

---

# YELP SENTIMENT ANALYSIS

---

**Moritz Wilksch**

Faculty of Economics and Social Sciences  
Potsdam University  
Potsdam  
wilksch@uni-potsdam.de

**Sofya Marchenko**

Faculty of Economics and Social Sciences  
Potsdam University  
Potsdam  
marchenko1@uni-potsdam.de

**Ferdinand Hoske**

Institute of Computer Science  
Potsdam University  
Potsdam  
hoske1@uni-potsdam.de

July 26, 2020

## ABSTRACT

In this project we built a model for sentiment analysis of Yelp user reviews. It classifies whether the tone of a review text is positive, negative or neutral. The star label of a review that is always assigned by a user along with his opinion serves as a label making this a supervised learning approach. We compare the performance of pre-trained *fasttext* word embeddings with self trained word embeddings on top of a convolutional neural networks (CNN) in with we experiment with various architectures and filter sizes to explore their effect on the performance.

## 1 Introduction

Yelp is an American internet company founded in 2004, who hosts a platform for crowd-sourced local business reviews and social networking. The main purpose of this platform is to provide information about local businesses such as restaurants, shops, hotels, and other services that people can rely on. On Yelp users write reviews in open-comment style along with a 1 - 5 stars rating, where a higher rating indicates more satisfaction with the service. With the growing popularity of social media and mobile devices, potential consumers are influenced heavier by such online reviews. A study shows that an increase in one star Yelp rating increases the revenue by 5 to 9 percent [2]. Thus, understanding what kind of attitudes customers post on the website is crucial for business owners and other stakeholders. Due to the steady growing amount of content, it is extremely challenging to determine the overall opinion manually. Thus automatic sentiment analysis is a task that became a popular research field in the past decade. Further a classification system can aid users in getting a representative opinion and thus assist in the decision-making process.

There are two approaches to sentiment analysis: dictionary-based and machine learning-based. Machine learning-based methods require labeled training data that are later represented as features and fed into a classifier. Sentiment Analysis is usually treated as a classification problem as the goal is to determine the polarity of a text (e.g., positive or negative).

The aim of this project is to predict a three category sentiment of a review solely based on its text. For this task we will use a supervised learning. We set a preliminary baseline for the performance of our deep neural network by training a Naive Bayes classifier. Further, we compare the performance of various multi layer CNN architectures together with pre-trained and self trained an word embeddings.

## 2 Related Work

There are various approaches to perform a automatic sentiment analysis classification with machine learning techniques. In the past traditional classification methods including the Naive Bayes classifier, Maximum Entropy, and Support

Vector Machines to solve such problems were reported [13] [14]. Eventough these algorithms lead to good results, modern approaches enabled by GPUs, based on deep learning models perform even better. Researchers propose CNNs for sentiment analysis and text classification on large-scale databases [12]. Other successful approaches for sentiment classification include recurrent neural networks or Long Short-Term Memory models (LSTM) [15]. However there is not one optimal approach that fits all kinds of sentiment classification and all kinds of data sets.

!One paragraph on word embeddings Word2Vec,Glove etc.

Specifically, for Yelp ratings there are several prior projects on classifications, that have various approaches in terms of how to capture the sentiment of a review. Classifying the reviews according to their star label as a numerical scale [7] is one possibility. Others drop the 3- star ratings and use solely the polarity (*positive vs. negative*) of the review [8]. As these neutral reviews are especially hard to classify, this approach gives a high prediction accuracy for the two remaining classes. We choose to keep the neutral reviews to examine how well these non-polar texts can be classified. These reviews might be of special interest, as they are likely to contain detailed explanations about what costumer enjoyed and suggested improvements.

### 3 Data Set

#### 3.1 Data Description

For this project we used a publicly available data set from Yelp [1]. The original data set contains information on over 8 million reviews and has a size of approximately 10 GB. Besides the star rating and the review information the data set provides information on the id of the user that wrote the review and the id of the business the review is written for. The data is stored in a json format. As for the sentiment analysis we are only interested in the text of the review and the star rating, we reduced the data set to these two features. Due to the limited time frame of our project, we decided to reduce the size of the data set to 400.000 reviews. Which were divided into a training, validation and a test set with the ratios 0.75, 0.15, 0.10 respectively. We mapped the star labels to three classes for the sentiment analysis as following:

- 0-2 stars : negative (encoded as -1)
- 3 stars : neutral (encoded as 0)
- 4-5 stars : positive (encoded as 1)

After mapping, we created a balanced training data set that contains an equal number of reviews for each class. This subsampling approach has been reported to be effective in unbalanced data sets [4]. For validation and test sets we use random subsamples from the original data set, thus aiming to keep the class distribution from the original data set to obtain a valid performance estimate in order to get a realistic performance estimation of the model [5].

#### 3.2 Preprocessing

There are several steps that natural language texts need to undergo before they can be used for training a neural network. First we tokenize every review into single words. Subsequently, we remove punctuation and convert all capital letters to lower case. We used Porter's algorithm for removing the commoner morphological and inflexional endings of words (stemming) [6]. We do not remove any stop words, as using precompiled lists of stop words has been reported to have a negative impact on the accuracy when it comes to sentiment classification [10] [11].

Before the preprocessed texts can be used to train a model they have to be numerically represented. Word embeddings are a way of representing words to be given as input to a Deep learning model. In this method, each word is represented as a word vector in a predefined dimension. Pre-trained vectors provide a good starting point for the model to learn.

Research shows that the choice of vector input representation has an impact of the performance of the sentiment meaning [8]. There are several pre-trained word embeddings available, most of which have been trained on extensive corpora like Wikipedia pages or Google News articles. To capture the semantic diversity of natural language most models employ embedding vectors of size 300. These vectors are learned either in a continuous-bag-of-words model (predicting a single word form a context) or a skip-gram model (predicting context from a single word). In this project we will use fasttext vectors. The fasttext model has been trained on the Wikipedia corpus [SOURCE] using the skip-gram model. Due to resource constraints we extract only the word vectors from the fasttext model that occur in the training data set. We are aware that this does not use the fasttext model to its fullest extent as it prohibits the use of sub-word embeddings for unknown words that fasttext would normally offer.

As convolutional neural networks require the input sequence length to be fixed, we decided to unify the length of all inputs to 183 words. Inputs shorter than this value will be zero-padded, longer inputs will be truncated. The length of 183 words represents the 90th percentile of the number of words per review on the training data.

The lexical preprocessing steps were conducted with the Python library Gensim, a powerful NLP toolkit. For preprocessing of the sequence we used keras.

## 4 Architecture and Training

As a starting point for modelling we employ a multi layer CNN architecture as reference [9]. This model consists of one embedding layer that transforms word indices into word vector by a lookup operation in the pre-trained embedding matrix, followed by three one-dimensional convolutional layers, each with 100 filters of sizes 3, 4, 5. Each convolutional layer is using a rectified linear unit (ReLU) activation function. In between these convolutional layers max pooling is applied to reduce the dimensionality. The whole architecture of this model is displayed in Fig. 1. A dropout is a technique used to reduce overfitting to the training set [3]. In each training stage individual nodes are dropped with probability  $p$ , the reduced neural net is updated and then the dropped nodes are reinserted. We apply Dropout to the hidden layer and to the input layer using  $p = 0.5$  in both cases.

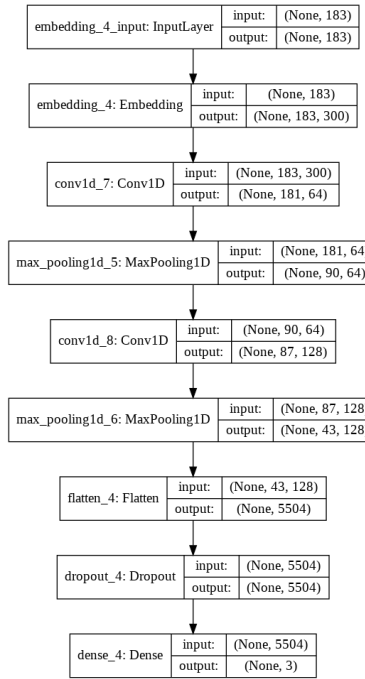


Figure 1: Base network architecture.

In the training phase, we used a batch size of 256 and 10 epochs.

## 5 Evaluation

To evaluate the prediction performance of the models created, we compared the actual sentiment in the validation data to the output of the model. Thus based the three classes were able to compute the number of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). Based these counts we computed precision, recall for each of the classes and accuracy to evaluate our model.

Precision is given by

$$precision = \frac{TP}{TP + FP}$$

Recall is the number of true positive out of the actual positive reviews, and it is given by

$$recall = \frac{TP}{TP + FN}$$

Accuracy refers to the proportion of correct classified reviews to the whole data set

$$acc = \frac{TP + TN}{TP + TN + FP + FN}$$

here an accuracy of 1.0 would mean that each item was classified correctly.

To assess the performance of any machine learning model, and for deep learning models in particular, one should compare it to a very simple baseline model. In our case, a natural language processing problem, the go-to baseline model is a Naive Bayes classifier. It is fast to train and can deliver surprisingly good results. In this work, we use a Naive Bayes model that is trained on a bag-of-words (BOW) representation of the training corpus. As in this BOW setting each feature simply models the number of times one specific word occurs in a document, it follows a multinomial distribution. We therefore use a Multinomial Naive Bayes classifier. Its performance on the test set looks as presented in Table 1.

Model	Positive	Precision		Positive	Recall		Accuracy
		Neutral	Negative		Neutral	Negative	
Naive Bayes	0.91	0.28	0.71	0.72	0.72	0.70	0.72

Table 1: Results from baseline Multinomial Naive Bayes model on test data.

## 6 Experiments and Results

In this section we report the performance obtained from various adjustments to our initial model ( $M_1$ ) that were made to architecture and embedding. First, we removed the last convolutional layer ( $M_2$ ), afterward we removed the last two convolutional layers ( $M_3$ ). Then we added and increased the filters of  $M_2$  to 64 and 128 ( $M_4$ ) and added a custom third layer with filter 256. The filter sizes were set to 3/4/5 ( $M_5$ ). Next we reversed the filter sizes to 5/4/3 ( $M_6$ ). Further  $M_4$  was tested with only size 2 filters ( $M_7$ ) and we also added a dense layer on top ( $M_8$ ). The results are presented in Table 2.

Model	Positive	Precision		Positive	Recall		Accuracy
		Neutral	Negative		Neutral	Negative	
$M_1$	0.96	0.28	0.77	0.73	0.39	0.80	0.74
$M_2$	0.95	0.33	0.81	0.81	0.64	0.81	0.79
$M_3$	0.95	0.32	0.80	0.80	0.64	0.81	0.79
$M_4$	0.95	0.35	0.80	0.83	0.60	0.81	0.80
$M_5$	0.95	0.31	0.78	0.79	0.63	0.79	0.78
$M_6$	0.95	0.30	0.81	0.78	0.68	0.77	0.76
$M_7$	0.96	0.32	0.85	0.80	0.69	0.79	0.79
$M_8$	0.96	0.30	0.86	0.79	0.75	0.71	0.76

Table 2: Accuracy results from architectures with pre-trained embeddings on validation data.

Looking at the accuracy we observe that we were able to improve the performance of the initial model with the best performance reached in  $M_4$  of an accuracy of 0.8.

Next we investigated how using our self-trained word embeddings affects the performance. First we used the architecture of  $M_4$  with 50 dimensional embeddings ( $M_9$ ), then we adjusted the dimension of the embeddings to 100  $M_{10}$  and last to 300  $M_{11}$ . The performance of these CNN models is displayed in Table 3.

Model	Positive	Precision		Positive	Recall		Accuracy
		Neutral	Negative		Neutral	Negative	
M <sub>9</sub>	0.95	0.28	0.77	0.76	0.64	0.78	0.75
M <sub>10</sub>	0.95	0.29	0.78	0.77	0.64	0.76	0.75
M <sub>11</sub>	0.95	0.27	0.76	0.75	0.62	0.77	0.74

Table 3: Accuracy results from architectures with own embeddings on validation data.

## 7 Discussion

In this project, we have compared CNN model architectures for classifying Yelp reviews into positive, negative or neutral sentiment. The best performed model improves the performance of the benchmark model by 8% resulting in an accuracy of 80%.

Points to touch on:

- Simpler CNN generalized better (for the small data set we used)
- Neutral reviews are hard to classify
- Training for more epochs (espc. for self trained Embeddings) made the model over fit
- Pre-trained vectors worked best (again: is this due to the small training corpus? >3M trainable params are too much for the small corpus)
- Final accuracy around 79-80% is pretty decent, other benchmark [8] achieves 89% (with fasttext on small corpus) but without the neutral class!
- State-of-the-art using fasttext on whole data set achieves accuracy of around 94% (once again, without the neutral class)

## References

- [1] <https://www.yelp.com/dataset>. accessed: 10.07.2020.
- [2] Michael Luca Reviews, Reputation, and Revenue: The Case of Yelp.com. In *HBS*, 2011.
- [3] Nitish Srivastava and Geoffrey Hinton and Alex Krizhevsky and Ilya Sutskever and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. In *Journal of Machine Learning Research*, pages 1929-1958, 2014.
- [4] Nitish Srivastava and Geoffrey Hinton and Alex Krizhevsky and Ilya Sutskever and Ruslan Salakhutdinov. Joint use of over- and under-sampling techniques and cross-validation for the development and assessment of prediction models. In *BMC bioinformatics*, 16(1), pages 1929-1958, 2015.
- [5] I. Triguero, M. Galar, S. Vluymans, C. Cornelis, H. Bustince, F. Herrera and Y. Saeys. Evolutionary Undersampling for Imbalanced Big Data Classification. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 715-722, 2015.
- [6] M. F. Porter An algorithm for suffix stripping In *Program: electronic library and information systems*, Vol. 14 No. 3 pages 130-137, 1980.
- [7] Y. Xu, X. Wu and Q. Wang Sentiment Analysis of Yelp’s Ratings Based on Text Reviews. 2014.
- [8] A. Salinca Convolutional Neural Networks for Sentiment Classification on Business Reviews. 2017
- [9] A. Zhang and Z. C. Lipton and M. Li and A. J. Smola Dive into Deep Learning 2019
- [10] T. Renault Sentiment analysis and machine learning in finance: a comparison of methods and models on one million messages. In *Digital Finance*, 2019, pages 1-13.
- [11] H. Saif, Y. He, M. Fernandez and H. Alani Contextual semantics for sentiment analysis of Twitter. In *Information Processing and Management* 52(1), 2016, pages 5-19.
- [12] A. Severyn and A. Moschitti Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th international workshop on semantic evaluation*, 2015, pages 464-469.
- [13] C. Troussas, M. Virvou, K. J. Espinosa, K. Llaguno and J. Caro. Sentiment analysis of Facebook statuses using Naive Bayes classifier for language learning. In *IISA*, IEEE, 2013, pages 1-6

- [14] A. Go, R. Bhayani and L. Huang. Twitter sentiment classification using distant supervision. *CS224N project report, Stanford*, 1(12), 2009.
- [15] J.H. Wang, T. W. Liu, X. Luo and L. Wang. An LSTM approach to short text sentiment classification with word embeddings. In *Proceedings of the 30th conference on computational linguistics and speech processing (ROCLING 2018)*, 2018, pages 214-223.