Sentiment analysis using neural networks and clustering algorithms

Benjamin Søtang Steenberg Olsen
Digital Innovation and Management
IT-University Of Copenhagen
Copenhagen, Denmark
Email: bsto@itu.dk

Dennis Leszkowicz
Software Development - Design Track
IT-University Of Copenhagen
Copenhagen, Denmark
Email: dele@itu.dk

Eimantas Jazukevicius
Software Development - Design Track
IT-University Of Copenhagen
Copenhagen, Denmark
Email: eija@itu.dk

***Abstract*—The aim of the project is to determine the sentiment of a review, by extracting meaningful information from text. The project was done as a group assignment for the Data Mining course at IT University of Copenhagen. Sentimental analyses like this projects is a core part of natural language processing techniques that are used widely across different industries. To determining sentiment we experimented with different methods such as random forest, clustering and multilayer perceptrons. The dataset used is "Reviews and ratings of airports 2015" from *airlinequality.com* website.**

May 19, 2017

I. Introduction

The demand for sentiment analysis has increased significantly in recent years, as the rate of data creation is growing rapidly. Reports, reviews and news scrapping techniques usually contain useful information that can be used for different purposes such as business intelligence and customer analysis. A problem we usually face by doing manual sentiment analysis is that a large text is only partially useful. Because of datasets containing thousands lines of text it is impossible for any human being process. For this purpose different text mining and natural language processing methods are used. With these methods it becomes possible to filter relevant parts of the text, make predictions on attributes and cluster it by relevance. In our task we took a dataset "Reviews and ratings of airports 2015" from *airlinequality.com* website. With this dataset we tried to cluster the text to find meaningful information and based on attempt to predict if the person recommended the given airport or not.
The dataset contained some basic information about reviewer - origin country, date and purpose of the flight, review itself and rating on the different parts of the airport like "wifi" or "shopping". Main attributes used for this project was review text, "recommended" value and "overall rating".

In the II part we briefly discuss the methods used in the project. In part III we explain our implementation - clustering, prediction using random forest and multilayer perceptron. In part IV the results are discussed and ideas for future work are suggested.

II. Methods

In this section, we introduce the methods that are later implemented in our project and their purpose.

A. Bag of words

The Bag of words method is mostly used for information retrieval (IR) and natural language processing (NLP). In this model a text is divided to separate words and the occurrences are counted which is then stored to a vector of numbers. The vectors can be used for feature extraction and purposes like email spam filtering [6]. However this model is not ideal for long text analysis. Words like "a", "the" are usually most frequent in any text and they doesn't provide any valuable information. For this purpose few preprocessing techniques are used - term frequency-inverse document frequency (TF-IDF) method or stop words removal using natural language processing libraries [4]. In our project bag of words will be used as first step in text processing. Vectors from bag of words output will be used for clustering and as input nodes in multilayer perceptron, random forest and clustering algorithms.

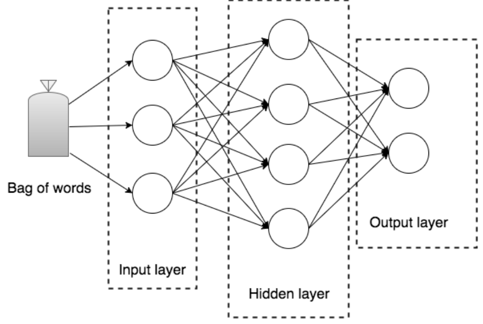
B. K-means

The K-means clustering algorithm is used to cluster the reviews based on its attributes - a combination of mixed data types counting strings and numerical values. The algorithms purpose is to partition n observations into k clusters. Given the input K, consisting of a set of observations - each observation fits into the cluster with the nearest mean. K-means starts by placing centroids $C_1 ... C_k$ at random locations. The algorithm repeats the following process until convergence:
For each point X_i it finds the nearest centroid C_j and assigns the point X_i to cluster j. For distance measure the Euclidean distance formula is normally used. However, because we are working with text analysis we have used the cosine distance. The function will be explained and discussed in the implementation section. For each cluster j = 1...K: a new centroid C_j mean of all points x_i is calculated and assigned to cluster j in previous step[1]. The algorithm stops when none of the clusters change. At this point it has converged.

C. Multilayer Perceptron

In this assignment we use the most popular form of a neural networks neural [1] a network which is trained with back propagation. Since all of the attributes for each text is going to be numeric using the bag of words method a neural network algorithm would be more appropriate than for example decision trees [1]. The algorithm assumes a network of nodes with a input layer, a number of hidden layers and a output layer. It is trained by first forward propagating the input, when it reaches the output layer, a prediction is made and the error of that predictions is then back-propagated updating weights of the edges between the nodes/neurons. This process is repeated for a number of iterations typically until prediction accuracy on a validation set stabilizes or start to decrease indicating over-fitting of training tuples. In Fig. 1 multilayer perceptron with bag of words input used in our project is shown.

Fig. 1. Multilayer Perceptron with bag of words input)



III. Implementations and Results

In this section we discuss our implementations of the individual algorithms and the results they produce.

A. Clustering text using K-means

With K-means we can't just use the bag-of-words model out of the box. Before we run the actual algorithm some transformations of the data is needed. We explain these preprocessing steps, full implementation and the results.

1) Data preprocessing: Before we even start to count words in each document some preprocessing decisions have to be made. First a few simple steps have been taken, At beginning all words are turned into lowercase and all numeric values and special characters are removed. These actions ensure that words that are actually equal does not seem different to the algorithm.

However, this is often not it enough, words such as "Airplane" and "Airplanes" are treated as two different words even though they refer to essentially the same thing, the difference only being in quantity. For this reason we use a stemming process in which the suffixes are removed [2]. We used the Porter stemming algorithm[3] a part of the ***NLTK*** python package.

We also removed stop words which is very common words in the English language such as "and", "the" and "of". These words often carry very little meaning, but due to their frequencies they would have had a large influence on the output [4]. Just removing the standard stop-words does not mean that we remove all frequent words which carry little meaning. In our dataset the word "airport" occurs in almost every review and thus do not provide extra information if one wants to distinguish between reviews. To get around this issue we do TF-IDF transformation for the k-means algorithm. TF is the "term frequency" and IDF is the "Inverse Document Frequency". The IDF of a term is given by:

$$IDF(t) = 1 + log\left(\frac{D}{C(D,t)}\right)$$

Where d is the total number of documents and $C(D,t)$ is the count of documents that contains t. TF-IDF is then given by:

$$TFIDF(t,d) = TF(t,d) \cdot IDF(t)$$

Where d is a specific document. TF-IDF thereby gives rare terms a boost, while terms that occur in almost every topic will have a smaller influence on the final clusters [4]. Words that occurred in less than 0.1 % of the documents was also excluded from the dataset. The last important preprocessing step is to generate n-grams. That it is sequences of n words which together gives a more correct meaning. In our case for example the one-gram "free" have different meanings depending on its usage it could refer to "free wifi" or "duty free"(only one them referring to something that is free). Finding n-grams make up for some of the lost information when using the bag-of-words approach. However, on the other hand just because two words occur together once does not make them relevant.

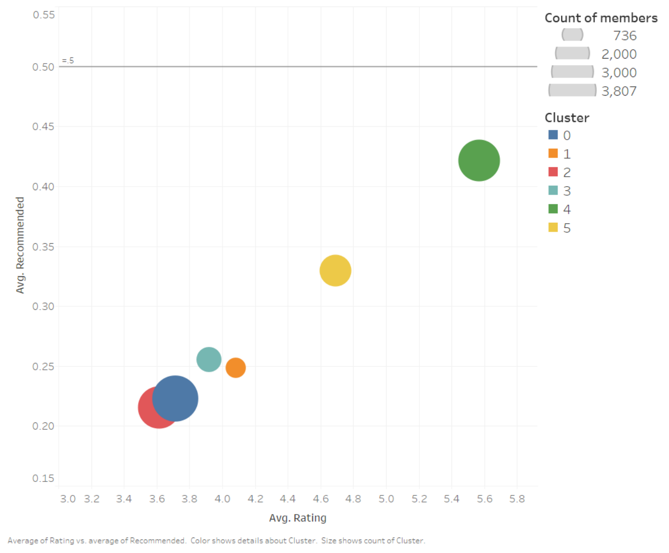
2) Distance measure: To measure the distances between documents we will use the cosine distance. It has the advantage that the length of the document is ignored and it just focuses on the textual content [4] It works better on very long and sparse vectors than Euclidean distance[1]. With TD-IDF transformed data the cosine distance will be a number between 0 and 1, where 0 indicates that they are 90 degrees to each other and that means they have no match [1]. The formula for Cosine similarity can be seen below:

$$sim(x,y) = \frac{x \cdot y}{||x||||y|}$$

Where $||x||$ is the euclidean norm of the vector $x = (x_1, x_2, ..., x_p)$

3) K-means Visualization and results: We have experimented with different visualizations of the clusters. Because of multiple dimensions visualization in meaningful way is difficult. One of the best way to visualize the generated clusters is by Rating (0-10), Recommended (0-1) and cluster size by shape. By doing this visualization, the x-axis represents average overall 0-10 rating, and the y-axis - average recommended (0-1) rating in the cluster. When visualizing clusters this way, each review from the respective cluster is placed on the axis. The result can be seen in the picture below (Fig 2.). From this graph we can assume that the higher overall average rating cluster is also more likely to have higher average recommended value. On the other end of specter we can also see that the lowest average overall rating clusters also have the lowest average recommended value.

Fig. 2. Recommended vs. Overall rating per cluster



We also visualized most relevant words in clusters. This method gave us an idea which topics are the most relevant in the cluster. The results can be seen below in Fig 3. and Fig 4.

Fig. 3. Most relevant words cluster 0

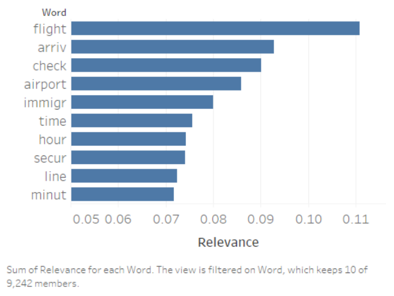
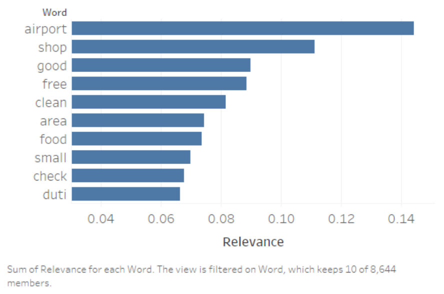


Fig. 4. Most relevant words cluster 4



Relevance is calculated using the formula:

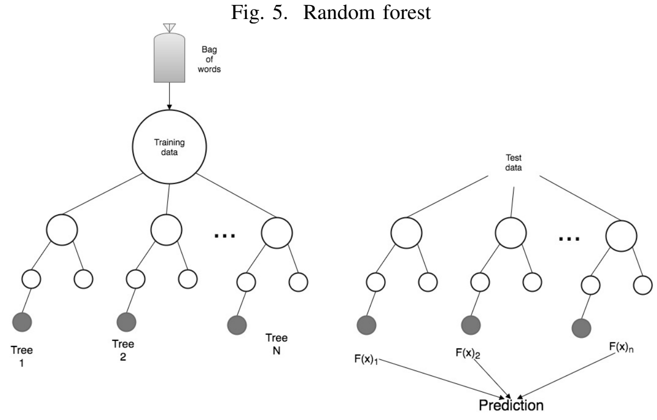
$$r(w,c|\lambda) = \lambda log(p(w|c)) + (1-\lambda)log(\frac{p(w|t)}{p(w)})$$

Where w is a word, c is the cluster and λ is a weight parameter set by the user to balance the to parts. We used $\lambda = 0.6$ as suggested by the study by Sievert and Shirley [5]. As we can see from the graphs in cluster 0, which is on the low end of average overall and average recommended value the most relevant words were ”flight” and ”arrival”. In cluster 5, which is more positive than cluster 0 the dominant words were ’clean’,’good’ and ’shop’.

B. *Prediction using random forest*

To evaluate the different prediction methods we implemented simple algorithm from the bag of words output. For this purpose we chose random forest method which is often used for classification. To classify an object (in our case - recommended value) we put the input vector to each of the trees in the forest. Each tree gives a classification, and we say the tree ”votes” for that class. The forest chooses the classification having the most votes over all the trees in the forest [7].

Fig. 5. Random forest



As an input vector we used output vectors from bag of words method. Beforehand the text was preprocessed - split to individual words, stop words were removed. The results are displayed as confusion matrix:

TABLE I
RANDOM FOREST CONFUSION MATRIX

		Predicted class	
		1	0
Actual class	1	**TP:** 4%	**FN:** 30%
	0	**FP:** 96%	**TN:** 70%

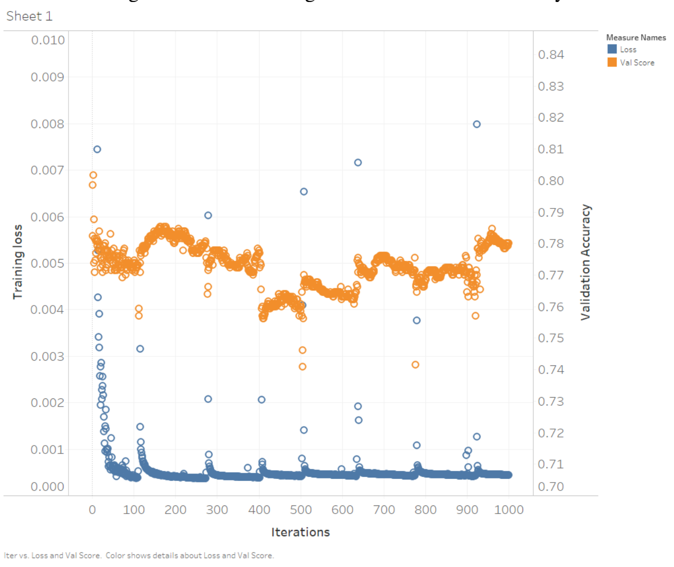
As we can see from the results, the recognition for the positive review are very low, thus we can assume that this method doesn’t satisfy our prediction needs.

C. *Prediction using Multilayer Perceptron*

For our Multilayer perceptron implementation few different libraries were used - ***pandas*** for easy CSV file handling and data extraction, ***sklearn*** for main classifier and bag of words model. We divided our dataset in 70:30 proportion for training and test sets. Each review was extracted from CSV file and transformed using CountVectorizer function from plain text to bag of words binary array. Binary vectors were fitted and predictions were made with test set.

1) Choosing the right number of iterations: To avoid overfitting we plotted the validation accuracy versus the training loss. From the graph one can see that we probably start to overfit after 160-170 iterations since the accuracy on the validation set starts to drop while the training loss continue to decrease. Note that for training loss the first couple of iterations are excluded since they were too high to be shown in the graph. Another interesting observation is that the highest accuracy is achieved in iteration 2 and 3, however this might just be to pure chance.

Fig. 6. Plot of training loss vs. Validation accuracy



2) K-fold cross validation: To make sure that the accuracy of the algorithm could not be attributed to a randomly chosen training and test set, we decided to use 10 fold cross validation to measure the total accuracy of the neural network produced. The data is separated into 10 subsets and then trained 10 times. In each iteration a new subset is kept as a test set thereby giving us 10 measurements of the accuracy of the network [1]. An example is shown in table II for a neural network with two layers of 40 neurons each. One can see that the accuracy change with up to 3 percentage points depending on the training example. Overall the accuracy was on average 79.25 % for this network.

TABLE II
CROSS VALIDATION RESULTS

Iteration	Accuracy
1	79.52 %
2	78.95 %
3	78.89 %
4	80.24 %
5	77.65 %
6	79.28 %
7	78.78 %
8	78.66 %
9	80.41 %
10	80.13 %

D. Experiments with parameters

With implemented Multilayer Perceptron model we performed multiple tests to determine how different parameters influence output results, mainly - recognition accuracy. For all accuracy tests we used 1 layer of 8 hidden neurons which provided optimal results. We tested on three different activation functions - ***logistic*** which is the logistic sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}}$$

tanh which is the hyperbolic tangent function:

$$f(x) = tanh(x)$$

and ***relu*** which is the rectified linear unit function:

$$f(x) = max(0, x)$$

From all these activation functions, the hyperbolic tangent function performed worst in all the cases so it will not be included in the results. In tests we also used different solvers for weight optimization. Solver function ***adam*** refers to a stochastic gradient-based optimizer proposed by Kingma Diederik and Jimmy Ba. Sovler `adam` is a method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients, `adam` also keeps an exponentially decaying average of past gradients [8]. This solver works better on relatively big datasets in terms of both training time and validation score. For reference, we also chose, ***lbfgs*** solver. It is an optimizer in the family of quasi-Newton methods who can converge faster and perform better for some datasets. This method works particularly well with a large number of variables. Results from logistic sigmoid function test:

TABLE III
LOGISTIC SIGMOID FUNCTION CONFUSION MATRIX

		Predicted class	
		1	0
adam solver	1	**TP:** 40.86%	**FN:** 15.69%
	0	**FP:** 58.31%	**TN:** 80.86%
lbfgs solver	1	**TP:** 57.68%	**FN:** 19.13%
	0	**FP:** 42.31%	**TN:** 80.86%

Results from rectified linear unit test:

TABLE IV
RECTIFIED LINEAR UNIT FUNCTION CONFUSION MATRIX

		Predicted class	
		1	0
adam solver	1	**TP:** 38.15%	**FN:** 14.61%
	0	**FP:** 61.84%	**TN:** 85.38%
lbfgs solver	1	**TP:** 54.15%	**FN:** 18.77%
	0	**FP:** 45.84%	**TN:** 81.22%

Our main interest are in **TP** (true positive) and **TN** (true negative) values which let's us determine the reviews outcome. From our tests logistic sigmoid function with lbfgs performed the best with 57.68% for true positive value and 80.86% for true negative. We have to keep in mind, that the dataset is imbalanced as we have only around 30% of positive reviews and that influences our results. Using these parameters another test was performed using ***nltk*** library and stop words removal it provides. After stop word removal the results are:

TABLE V
LOGISTIC SIGMOID FUNCTION AFTER STOP WORD REMOVAL CONFUSION MATRIX

		Predicted class	
		1	0
adam solver	1	**TP:** 66.45%	**FN:** 26.83%
	0	**FP:** 33.54%	**TN:** 73.16%

After stop word removal we observe radical changes to **TP** and **TN** values. **TP** value increased to 66.45% and **TN** decreased to 73.16%. In total we observed 8.77% increase in **TP** value and 7.7% decrease in **TN** value from best test case without stop words removal. Average recognition of True (**TN** and **TP**) values with Logistic sigmoid function and ***lbfgs*** solver without stop words removal:

$$P(x) = \frac{TP + TN}{2} = \frac{57.68 + 80.86}{2} = 69.27\%$$

Average recognition of True (**TN** and **TP**) values with Logistic sigmoid function and ***lbfgs*** solver with stop words removal:

$$P(x) = \frac{TP + TN}{2} = \frac{66.45 + 73.16}{2} = 69.8\%$$

From the calculations we can assume that stop words changes prediction recognition on sentimental level of positive/negative reviews, but the average true recognition stays almost the same.

IV. DISCUSSION

A. Clustering results

We experimented with number of clusters, by creating 3, 6, 10, 20 and 25 clusters. 6 clusters seemed most appropriate based on a qualitative assessment of the visualizations created. For visualization different approaches were tested, the one that worked the best has been presented in figure 2. Furthermore we found it interesting to visualize top-10 words from each cluster. The results suggest that positive reviews tend to speak more about ”shopping”, ”area”, ”food” and it mentions the word ”free”. Negative reviews tend to speak about ”flights”, ”arrival”, ”security” and ”immigration”. From these results we can conclude that most negative feelings people tend to get at the airport which is not working properly such as long waiting lines or delayed flights. As for the positive reviews they usually mention additional amenities such as shopping or cleanliness.

B. Multilayer Perceptron results

We found that the best number of epochs for our neural network is probably in the range of 160-180, as our validation scores started to destabilize after this amount of iterations while the loss on the training data still continued to decrease, indicating some sort of over fitting. One should also be aware of the randomness introduced by different samples. We used cross validation to overcome this problem.
When using Multilayer perceptron for recommended value prediction on reviews, the results were remarkably higher than using random forest or clustering algorithms. Algorithms like random forest can work well on relatively small datasets like filtering spam emails, however it falls short on large datasets in which topics vary. Comparing different activation functions and solvers for our multilayer perceptron gave us very different results. For every test we ran the hyperbolic tangent function performed worse and for that reason we did not include it in the results. Logistic sigmoid function and rectified linear unit activation functions performed quite similar. In all cases ***lbfgs*** gave us better results, however fitting data with this solver took more time. It is understandable as ***adam*** supposed to work best for the large datasets as stated in the research [8]. The most intriguing result we got when comparing prediction accuracy before stop word removal with ***nltk*** library and after that. After stop word removal we notice a big increase towards positive review recognition, but also a drop for negative reviews. The calculated true average value for both cases showed that the average recognition stayed the same. We can conclude that removing stop words helped to highlight more influential positive words in reviews, and also - hide some negative ones which led to **TN** accuracy decrease. Therefore one should think carefully before removing stop words [4] since they might carry some information in sentiment analysis. It also shows that sometimes there can be trade-offs between getting high accuracy among positive or negative reviews.

C. Future work

We have not been able to implement all desired approaches and techniques. In addition to existing clustering approach, it would be interesting to cluster based on other attributes e.g. length of the text, country and queuing rating etc. The visualization can be done on other attributes. Another way of clustering the reviews could be based on positive and negative words. This can be done by having list of positive and negative words, and clustering the reviews based that.
From the multilayer perceptron perspective a few things can be improved. For stop word removal a new list of stop words could be made specifically for our dataset. The stop words from ***nltk*** library don't include words like ”airport” or ”car”

and these words don't give any valuable meaning or influence towards determining if the review is positive or negative. One interesting procedure for future testing would be to remove all the words except adjectives and n-gram words (like "airport security") and try to fit the model with it. For the one part this method would lower the processing time as the review length would decrease. Also it should help the algorithm to focus on the most influential words. Another interesting technique to test would be implementing the convolutional neural networks. In our multilayer perceptron case our network is static which means we have fixed number of hidden layers and neurons.

V. Conclusion

After K-means clustering we found out that although clusters can not be used for classification. We discovered some patterns among the main topics in the clusters and the average rating. This can be used by airports to focus on areas to improve customer experience.

In our opinion the overall performance of multilayer perceptron with bag of words input was acceptable. From the different test cases we manage to find the highest accuracy activation function and solver. In general the multilayer perceptrons produced respectable accuracy. We also discovered an interesting relation between stop word removal and accuracy change. It was seen that stop word removal managed to extract more features for predicting positive review, but also lowered accuracy of negative once. These features can be researched more in future work using more sophisticated algorithms such as convolutional neural networks.

References

[1] J. Han, M Kamper and J. Pei *Data Mining - Concepts and Techniques* 3rd ed. Morgan Kaufmann Publishers Inc. 2012

[2] S. Debortoli, O. Müller, I. Junglas and J. vom Brocke *Text Mining For Information Systems Researchers: An Annotated Topic Modeling Tutorial* Communications of the Association for Information Systems: Vol. 29, Article 7. 2016

[3] M.F. Porter *An algorithm for suffix stripping* Program, Vol. 14 Issue 3, pp.130-137

[4] F. Provost and T. Fawcett *Data Science for Business - What you need to know about data mining and Data-Analytical Thinking* 1st ed. O'Reilly, 2013

[5] C. Sievert and K.E. Shirley *LDAvis: A method for visualizing and interpreting topics.* Proceeding of the workshop on interactive language, learning visualization and interfaces, 2014.

[6] H. Shimodaira *Text Classification using Naive Bayes* Learning and Data Note 7. 2015 February 10

[7] L. Breiman and A. Cutler *Random forests* Date Accessed - May 13, 2017. Retrieved from https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

[8] Diederik P. Kingma, Jimmy Lei Ba *Adam: A Method For Stochastic Optimization* arXiv:1412.6980v9 [cs.LG] 30 Jan 2017