Predicting Popular News Comments

Johannes Filter
Hasso-Plattner-Institute
Potsdam, Germany
johannes.filter@student.hpi.de

Cornelius Hagmeister
Hasso-Plattner-Institute
Potsdam, Germany
cornelius.hagmeister@student.hpi.de

Thomas Kellermeier
Hasso-Plattner-Institute
Potsdam, Germany
thomas.kellermeier@student.hpi.de

ABSTRACT

As the amount of comments that is posted daily on online newspapers increases, it becomes more and more difficult to moderate these sections. While this is mostly done manually, the advances in technology allow for machine supported moderation. In this paper we analyze how to automatically distinguish between popular and unpopular news comments. Based on data from the Guardian, we introduce a preprocessing which allows us to segment data based on normalized user up-votes. We create different datasets using only a part of the most popular and least popular comments, while discarding part of the comments to increase the contrast. In our experiments we use several classifiers, where our best classifier achieves an accuracy of 72%.

KEYWORDS

Natural Language Processing   Deep Learning   Supervised Learning

ACM Reference format:
Johannes Filter, Cornelius Hagmeister, and Thomas Kellermeier. 2018. Predicting Popular News Comments. In *Proceedings of Text Mining in Practice Seminar 2018, Potsdam, Germany, July 2018 (TMIP'18),* 4 pages.
DOI: 10.475/123_4

1 INTRODUCTION

Online newspapers are in an endeavor to cope with the vast amount of user comments. While there are certainly high-quality comments that enrich the articles, there are also nonconstructive comments ranging from toxic to nonsense. To keep a good discussion culture, newspapers are required to moderate the comments. But because of the economic pressure within the news industry, resources are sparse. As a consequence, a large number of newspapers restrict their comment section. Considering the increasing availability of data, machine learning might be suited for supporting journalists to manage news comments. In that bigger picture, the goal of this work is to predict the popularity of news comments by only considering a comment's text. Since labeled data is scarce, we preprocess the comment's up-votes to turn the task into a binary classification problem.

There is a long history of supporting journalistic work with computers. It started in the late 60s with computer-assisted reporting[1] and leading to current ideas about automatic reporting[2]. Our work is focuses on different area: Supporting journalists in managing news comments. There is work done by Park et al. in the field of Human-Computer Interaction [15] where they build an end-to-end system incorporating feature-based traditional machine learning. Recent work in the research area of comment analysis focuses on identifying high quality or *constructive* comments on e.g. New York Times website [11][4][10]. Other research focused on identifying valuable discourses[5][14]. Some earlier work analyzes user-generated content on Slashdot[3] [12][6], and predicts popular content on Digg[4] and YouTube[5] [17].

The remainder of this paper is structured as follows: In Section 2 we present our preprocessing and how we turned up-votes into binary labels. Afterwards we describe the applied machine learning methods in Section 3 before evaluating these in Section 4 and finally concluding in Section 5.

2 PREPROCESSING

The dataset that is used in this paper consists of about 61 million comments that were extracted between March 2006 and November 2017 from the online edition of the Guardian [6]. They were sampled from all categories of the American as well as of the British edition. The criteria to distinguish good comments and bad comments in this paper is the number of received up-votes for each comment. Because the number of up-votes a comment received might not be significant, it is in many cases difficult to classify comments properly. We therefore propose to filter and preprocess the comments, so that we can train our networks on the remaining comments with more clear classes. The following list shows an overview over our preprocessing steps, which we will explain in more detail afterwards.

(1) Consider politics category only
(2) Filter non root comments
(3) Remove articles with few comments
(4) Remove articles with few up-votes
(5) Rank comments by chronological order
(6) Consider first ten comments per article only
(7) Label most popular and least popular comments

To have a dataset with uniform topics and similar wording, we restrict our training data to one category of comments. In this paper

[1]https://en.wikipedia.org/wiki/Computer-assisted_reporting
[2]https://en.wikipedia.org/wiki/Automated_journalism
[3]https://slashdot.org/
[4]http://digg.com/
[5]http://youtube.com/
[6]https://www.theguardian.com

we exclusively train on comments from the politics category, since we assumed the most controversial and emotional discussions to be among these comments.

We further filter all comments that were not root comments, because replies on other comments depend on the context of these local discussions. Training on each comments separately would lack these contextual dependencies. To avoid having comments with no feedback, articles that received less then ten comments or where the first ten comments did not receive more than 20 up-votes in total are filtered, too. Having articles with many comments and up-votes ensures a higher quality of labels.

Since comments that are more visible are also more likely to get up-votes and the Guardian uses the time of creation to order comments, we restrict our training comments on the first ten published comments per article. We call the position of a comment within the chronological order its rank.

To make comments from articles with varying responses comparable, the up-votes are normalized using the total up-votes of the first ten published comments. In the following work we call these extracted values the relative up-votes. Because we want to put more emphasis on the content itself, the influence of the comments rank on the label is neglected. We achieve this by creating datasets with an equal amount of popular and unpopular comments for each rank.

To measure the influence on relative comment up-votes we use three datasets in our experiments. The first one consists of 10% of the comments with the highest relative up-votes and 10 % with the lowest relative up-votes. Accordingly we created a dataset consisting of the best and worst 25% and 50% comments according to the relative up-votes. This results in datasets with 20.360, 53.400 and 106.000 comments, which will be called *pol-10*, *pol-25* and *pol-50* in the remainder of this paper.

3 METHODS

In this section we will present the network architecture we have chosen for our experiments including a feature-based approach, but also more sophisticated Convolutional Neural Networks (CNN) and Gated Recurrent Unit (GRU) networks. While the feature-based approach focuses on meta-informations with clearly defined features, the CNN as well as the Recurrent Neural Network (RNN) approaches are focused on the content of the comments and try to extract language features, which define the quality of a comment. These architecture have shown to yield good results in the context of Natural Language Processing [18].

3.1 Feature-based Classification

As our baseline we use a feature based model based on work by Park et al. [15]. They present an approach to support moderators in the ranking of comments based on features, which were chosen based on interviews and surveys. Since we are mainly focusing on the comments and not the articles in this paper we chose the following features for our model:

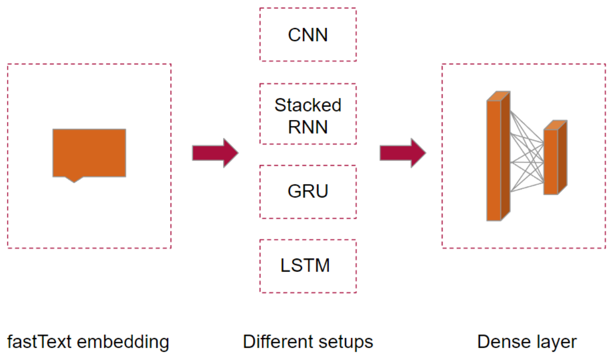

Figure 1: The general setup of our experiments. Using fastText embedding to convert input into vectors, then using different neural networks architectures to extract features and 2 dense layers to scale the features back to a binary output.

Used Features
Comment length
Comment readability
Average comment length per user
Average comment readability [13] per user
Average comment up votes per user

These features consist of information about a specific comment as well as information about average values for the author of the comment. To process the features we use a network consisting of 3 fully-connected dense layers with the sizes 400, 100 and 1.

3.2 Deep Learning Architectures

To further push the accuracy we achieve with our baseline, we set up multiple experiments utilizing different cell architectures. A pretrained fastText [1] embedding is inserted for feeding the comments to our network. The general setup can be seen in Figure 1, which consists of a pretrained fastText embedding layer.

Convolutional Neural Network

This approach is based on Kim's work. [8], which uses 1-dimensional CNNs to extract features for adjacent words in the input sentences. The advantage of using CNNs instead of RNNs lies in the low complexity of the network i.e. few variables because of the convolution and the max-over-time-pooling, which makes the network comparably easy to train.

LSTM and GRU

To allow our model to also detect long-term dependencies between words, we make use of LSTMs [7] and GRU cells [2]. Besides being more powerful, LSTM networks tend to be difficult to train, especially on small datasets. GRU cells have a similar structure, but are only based on 2 gates to weigh input and output. This makes them less powerful, but as experiments have shown GRU based networks tend to perform better on certain tasks [3]. Especially on small datasets gated recurrent units tend to show a good performance.

Stacked Recurrent Neural Networks

As a common enhancement for DNNs we add layers of abstractions to RNN networks by stacking multiple RNNs. Stacked LSTMs were introduced in 2013 by Graves, et al. [16] in their application of LSTMs in speech recognition, which performed better than common networks on the task of language modeling. They therefore proved, that LSTMs can not only profit from depth in time, which is inherently provided by the long-short term memory, but also depth in space which is provided through multiple stacks of LSTM layers. We experimented with stacked LSTMs as well as with stacked GRUs.

4 EVALUATION

To evaluate our preprocessing from Section 2 we set up experiments with the aforementioned neural networks.

4.1 Experiment Setup

For our experiments we used the three datasets presented in Section 2: *pol-10*, *pol-25* and *pol-50*. The data was split into training, validation and test set with a ratio of 80:10:10. As metric we chose accuracy because the number of positive and negative samples is equal. For the word embedding, we used fastText embeddings [1] that were created on all, over 30 million Guardian comments with 300 dimensions, using the skipgram mode. The whole text is lowercased and user mentions and URLs are replaced with special tokens. When training for the actual classifier, the weights for the embeddings were frozen. For setting the hyper-parameters, used random search. As optimizer, we used an Adam-optimizer [9].

As a baseline, we used the feature-based classifier as presented in Section 3.1. We compared it to the performance of our five word-embedding-based models presented in Section 3.2.

4.2 Results

After tuning the hyper-parameters on the validation set, the results are gathered on the test set.

Experiment on pol-10. The smallest dataset *pol-10* consists of 20.360 comments. All DNNs have a small advantage compared to the baseline, as can be seen in Figure 2. The best accuracy 0.7196 was achieved by the CNN. We did not manage to train the more complex models for this dataset.

Experiment on pol-25. For the second dataset *pol-25* with 53.400 comments, all methods have a significant increase of accuracy compared to the baseline. Our result as can be seen in Figure 3 that a stacked GRU network, achieves the best accuracy, which might be related to the fact that more data was available to train on. In this setting the CNN achieved the worst accuracy as a DNN, but it is still significantly better than our baseline. Furthermore both GRU-based approaches tend to be a bit better than the LSTM-based approaches.

Experiment on pol-50. In Figure 4 we present the results for our biggest datasets *pol-50* with 106.000 comments. It contains all available data after applying our filtering steps presented in Section 2. The accuracy generally decreases again for all methods while the

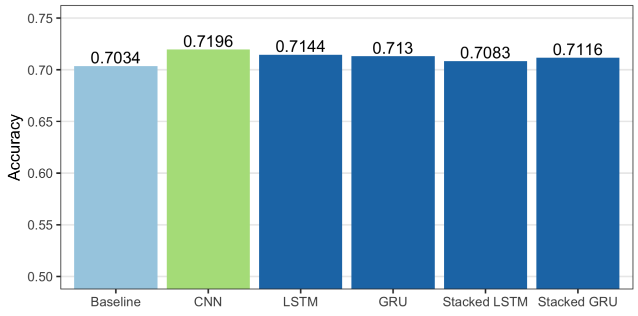

Figure 2: Accuracy on dataset *pol-10*, with only the 10% most popular comments as positive examples and 10% least popular comments as negative examples.

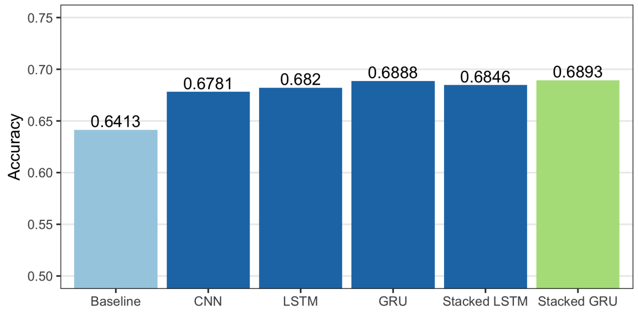

Figure 3: Accuracy on dataset *pol-25*, with only the 25% most popular comments as positive examples and 25% least popular comments as negative examples.

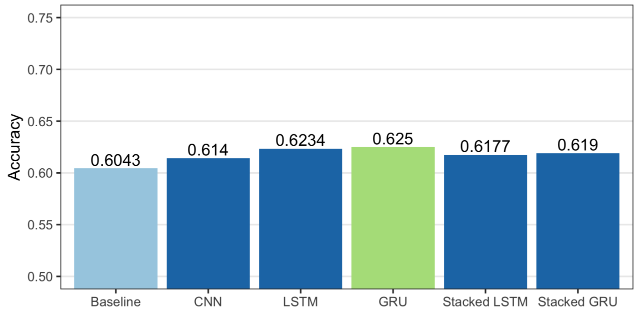

Figure 4: Accuracy on dataset *pol-50*, with the 50% most popular comments as positive examples and 50% least popular comments as negative examples.

GRU-based methods again have a small advantage. The stacked GRU performs worse than the single GRU. Most likely, we did not manage to find an appropriate configuration for the hyper-parameters.

As an overview of our results we compare the baseline against the DNN that performed best on each of the three datasets in Figure 5. In the following section we will discuss our results.

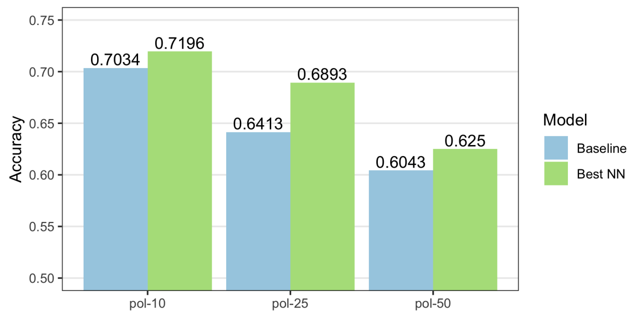

Figure 5: For each dataset we compare our baseline with the best-performing NN model.

4.3 Discussion

Our experiments show that there is a significant trade-off between high contrast labels and a high amount of data. The smallest dataset *pol-10* which has the highest contrast between popular and unpopular comments enables us to achieve a relative high accuracy. In exchange, the DNNs do not perform much better than the baseline, because they usually require a lot of data. The same phenomenon can be observed for the biggest dataset, which consists of a high amount of data, but in return makes it difficult to label comments close to the decision boundary. Finally the results for dataset *pol-25* confirm our hypothesis, as well. The accuracy between baseline and DNN differs by about 5 percent. This lets us conclude that the comments content is worth looking at for predicting a comments quality, rather than just looking at meta-information like the comments length.

Regarding the performance for *pol-50*, considering that 50% accuracy is the worst result for a binary problem, our results let us infer that the data is too noisy. The predefined labels might be too difficult to predict for the comments around the border between the two classes. Only one more up-vote may change a label from positive to negative and vice versa.

A shortcoming is that we did not aim for real-world usage. For instance, we did not make use of FastText n-gram features. Because the embedding were trained on 30 million comments, we know that there are no unknown words. This should be addressed in future work. We also did not split up our data time-based, which looking at the use case of predicting the quality of newly published comments, should be addressed in future experiments.

5 CONCLUSIONS

We first use up-votes of news comments to label them as popular and unpopular. With the same method, we constructed three different datasets. Second, we use feature-based and word-embedding-based artificial neural networks to solve the binary classification problem. We bet our baselines and achieved the highest accuracy when only considering the most and the least popular comments. There are several shortcomings when it comes to the real-life usage. For future work, one should consider the n-gram features of fastText. And also evaluate how far a model can predict into the future. Due to the fast-paced nature of the news, this is a special constraint we neglected in this work. In addition, we currently do not relate a comment to its article or other comments. This is a rather big simplification and should be considered in future.

REFERENCES

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606* (2016).

[2] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[3] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[4] Nicholas Diakopoulos. 2015. Picking the NYT Picks: Editorial Criteria and Automation in the Curation of Online News Comments. 5, 1 (2015), 20.

[5] Nicholas Diakopoulos and Mor Naaman. 2011. Towards Quality Discourse in Online News Comments. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work (CSCW '11)*. ACM, New York, NY, USA, 133–142. DOI: http://dx.doi.org/10.1145/1958824.1958844

[6] Vicenç Gómez, Andreas Kaltenbrunner, and Vicente López. 2008. Statistical Analysis of the Social Network and Discussion Threads in Slashdot. In *Proceedings of the 17th International Conference on World Wide Web (WWW '08)*. ACM, New York, NY, USA, 645–654. DOI: http://dx.doi.org/10.1145/1367497.1367585

[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[8] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *CoRR* abs/1408.5882 (2014). http://arxiv.org/abs/1408.5882

[9] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). http://arxiv.org/abs/1412.6980

[10] Varada Kolhatkar and Maite Taboada. 2017. Constructive language in news comments. In *Proceedings of the First Workshop on Abusive Language Online*. 11–17.

[11] Varada Kolhatkar and Maite Taboada. 2017. Using New York Times Picks to Identify Constructive Comments. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism*. 100–105.

[12] Cliff Lampe and Paul Resnick. 2004. Slash (dot) and burn: distributed moderation in a large online conversation space. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 543–550.

[13] G Harry Mc Laughlin. 1969. SMOG grading-a new readability formula. *Journal of reading* 12, 8 (1969), 639–646.

[14] Courtney Napoles, Joel Tetreault, Aasish Pappu, Enrica Rosato, and Brian Provenzale. 2017. Finding good conversations online: The Yahoo News annotated comments corpus. In *Proceedings of The 11th Linguistic Annotation Workshop*. 13–23.

[15] Deokgun Park, Simranjit Sachar, Nicholas Diakopoulos, and Niklas Elmqvist. 2016. Supporting Comment Moderators in Identifying High Quality Online News Comments. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1114–1125. DOI: http://dx.doi.org/10.1145/2858036.2858389

[16] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026* (2013).

[17] Gabor Szabo and Bernardo A Huberman. 2010. Predicting the popularity of online content. *Commun. ACM* 53, 8 (2010), 80–88.

[18] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative Study of CNN and RNN for Natural Language Processing. *CoRR* abs/1702.01923 (2017). http://arxiv.org/abs/1702.01923