reviews have been obtained from Mechanical Turk. Details about the collection of these reviews can be found in the papers of Ott et al. [4,5].

The data set contains 800 negative reviews in total; 400 of these reviews are truthful and the other 400 reviews are deceptive. Each review is stored in a separate txt-file. Truthful and deceptive reviews are differentiated by the first letter of their filename (either 'd' or 't').

The training set contains 640 reviews, which are used for training the classifiers and for the tuning of hyperparameters. The other 160 reviews are then used for testing the performance of the classifiers.

2.0.1 Data Exploration

The following figures show the most common unigrams and bigrams (sequence of two adjacent words) in the data set. Intuitively, these frequent words are probably of less importance when distinguishing between the truthful and deceptive reviews.

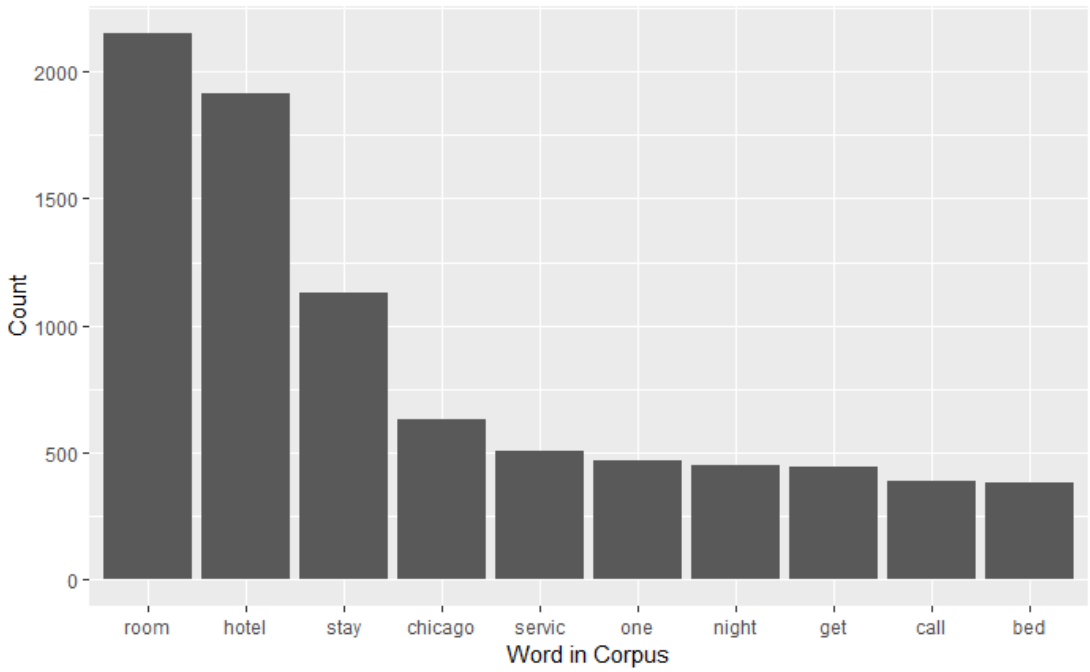

Figure 1: Most Frequent Words

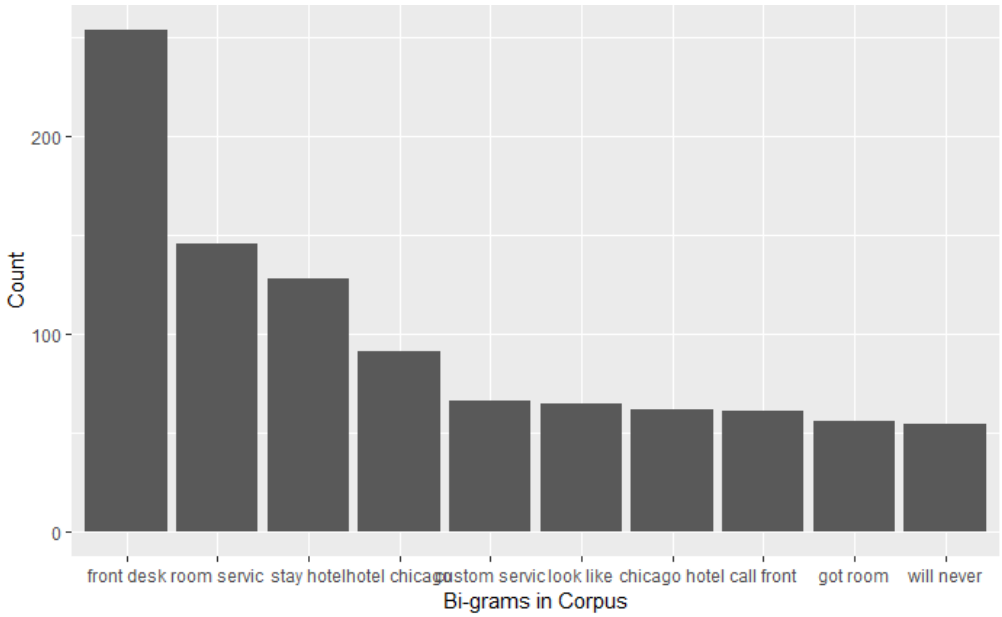

Figure 2: Most Frequent Bigrams

3 Experiment Approach

3.1 Research Understanding

The objective of the current study is to establish which classifier performs better in identifying deceptive reviews from a dataset of negative reviews and to identify indicative words that point either to a truthful review or to a deceptive review.

For this research five folds were acquired as described in the *subsection Data Understanding*. Folds 1-4 (640 reviews) are used as a training set and for hyperparameter tuning. Fold 5 (160 reviews) is used to estimate the performance of the classifiers that were selected on the training set, in other words it is selected as a test set. Also, the evaluation of the experiment was conducted in two parts: The first part including only unigram features and the second part including both unigram and bigram features.

3.2 Data Understanding

The data were accessible through the website [1] of the author [7] Ott et al. (2012), were a more in-depth insight can be acquired [2]. The dataset as already described in *Section* 2

[1] http://myleott.com
[2] http://myleott.com/op-spam.html

contains 800 negative reviews in total; 400 of these reviews are truthful and the other 400 reviews are deceptive. The amount of reviews was considered large enough to proceed with the study and compare the performance of the different classifiers.

3.3 Data Preparation

For this initial exploration of the data the authors used the "tm" text mining package([4] Meyer et al., 2008). A proportion of time was dedicated into loading the data to "Rstudio", professional software for the R statistical computing environment[3]. The authors created a text corpus which can easily be digitally processed and which is the main data structure used in this research. However some pre-processing was required before the data was viable for classification. The following cleaning methods were used:

- Removal of punctuation:
  The use of punctuation is inconsistent in online reviews and complicates the classification of words.

```
corpus <- tm_map(corpus, removePunctuation)
```

- Removal of capital letters:
  The use of capital letters is also inconsistent and unnecessarily increases the number of predictors.

```
corpus <- tm_map(corpus, content_transformer(tolower))
```

- Removal of English stop words:
  Stop words are regularly used in text but they do not say anything about the meaning of the text on their own.

```
corpus <- tm_map(corpus, removeWords, stopwords("english"))
```

- Removal of numbers:
  Numbers are irrelevant for classification.

```
corpus <- tm_map(corpus, removeNumbers)
```

- Removal of excess white space

```
corpus <- tm_map(corpus, stripWhitespace)
```

- Stemming:
  Makes text more comparable across observations and groups words together with similar meaning.

```
corpus <- tm_map(corpus, stemDocument)
```

[3] https://www.rstudio.com/

3.4 Modeling

This section contains information about the ways in which the different classifiers were configured for this research.

3.4.1 Naive Bayes

For the Naive Bayes classifier, the authors used the code provided by the coordinator of the Data Mining course. Some experiments were done regarding feature selection, however doing so did not result in significant improvement. The sparse terms with a percentage of 95% were removed for unigrams to greatly reduce the number of features. For the set containing both the unigrams and the bigrams, the sparse terms with a percentage of 99% were removed.

3.4.2 Logistic Regression

The package "gmlnet" ([3] Friedman et al., 2009) was used for this algorithm. Before the regularized logistic regression, cross validation was used to determine the right lambda-value. The results from the cross validation for both unigrams and bigrams can be observed in figures 3 and 4 below. The 'lambda.1se' value was chosen to make sure that the error is within one standard error of the minimum error. The results obtained by using logistic regression with lasso penalty can be observed in the Results section.

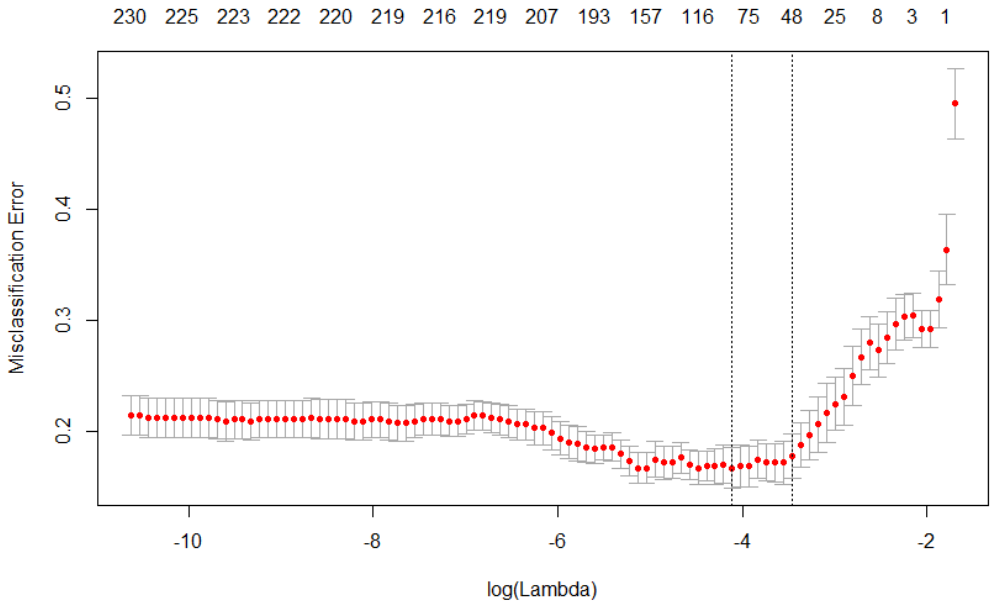

Figure 3: Cross-Validation on lambda for unigrams

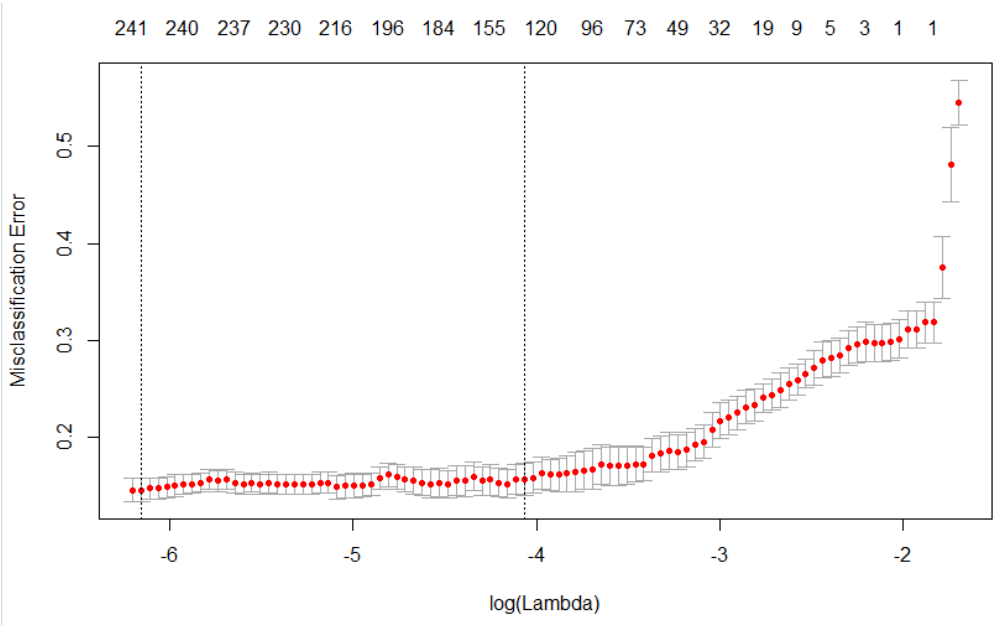

Figure 4: Cross-Validation on lambda for Bigrams

3.4.3 Classification Trees

For the classification tree, the "rpart" library was used for creating decision trees([8] Therneau, Atkinson, et al., 1997). The specific library ("rpart") utilizes the CART (classification and regression trees) algorithm. The idea is to use the classification algorithm and to permit the decision tree to grow fully, thus the cost-complexity pruning parameter was set to zero cp value 0. Next, the decision was to post-prune the classification tree for unigrams and bigrams with a cp value of 0.025.

```
> reviews.rpart$cptable
          CP nsplit rel error    xerror       xstd
1 0.41562500      0  1.000000 1.093750 0.03935438
2 0.02500000      1  0.584375 0.584375 0.03595257
3 0.01562500      2  0.559375 0.606250 0.03633526
4 0.01458333      3  0.543750 0.587500 0.03600876
5 0.00937500     11  0.390625 0.603125 0.03628210
6 0.00750000     13  0.371875 0.603125 0.03628210
7 0.00625000     18  0.334375 0.606250 0.03633526
8 0.00312500     22  0.309375 0.628125 0.03669360
9 0.00000000     24  0.303125 0.656250 0.03711965
```

Figure 5: Cp value for Classification Tree with unigrams

```
> reviews.tree2$cptable
           CP nsplit rel error    xerror       xstd
1  0.415625000      0  1.000000 1.106250 0.03930472
2  0.025000000      1  0.584375 0.584375 0.03595257
3  0.015625000      2  0.559375 0.600000 0.03622844
4  0.014583333      4  0.528125 0.593750 0.03611961
5  0.009375000     12  0.375000 0.600000 0.03622844
6  0.007812500     14  0.356250 0.596875 0.03617428
7  0.006250000     16  0.340625 0.593750 0.03611961
8  0.004166667     18  0.328125 0.609375 0.03638793
9  0.003125000     22  0.306250 0.603125 0.03628210
10 0.000000000     25  0.296875 0.628125 0.03669360
```

Figure 6: Cp value for Classification Tree with unigrams

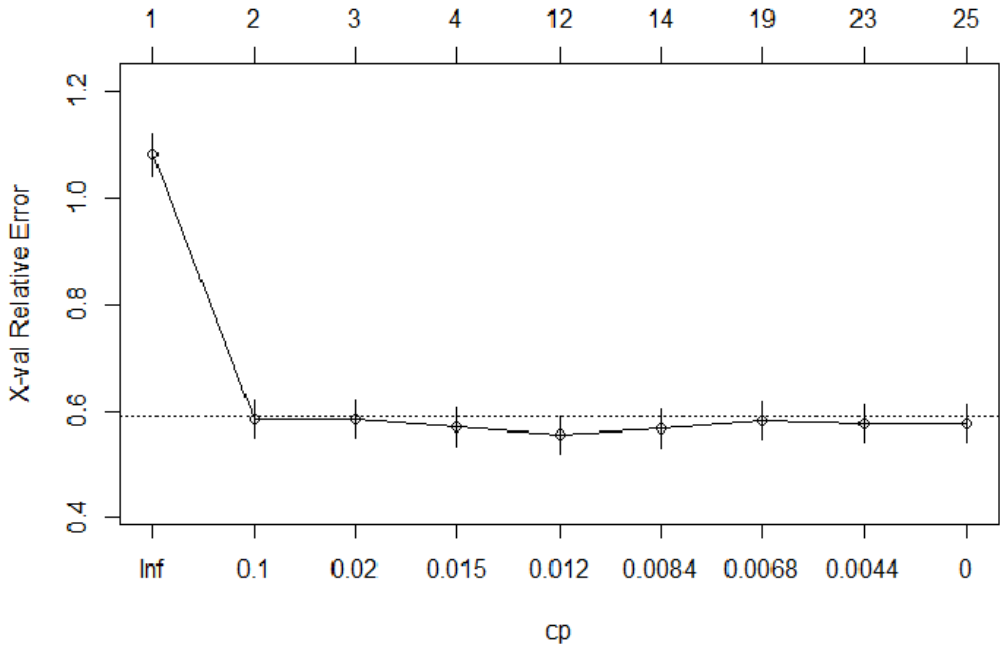

Figure 7: Cp value for Classification Tree with Bigrams

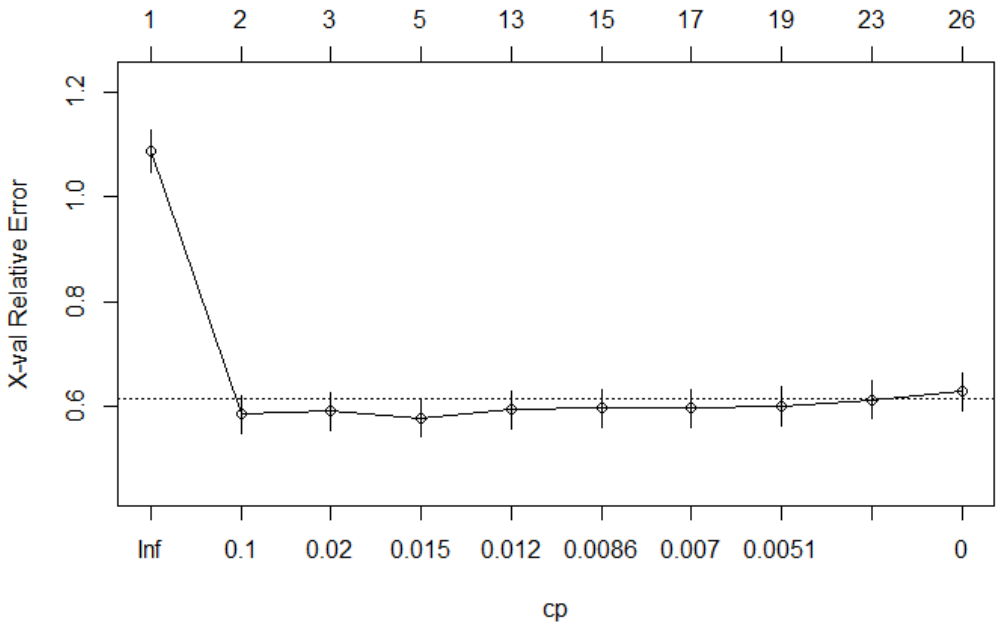

Figure 8: Cp value for the Classification Tree with Bigrams

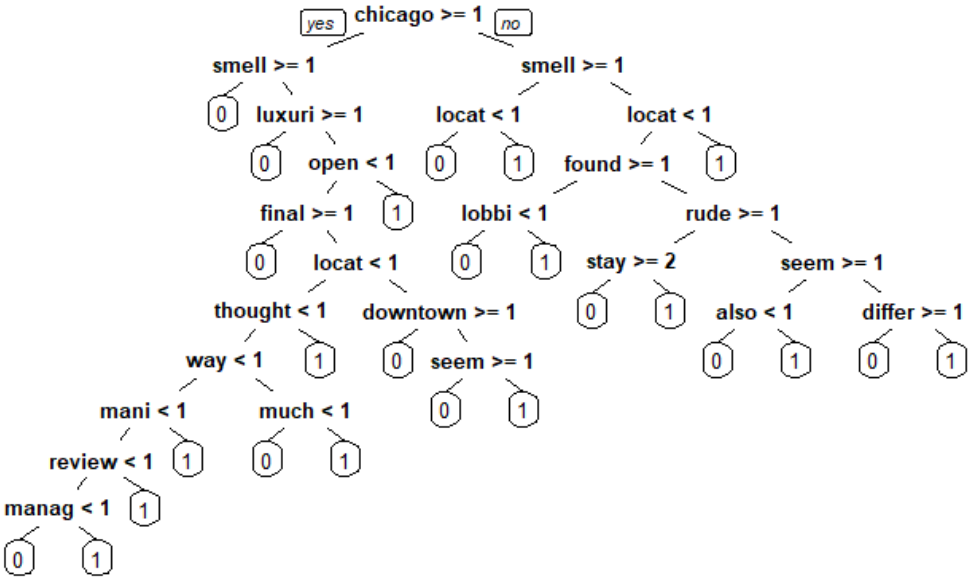

Figure 9: Classification Tree unigrams

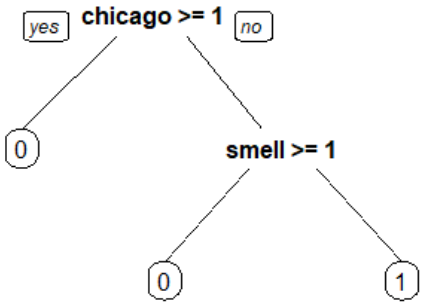

Figure 10: Pruned Version of the Classification Tree with unigrams

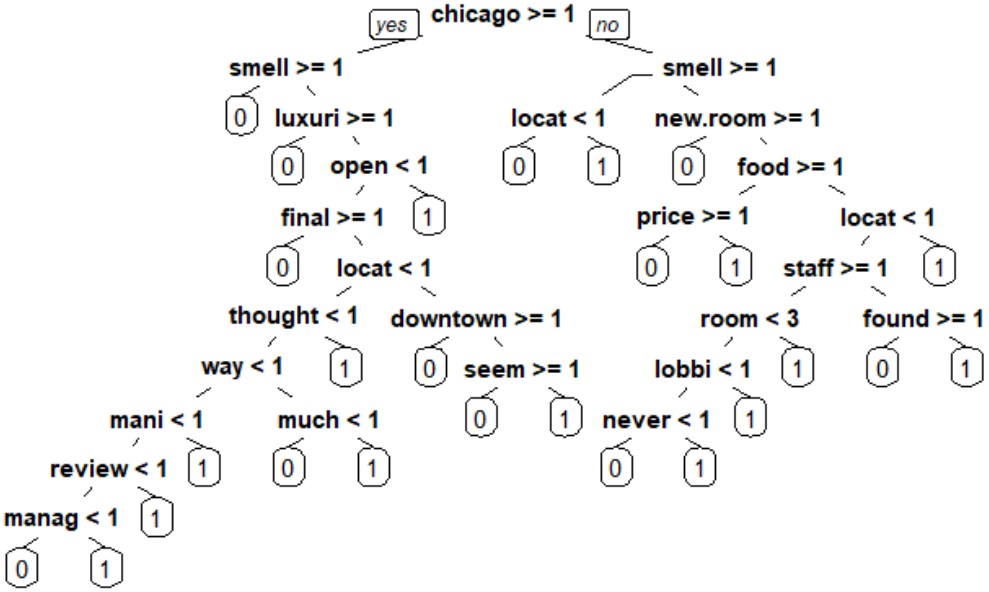

Figure 11: Classification Tree with Bigrams

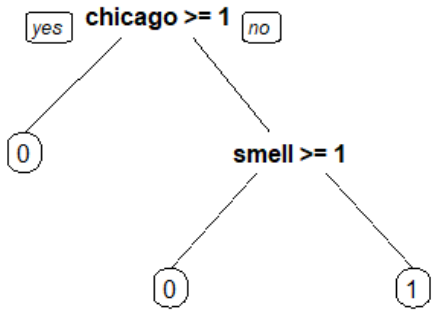

Figure 12: Pruned Version of the Classification Tree with Bigrams

3.4.4 Random Forest Classifier

A Random Forest classifier is a stronger tool than a standard classification tree, however in order to set the amount of trees to grow, it is necessary to observe the error produced while the number of trees is increasing. Thus, observing *Figure* 13 and *Figure* 14, the optimum amount of trees selected is 100. The number of features considered at each split are computed with $\sqrt{mtry}$ ([1]Altmann et al., 2010). The total number of features in the test set is 321, and the number of selected features for each split is 17.

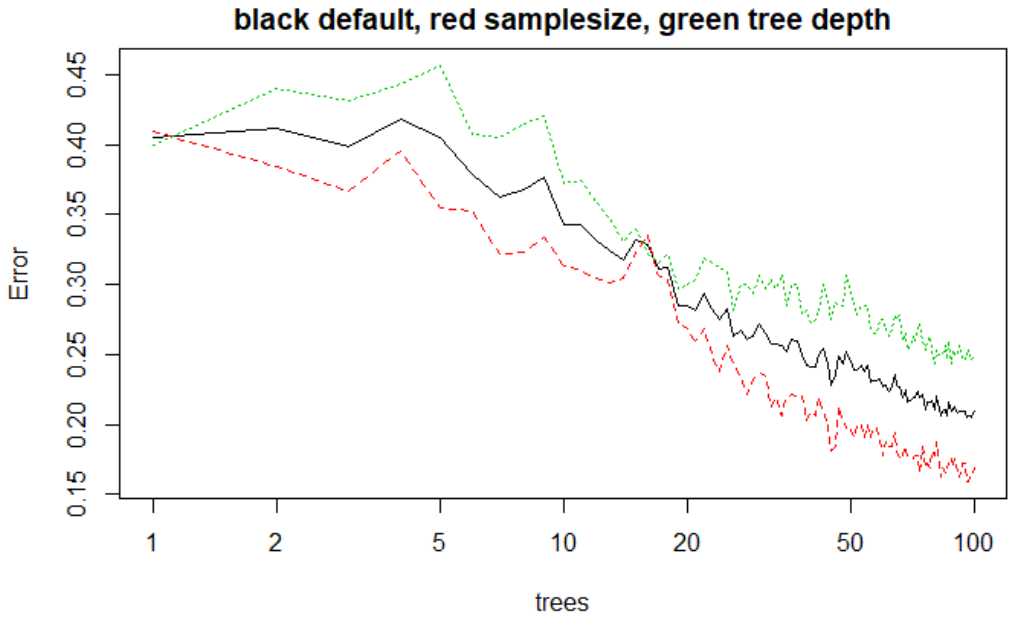

Figure 13: Random Forest Error with Bigrams

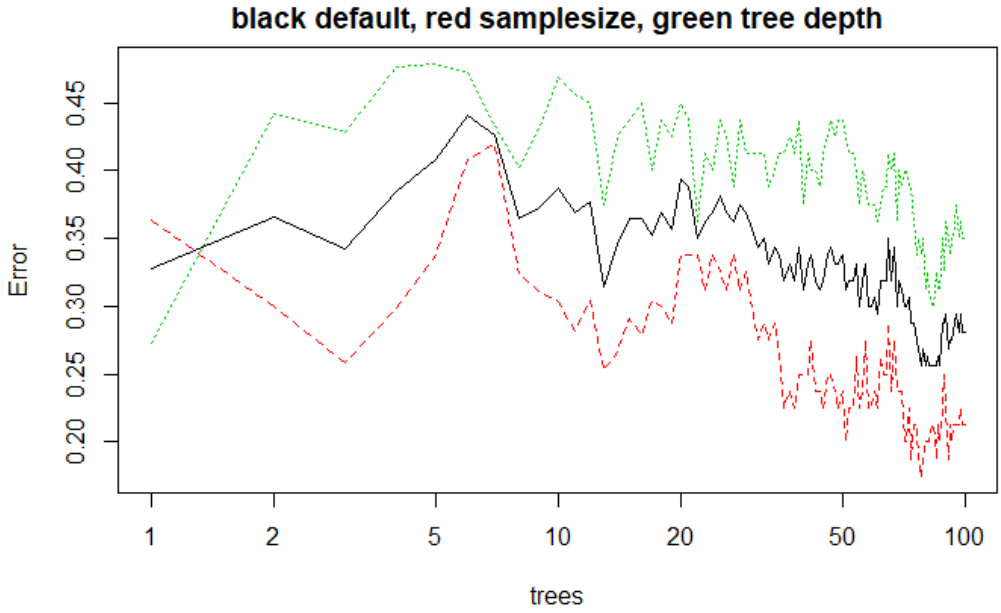

Figure 14: Random Forest Error with Bigrams

3.5 Evaluation

In this section the results of the experiment will be presented. The confusion matrices generated from the data calculated by the classification algorithms can be seen in the tables on the following pages. The vertical columns indicate the classes of the observations provided by the dataset, whereas the horizontal rows indicate the classes of the observations predicted by the classification algorithm . The bottom part of the table (below the grey row) contains three rows with the Precision, Recall, and Accuracy values based on the results in the top part of the table.

3.5.1 Naive Bayes

		True Class	
		Class 0	**Class 1**
Predicted Class	**Class 0**	60	11
	Class 1	20	69
Accuracy	(60+69) / (60+11+20+69)≈ 0.81		
Precision	60 / (60+11)≈ 0.84		
Recall	60 / (60+20)≈ 0.75		
F1 Score	(2 * Precision * Recall) / (Precision + Recall)≈ 0.79		

Table 1: Naive Bayes

		True Class	
		Class 0	**Class 1**
Predicted Class	**Class 0**	54	6
	Class 1	26	74
Accuracy	(54+74) / (54+6+26+74)≈ 0.80		
Precision	54 / (54+6)≈ 0.90		
Recall	54 / (54+26)≈ 0.68		
F1 Score	(2 * Precision * Recall) / (Precision + Recall)≈ 0.77		

Table 2: Naive Bayes with Bigrams

3.5.2 Logistic Regression

		True Class	
		Class 0	**Class1**
Predicted Class	**Class 0**	56	8
	Class 1	24	72
Accuracy	(56+72) / (56+8+24+72)≈ 0.8		
Precision	56 / (56+8)≈ 0.88		
Recall	56 / (56+24)= 0.7		
F1 Score	(2 * Precision * Recall) / (Precision + Recall)≈ 0.77		

Table 3: Logistic Regression

		True Class	
		Class 0	**Class1**
Predicted Class	**Class 0**	48	7
	Class 1	32	73
Accuracy	(48+73) / (48+7+32+73)≈ 0.75		
Precision	48 / (48+7)≈ 0.87		
Recall	48 / (48+32)= 0.6		
F1 Score	(2 * Precision * Recall) / (Precision + Recall)≈ 0.71		

Table 4: Logistic Regression with Bigrams

3.5.3 Classification Tree

		True Class	
		Class 0	**Class1**
Predicted Class	**Class 0**	56	30
	Class 1	24	50
Accuracy	(56+30) /(56+30+24+50)≈ 0.67		
Precision	56 / (56+30)≈ 0.65		
Recall	56 / (56+24)≈ 0.7		
F1 Score	(2 * Precision * Recall) / (Precision + Recall)≈ 0.67		

Table 5: Pruned classification Tree

		True Class	
		Class 0	**Class1**
Predicted Class	**Class 0**	42	15
	Class 1	38	65
Accuracy	(42+65) / (42+15+38+65)≈ 0.65		
Precision	42 / (42+15)≈ 0.73		
Recall	42 / (42+38)= 0.52		
F1 Score	(2 * Precision * Recall) / (Precision + Recall)≈ 0.61		

Table 6: Pruned classification Tree with Bigrams

3.5.4 Random Forest

		True Class	
		Class 0	**Class 1**
Predicted Class	**Class 0**	56	9
	Class 1	24	71
Accuracy	(56+71) / (56+9+24+71)≈ 0.79		
Precision	56 / (56+9)≈ 0.86		
Recall	56 / (56+24)= 0.7		
F1 Score	(2 * Precision * Recall) / (Precision + Recall)≈ 0.77		

Table 7: Random Forests Classification Algorithm

		True Class	
		Class 0	**Class1**
Predicted Class	**Class 0**	41	13
	Class 1	39	67
Accuracy	(41+67) / (41+13+37+67)≈ 0.67		
Precision	41 / (41+13)≈ 0.75		
Recall	41 / (41+37)≈ 0.51		
Score	(2 * Precision * Recall) / (Precision + Recall)≈ 0.61		

Table 8: Random Forest with bigrams

3.6 Summary

Table 9 shows an overview of all the results. While the other two tables in this subsection show the most important features for both unigrams and bigrams, which were identified during the cross validation.

3.6.1 Overview of all the results

Classifier	Features	Accuracy	Precision	Recall	F1 Score
Naïve Bayes	Unigrams	0.81	0.84	0.75	0.79
	Bigrams	0.80	0.90	0.68	0.77
Logistic Regression	Unigrams	0.80	0.88	0.7	0.77
	Bigrams	0.75	0.87	0.60	0.71
Classification Trees	Unigrams	0.67	0.65	0.7	0.67
	Bigrams	0.65	0.73	0.52	0.61
Random Forest	Unigrams	0.79	0.86	0.7	0.77
	Bigrams	0.67	0.75	0.51	0.61

Table 9: Summarized Results

Unigrams			
Truthful		**Deceptive**	
Feature	**Coefficient**	**Feature**	**Coefficient**
elev	0,6133	recent	-0,9060
star	0,6082	millenium	-0,8405
coffe	0,4483	smell	-0,8220
cant	0,4401	luxuri	-0,7706
concierg	0,3882	chicago	-0,6955

Table 10: Most Important Features for Unigrams

Bigrams			
Truthful		**Deceptive**	
Feature	**Coefficient**	**Feature**	**Coefficient**
book hotel	0,9322	homewood suit	-1,1180
mani	0,8873	millenium	-0,9083
elev	0,6737	luxuri	-0,8939
four season	0,6662	smell	-0,8612
star	0,6446	arriv room	-0,8449

Table 11: Most Important Features for Bigrams

4 Conclusions

When the performance of the generative linear model (Naive Bayes) and the discriminative linear model (Logistic Regression) are compared, one can see that the performance is very similar. In this case, Naive Bayes seems to be slightly better in comparison to Logistic Regression.

The Random Forest classifier was not able to perform better than both the Logistic Regression and the Naive Bayes classifier, but for unigrams the difference in performance between these algorithms is not very large. However the Random Forest classifier performed significantly worse when the bigrams were included. For a more thorough overview, results are depicted in table 9 where Accuracy, Precision, Recall and F1 Score are acquired for all the classifiers.

In general, it seems that including bigrams did not improve the performance of the classifiers. For the Naive Bayes classifier, the results for unigrams and bigrams were very similar. The precision for bigrams was slightly better, but the recall was worse. For Logistic Regression, including bigrams did not improve any of the performance measures at all. Concerning the Classification Tree, the performance was also not improved when including bigrams. As a matter of fact only the recall was slightly better. Finally, the Random Forest classifier performed particularly worse when including the bigrams. These results could be explained by the relatively small size of the training data. Research in the papers of Ott et al. [4,5] suggests that including bigrams could have a positive effect on the results of text classifiers for larger datasets, but the difference is not very large.

Lastly, all classifiers seem to be fairly good at distinguishing between the truthful and deceptive reviews, with an accuracy between 0.65 and 0.81, good precision and decent recall. The accuracy was not expected to be perfect since words can have a different meaning depending on the context. For further research, it might be a good idea to also examine longer n-grams of text because of this.

References

[2] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). Crisp-dm 1.0 step-by-step data mining guide.

[5] Ott, M., Choi, Y., Cardie, C., & Hancock, J. T. Finding deceptive opinion spam by any stretch of the imagination. In: *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*. Association for Computational Linguistics. 2011, 309–319.

[6] Ott, M., Cardie, C., & Hancock, J. T. Negative deceptive opinion spam. In: *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*. 2013, 497–501.

[7] Ott, M., Cardie, C., & Hancock, J. T. Estimating the prevalence of deception in online review communities. In: *Proceedings of the 21st international conference on world wide web*. ACM. 2012, 201–210.

[4] Meyer, D., Hornik, K., & Feinerer, I. (2008). Text mining infrastructure in r. *Journal of statistical software*, *25*(5), 1–54.

[3] Friedman, J., Hastie, T., & Tibshirani, R. (2009). Glmnet: Lasso and elastic-net regularized generalized linear models. *R package version*, *1*(4).

[8] Therneau, T. M., Atkinson, E. J. et al. (1997). An introduction to recursive partitioning using the rpart routines.

[1]Altmann, A., Toloşi, L., Sander, O., & Lengauer, T. (2010). Permutation importance: A corrected feature importance measure. *Bioinformatics*, *26*(10), 1340–1347.