Utrecht University

Data Mining: Assignment 2

TEXT CLASSIFICATION FOR THE DETECTION OF OPINION SPAM

Ana Borovac
6584446

Argyro (Iro) Sfoungari
6528015

October, 2018

Contents

Abstract

Our task was to compare different classifiers (Naive Bayes, Logistic regression, Classification tree, Random forests), which predict if a hotel review is deceptive or truthful. We trained our models on 640 negative reviews of hotels in Chicago area; half of them were deceptive and half of them were truthful reviews. Furthermore, we analysed the effect of hyperparameters, e. g. coplexity parameter for pruning the Classification tree. We concluded....

1 Introduction

Nowadays email filters are able to classify spam and not spam emails quite successfully. Imagine now that you have a web site with hotel reviews and your goal is to offer the most truthful reviews but you can not check every single review and it is also hard to recognise when the review is not truthful. Therefore you would like to have a mechanism which is going to help you to achieve your goal.

In this assignment we tried to solve the above problem with 4 methods; Naive Bayes, logistic regression, classification tree and random forests. Before we started, we analysed the work that has been done already (section 2). Next, we did some data preprocessing, it is described in the section 3. Used methods are explained in the section 4 and analysis of the results (section 5) is in the section 6.

2 Related work

The authors of "Finding Deceptive Opinion Spam by Any Stretch of the Imagination" [8] compared truthful and deceptive positive reviews for hotels. They used Naive Bayes and Support Vectors Machine classifiers. For comparison they also had 3 human untrained judges which tried to predict if a review was real or fake. The results had shown that automated classifiers outperform human judges in almost every metric (precision, recall, F-score). They explain that with that untrained humans often focus on unreliable cues to deception. One of the results was also that models trained only on unigrams outperformed all non–text-categorizations approaches (genre identification and psycholingustic deception detection). Furthermore, the results were even better when bigrams were used.

In the article "Negative Deceptive Opinion Spam" [7] the authors created corpus of gold standard 400 reviews on 20 Chicago hotels and then used them to compare n-gram–based Support Vector Machine classifiers with untrained human judges. They concluded that the best detection performance was achived through automated classifiers.

3 Data

Our data consisted of 400 negative deceptive and 400 negative truthful hotel reviews that have been collected by Myble Ott and others ([7], [8]).

3.1 Data preprocessing

Before modeling we did some data preprocessing in order to get better models for predicting if a review is fake or real. First step in data preprocessing was

to remove punctuation marks, after that we made every letter lower case, we also removed stopwords, numbers and excess whitespace.

Next, we created a test and a training set. Because we wanted to use cross validation, we divided our data into 5 folds (each of size 160 samples - 80 fake reviews and 80 truthful reviews). So, 4 of the folds represented a training set, the remaining 5th fold was a test set.

At that moment every unique word from the training reviews was a feature. A number of features was large therefore we removed the words as features which occur less than 1 % of training documents.

After that we created a new training set which consisted all the features from the previous training set and bigrams. We again removed bigrams that occur in lass that 3 % of training documents. At the end we had two training sets ready to be used.

How we chose the percentage of the features which were used is more detailed explained in the subsection 5.1.

4 Methods

We used different classifiers to model our task:

- Naive Bayes (generative linear classifier); Naive Bayes is a probabilistic classifier. Class is predicted from features which has highest probability with independence assumption (features are independent within each class) [2]:

$$\hat{c} = \arg\max_{c \in C} P(c) \prod_{i=1}^{m} P(x_i|c)$$

- Logistic regression (discriminative linear classifier); Logistic regression is one of discriminative classification methods [4]. That means that modelling of probability, how likely are we to predict a class c with given input, is direct. In a binary case we predict class 1 if it holds:

$$\frac{P(Y=1|x)}{P(Y=0|x)} > 1; \quad P(Y=1|x) = \frac{1}{1+e^{-\beta^T x}}$$

  otherwise we predict class 0.

- Classification tree (flexible classifier); Classification trees are usually not the best models [3], but are easy to interpret and can handle both numerical and categorical attributes.

- Random forests (flexible classifier); Random forests are improved classification trees, where the best split is chosen among k random features and not among all of them [6].

5 Results

After the preprocessing procedure, we started analysing the data based on the aforementioned classifiers. Initially, we will present you the data analysis step by step, as we worked during the assignment.

5.1 Naive Bayes

Initially we started with the Naive Bayes classifier trained with the code from the lectures [1]. We considered that it would be useful to create several combinations of unigrams and bigrams by changing the appearance rate of the terms, in order to receive the highest accuracy percentage. Although in the beginning we considered that the highest percentage was achieved by removing the terms that occur less than 1 % for unigrams and less than 35 % for bigrams (table 1), later we realised that this combination did not involve any bigrams at all. So we had to continue our research by choosing a different combination of percentages, in order to include results for bigrams as well and draw to specific conclusions. The percentage we finally chose was 99 % for unigrams and 97 % for bigrams.

Terms presence \ Accuracy		Unigrams	Bigrams
99 %	99 %	0,87	0,53
	97 %	0,85	0,49
	80 %	0,84	0,63
	76 %	0,87	0,77
	70 %	0,87	0,87
	68 %	0,86	0,86
	65 %	**0,87**	**0,87**
95 %	65 %	0,84	0,84
80 %	70 %	0,67	0,67
70 %	80 %	0,60	0.5
	70 %	0,60	0,60

Table 1: First two columns represent terms presence; in how many training documents did a specific unigram or bigram appeared. Last two columns show accuracies of Naive Bayes classifiers trained on the selected features.

For the other classifiers, we tried to compare the accuracy not only using their pure forms but also changing the hyper parametric values to compare the differences for unigrams and bigrams.

5.2 Classification tree

For Classification trees we followed the following process: we used the `rpart` library of R [9] in order to grow a tree and then the function `prune`, in order

to prune it. Initially we used the complexity parameter equals to 0,037 but afterwards we used the `plotcp` function in order to find the optimal value for the complexity parameter. The function uses cross-validation after which the results are represented in a plot.

Following the graph (figure 1), we tested different values for the complexity parameters, some of them are presented in the table 2.

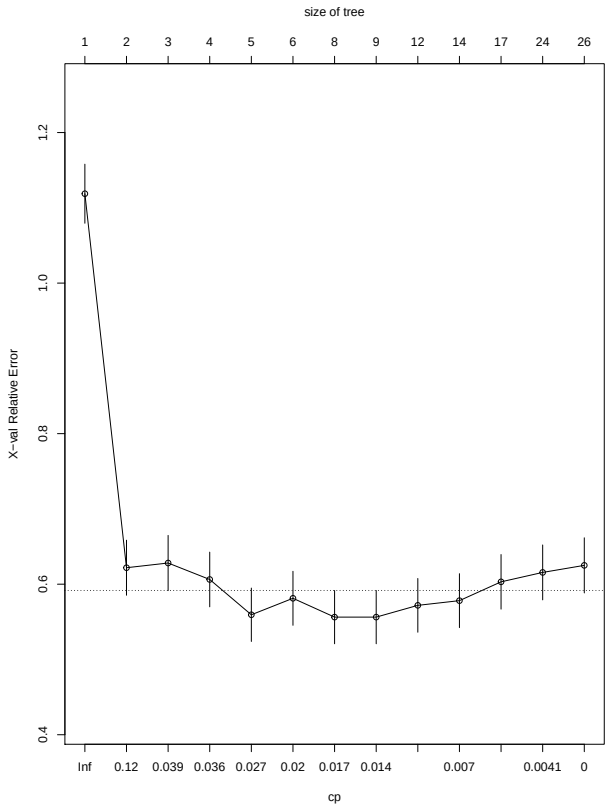

Figure 1: Plot returned from a function `plotcp`, which gives a visual representation of the cross-validation results in a Classification tree.

Complexity parameter \ Accuracy	Unigrams	Bigrams
0,12	**0,70**	**0,70**
0,037	0,68	0,68
0,027	0,69	0,69
0,007	0,68	0,67
0,00041	0,68	0,68

Table 2: Accuracies from Classification trees pruned with the complexity parameter from first column.

As it can be seen clearly from the table, the optimal value for the complexity parameter is 0,12 as it gives the highest accuracy. At this point we noted that for both cases the accuracy remains the same. So we concluded that bigrams do not contribute to classification trees.

5.3 Random Forests

For random forests we worked as follows. We used the function `randomForest` (from R libraby verb|randomForest| [6]) and we implemented various tests, changing the number of trees (`ntree`) and the number of variables randomly sampled as candidates at each split (`mtry`).

Number of trees, mtry \ Accuracy		Unigrams	Bigrams
100	4	0,75	0,42
200	4	0,77	0,39
	6	**0,82**	**0,44**
	8	0,81	0,45
300	4	0,79	0,43
	6	0,80	0,45
	8	0,80	0,44

Table 3: First column presents the number of trees in a Random forest, second column is the numbers of features selected in each step of the algorithm and the last two columns represent calculated accuracies.

From the table 3 we came to the conclusion that the best combination is achieved by 200 trees and mtry equals to 6, while accuracy levels for bigrams are very low.

Lambda \ Accuracy	Unigrams	Bigrams
lambda.1se	0,80	0,78
lambda.min	**0,81**	**0,82**

Table 4: Accuracies for two different Logistic regression models; one with largest value of lambda which is 1 standard error apart from minimum, and the minimum lambda.

5.4 Logistic regression

For logistic regression model we used `cv.glmnet` function from `glmnet` library [5]. The function does k-fold cross-validation and as a result returns a value for `lambda`. We left the default value of k, which is set to 10. Since we know that our model is binomial (`family = ‘‘binomial’’`), we were able to set `type.measure` to "class". This means that the loss which is used for cross-validation is misclassification error. We tried to make some tests changing

the value of the user-supplied lambda sequence (table 4).

Accuracy, Precision, Recall and F1_score with (99 % of unigrams, 97 % of bigrams, number of trees = 200, `mtry` = 6, lambda.1se and `cp` = 0,12) for 8 models are represented in the table 5.

		A	P	R	F1
Naive Bayes	Unigrams	0,81	0,82	0,80	0,80
	Bigrams	0,47	0,40	0,71	0,52
Classification Trees	Unigrams	0,71	0,61	0,65	0,63
	Bigrams	0,71	0,61	0,65	0,63
Random Forests	Unigrams	0,81	0,85	0,74	0,77
	Bigrams	0,47	0,70	0,93	0,49
Logistic Regression	Unigrams	0,79	0,67	0,81	0,73
	Bigrams	0,79	0,66	0,84	0,74

Table 5: Accuracies (A), Precisions (P), Recalls (R) and F1_scores (F1) for 8 models.

6 Analysis

After studying the results in the section 5, we have concluded to some specific assumption which we will present you below. Initially, it would be beneficial to compare the generative linear model (Naive Bayes) with the discriminative linear model (Logistic regression). As it is clear from the tables, Naive Bayes gives a slightly higher accuracy when it comes to unigrams, therefore we can talk about a negligible difference. However, when we include bigrams in our analysis, the discriminative linear model seems to be much more efficient and accurate than Naive Bayes as the difference is enormous. Then we compared the discriminative linear model with Random forests. Once again the precision percentage is almost the same for the two classifiers, or it may become a bit higher for Random forests if we change the hyperparameters properly (number of trees, randomly selected features) for unigrams. Concerning the bigrams Logistic regression is repeatedly the best choice, as it gives by far the highest accuracy, while on the contrary Random forests give the lowest. As it mentioned earlier (Naive Bayes classifier case), the highest accuracy was achieved without the existence of bigrams, so we could say that bigrams do not contribute in performance improvement.

Regarding the most important terms towards truthful or deceptive reviews, in order to receive some accurate results we used the `importance` function, this function extracts the important measures produced by `randomForest`.

We ran the code twice. Once setting the `class` parameter equals to 0 (deceptive) and another one setting `class` equals 1 (truthful). Consequently

we receive as the most important terms for fake reviews the following:

1. Chicago,
2. location,
3. seemed,
4. hotel.chicago,
5. luxury.

Concerning the genuine reviews the most important terms presented in priority order are:

1. Chicago,
2. location,
3. luxury,
4. rude,
5. seemed.

References

[1] Ad Feelders. Code for naive bayes. `www.cs.uu.nl/docs/vakken/mdm/mnb.txt`. Online; accessed 24 October 2018.

[2] Ad Feelders. Text classification. `www.cs.uu.nl/docs/vakken/mdm/Slides/dm-text-naivebayes.pdf`, October 2017. Online; accessed 30 October 2018.

[3] Ad Feelders. Classification trees (1). `www.cs.uu.nl/docs/vakken/mdm/Slides/dm-classtrees-1-2018.pdf`, 2018. Online; accessed 30 October 2018.

[4] Ad Feelders. Logistic regression. `www.cs.uu.nl/docs/vakken/mdm/Slides/dm-logreg2018.pdf`, 2018. Online; accessed 30 October 2018.

[5] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[6] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[7] Myle Ott, Claire Cardie, and Jeffrey T. Hancock. Negative deceptive opinion spam. In *in HLT-NAACL*, 2013.

[8] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. 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*, HLT '11, pages 309–319, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[9] Terry Therneau, Beth Atkinson, and Brian Ripley. *rpart: Recursive Partitioning and Regression Trees*, 2015. R package version 4.1-10.