Identifying Clickbait Posts on Social Media with an Ensemble of Linear Models

The Carpetshark Clickbait Detector at the Clickbait Challenge 2017

Alexey Grigorev
OpenDataScience

arXiv:1710.00399v1 [cs.IR] 1 Oct 2017

ABSTRACT

The purpose of a clickbait is to make a link so appealing that people click on it. However, the content of such articles is often not related to the title, shows poor quality, and at the end leaves the reader unsatisfied.

To help the readers, the organizers of the clickbait challenge[1] asked the participants to build a machine learning model for scoring articles with respect to their "clickbaitness".

In this paper we propose to solve the clickbait problem with an ensemble of Linear SVM models, and our approach was tested successfully in the challenge: it showed great performance of 0.036 MSE and ranked 3rd among all the solutions to the contest. The code for the solution is available on GitHub[2].

1. INTRODUCTION

Similar to *baits* that fishermen use to catch fish, clickbaits aim to catch the readers by getting their attention with an appealing title. These titles intent to spark the curiosity of the readers such that they want to follow the link.

The following titles are good examples of clickbaits:

- Man tries to hug a lion. You won't believe what happens next!
- 21 celebrities who ruined their faces with plastic surgeries
- 15 hilarious tweets of stupid people

However, making the reader click the link and then displaying advertisements is often the sole purpose of the clickbaits – and the quality of the content is secondary for the publishers. The curiosity is usually not satisfied and the reader leaves the website.

Social media websites such as Twitter or Facebook contain a large amount of clickbaits. It is possible to save the time of the readers by detecting if a title is clickbaiting and warning them, or even hiding the link altogether. This was the motivation of the clickbait challenge (http://www.clickbait-challenge.org/): the organizers invited participants to develop a machine learning model for predicting if a post on social media is a clickbait. They prepared a large dataset of posts from Twitter with each post labeled as clickbaiting or not.

1.1 Related Work

It is not the first time the research community is concerned with the problem of detecting clickbaits.

It is believed that Facebook downscores clickbaits in the user feeds, but there are no publications on their approach, only an announcement on their newsroom blog[3].

Use of Machine Learning to treat this problem was first reported by Potthast and others [9], who collected a corpus of posts from Twitter and used crowdsourcing to annotate them.

To discover if their results can be improved, the same researches later organized the present challenge – the clickbait challenge [10] on the TIRA platform [8]. In addition to the previous dataset, they gathered more data and released a significantly larger corpus of annotated posts from Twitter [11].

2. DATASET AND CHALLENGE DESCRIPTION

The provided dataset contains posts from a social media platform "Twitter". This platform is often used by content providers to publish the links to their websites. Each post, "tweet", is a short message (up to 140 characters), which can be accompanied by a link and a picture.

This is the information collected by the organizers: the content of each tweet, the link and the picture it contains. Each post is provided as a JSON object. For example:

```
{
  "id": "608999590243741697",
  "postTimestamp":
      "Thu Jun 11 14:09:51 2015",
  "postText":
      ["Some people are such food snobs"],
  "postMedia":
      ["608999590243741697.png"],
  "targetTitle":
      "Some people are such food snobs",
  "targetDescription":
      "You'll never guess one...",
  "targetKeywords":
      "food, foodfront, food waste...",
  "targetParagraphs": [
    "What a drag it is, eating kale ...",
    "A new study, published this ...",
    "..."
  ],
  "targetCaptions": ["(Flikr/USDA)"]
}
```

The following information is available for each tweet:

[1] http://www.clickbait-challenge.org/

[2] https://github.com/alexeygrigorev/wsdmcup17-vandalism-detection

[3] https://newsroom.fb.com/news/2014/08/news-feed-fyi-click-baiting/

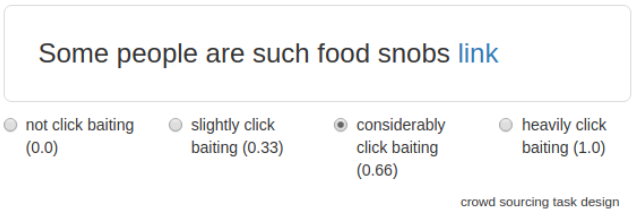

Figure 1: The design of the task evaluation. The screenshot is taken from http://clickbait-challenge.org/.

Dataset	Posts	Clickbaits	Not Clickbaits
Training	2495	762	1697
Validation	19538	4761	14777
Unlabeled	80012	n/a	n/a
Testing	?	?	?

Table 1: The datasets provided by the organizers.

- `postText`: the content of the tweet,
- `postMedia`: the image that was posted alongside with the tweet (also made available by the organizers),
- `targetTitle`: the title of the actual article,
- `targetDescription` and `targetKeywords`: the description and keywords from the meta tags of the article,
- `targetParagraphs`: the actual content of the article,
- `targetCaptions`: all captions in the article.

Each post is then assigned a "clickbaitness" score by human evaluators . It was possible to assign the post one of the following options (see fig. 1):

- not click baiting (0.0),
- slightly click baiting (0.33),
- considerably click baiting (0.66),
- heavily click baiting (1.0).

Multiple people evaluated each post, and all the responses were saved. For example, the post titled "Apples' iOS 9 'App thinning' feature will give your phone's storage a boost" was evaluated by five people and the following scores were obtained:

- 0.00 – 3 times,
- 0.33 – once,
- 0.66 – once.

Then the mean score was calculated, which is 0.2 for this post. This score indicates that the post not clickbaiting. This is the target variable the contestants should predict: the mean clickbaitness score of each post. The solutions then were evaluated by Root Mean Squared metric – a common metric for evaluating regression models.

Two labeled datasets were provided by the organizers: first, the dataset from the original work by Potthast et al [9] with 2,495 posts: 762 clickbaits and 1,697 not clickbaits. When the competition was already in progress, the second dataset was released: it contained 19,538 posts with 4,761 clickbaits and 14,777 not clickbaits [11]. The third labeled dataset was not provided: it was used for evaluating the models of the contestants.

In addition to the labeled dataset, another set of approximately 80,000 tweets was available for use. The records in this dataset followed the same format, but no ground truth labels were collected for these posts (see table 1).

Finally, the contestants were free to use any external data source.

3. APPROACH

In this section we present our approach to the challenge in details. First, we describe the hardware and software used for the solution, then talk about external datasets we used for the challenge. After that we describe the features we extracted from the challenge dataset as well as the models built on these features.

3.1 Environment

The experiments were performed on a Linux Ubuntu server with 32GB RAM and 8 Cores.

We used Python 3.4 and the PyData stack for our development:

- `numpy` 1.12.1 for numerical operations [12];
- `scipy` 0.19.0 for storing sparse data matrices [4];
- `pandas` 0.18.1 for tabular data manipulation [6];
- `nltk` 3.2.1 for text manipulation [1],
- `scikit-learn` 0.18.1 for data preprocessing and machine learning [7].

We used Anaconda – a distribution of Python with many scientific libraries pre-installed[4], including all the aforementioned libraries.

3.2 External Data

In the competition the participants were allowed to use any external data source. The provided labelled datasets are not very large, which is why we decided to obtain additional data: we believed that it should help improve the quality of the resulting models.

To obtain this additional data, we used the approach described in the post "Identifying Clickbaits Using Machine Learning" by A. Thakur[5]. Namely, we identified multiple Facebook groups that contained mostly clickbaiting posts, and used the information there as positive labeled training instances. Likewise, we picked up multiple groups with news which we deemed not clickbaiting and assigned them the negative label. To retrieve the information from Facebook we used the Facebook scraper opensourced by M. Wolf[6].

This way we obtained the content of the following Facebook groups:

- clickbaiting (88.7k posts):
  - buzzfeed (42.8k posts),
  - clickhole (14k posts),
  - upworthy (30k posts),
  - StopClickBaitOfficial (1.9k posts).
- not clickbaiting (154.9k posts):
  - CNN (65.7k posts),

[4]https://www.continuum.io/downloads
[5]https://www.linkedin.com/pulse/identifying-clickbaits-using-machine-learning-abhishek-thakur/
[6]https://github.com/minimaxir/facebook-page-post-scraper

Feature	MSE	Time, s	Best C
postText	0.039	1.7	0.1
targetKeywords	0.060	1.3	0.5
targetDescription	0.054	1.02	0.005
targetCaption	0.053	1.7	0.001
targetParagraphs	0.044	12.6	0.001
targetTitle	0.047	1.7	0.01
all concatenated	0.042	16.3	0.001

Table 2: The performance of individual `LinearSVR` models for predicting the mean.

Feature	MSE	Time, s	Best C
postText	0.014	0.7	0.01
targetKeywords	0.014	0.3	0.01
targetDescription	0.014	0.9	0.005
targetCaption	0.014	2.0	0.001
targetParagraphs	0.015	11.5	0.001
targetTitle	0.014	1.0	0.01
all concatenated	0.015	14.6	0.001

Table 3: The performance of individual `LinearSVR` models for predicting the standard deviation.

Features	MSE
Mean only	0.0331
Mean + Facebook	0.0327
Mean + std	0.0326
Mean + std + Facebook	0.0326

Table 4: The performance of the ensemble models.

  - Wikinews (2.8k posts),
  - NYTimes (86.4k posts).

In total we collected approximately 244.5k posts, and around 40% of them were clickbaiting.

A similar approach was used by the author when developing a clickbait detection model at Searchmetrics, a company that is doing Search Engine Optimization.

3.3 Modeling

Data Preparation and Features

Each post in the provided dataset contains multiple text fields and a picture. Before using the text fields in the modeling, we applied to them the following preprocessing procedure:

- remove repeating sentences,
- remove HTML tags like `<i>`, `<b>` and so on,
- remove English stop words,
- extract the stem of each word with a Porter stemmer [5],
- replace each number with a special token "`[n]`",

The same preprocessing procedure was applied to the external dataset from Facebook.

Then we used the standard Bag-of-Words approach to encode the textual information into a vector space [5]. In additional to single tokens we also used 2- and 3-grams to capture the sequential nature of the textual data.

We decided not to use image data as it seemed quite difficult and we believed it would not provide a significant improvement over the text-only solution.

Machine Learning models

Our approach to the challenge was to train multiple linear models and then combine them with a tree-based model.

For model selection and validation we used the 5-fold cross-validation scheme, and the tables below report the mean score across all 5 folds.

For each text feature we trained an SVM regression model with a linear kernel (`LinearSVR` from scikit-learn, which is based on the LIBLINEAR library [2]). These sets of models were trained to predict the mean clickbaitness score of each post. The performance of each feature is reported in the table 2. The best performing individual model is the model based on the post text – the actual message written on twitter. This model achieves 0.039 MSE on cross-validation.

Since our model is linear, it is possible to look at the coefficients to discover what are the strongest indicators of clickbaitness. For example, the following are the top 10 features with the largest positive weight (note that the tokens are stemmed):

- `[n] pictur`
- `[n] thing`
- `[n] artist`
- `[n] way`
- `[n] celebr`
- `here come`
- `shocker`
- `whoa`
- `[n] meme`
- `wat`

This suggests that the numbers (the "`[n]`" token) in the post is a very good sign that the post is clickbaiting.

In addition to that we trained another set of models which were used to predict the standard deviation of clickbaitness for each post (see table 3). For this target all features performed similarly achieving around 0.014 MSE.

Lastly, we trained a binary classification model on our external data. For that we used SVM with a linear kernel, `LinearSVC` from scikit-learn, based on the LIBLINEAR library [2]. To evaluate the performance there we used a hold-out dataset, and obtained AUC of 95%. The trained model was then applied to the textual fields of the competition dataset, and the output was used an a set of additional features.

Ensembling

After training individual models we combined them using stacking [13]: the output of these models was used as input to another second-level model. The individual linear models were stacked using Extremely Randomized Trees [3] (`ExtraTreeRegressor` in scikit-learn) – a variation of the Random Forest model that works especially good for stacking because it rarely overfits.

We trained the ensemble on different feature subsets, and established that the model that uses only mean and standard deviation features was the best one. The model that also used the extra data from Facebook achieved the same performance, which is why we

Feature	Importance
`postText` mean	0.32
all concatenated mean	0.29
`targetTitle` mean	0.15
`postText` std	0.08
`targetKeywords` mean	0.07
`targetTitle` std	0.02
all concatenated std	0.02
`targetKeywords` std	0.01

Table 5: Feature importance: fraction of times the feature was used for a split in the tree.

Team name	MSE	Running Time
zingel	0.0332	00:03:27
emperor	0.0359	00:04:03
carpetshark	0.0362	00:08:05
arowana	0.0390	00:35:24
pineapplefish	0.0413	00:54:28

Table 6: Top 5 participants of the challenge.

decided to use the simpler model without the Facebook features (see table 4).

According to the output of the ExtraTree model, the most important feature is the `postText` model for predicting the mean, which is not surprising, since it is best performing single model (see table 5).

4. EVALUATION RESULTS

For the final evaluation we selected the best performing mean+std model and executed it against the withheld testing data on the TIRA platform.

Our approach took the 3rd position with 0.0362 MSE on the testing dataset (see table 6).

5. CONCLUSION

In this paper we approach the problem of identifying clickbaiting posts on social media. We show how to address the challenge with a small ensemble of linear models, and we conclude that the results are competitive: our model ranked high on the final standing.

References

[1] S. Bird. Nltk: the natural language toolkit. In *Proceedings of the COLING/ACL on Interactive presentation sessions*, pages 69–72. Association for Computational Linguistics, 2006.

[2] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[3] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[4] E. Jones, T. Oliphant, P. Peterson, et al. Scipy: Open source scientific tools for python, 2001. *URL http://www. scipy. org*, 73:86, 2015.

[5] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

[6] W. McKinney. Pandas, python data analysis library. 2015. *Reference Source*, 2014.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[8] M. Potthast, T. Gollub, F. Rangel, P. Rosso, E. Stamatatos, and B. Stein. Improving the Reproducibility of PAN's Shared Tasks: Plagiarism Detection, Author Identification, and Author Profiling. In *CLEF*, pages 268–299. Springer, 2014.

[9] M. Potthast, S. Köpsel, B. Stein, and M. Hagen. Clickbait Detection. In N. Ferro, F. Crestani, M.-F. Moens, J. Mothe, F. Silvestri, G. Di Nunzio, C. Hauff, and G. Silvello, editors, *Advances in Information Retrieval. 38th European Conference on IR Research (ECIR 16)*, volume 9626 of *Lecture Notes in Computer Science*, pages 810–817, Berlin Heidelberg New York, Mar. 2016. Springer. .

[10] M. Potthast, T. Gollub, M. Hagen, and B. Stein. The Clickbait Challenge 2017: Towards a Regression Model for Clickbait Strength. In *Proceddings of the Clickbait Chhallenge*, 2017.

[11] M. Potthast, T. Gollub, K. Komlossy, S. Schuster, M. Wiegmann, E. Garces, M. Hagen, and B. Stein. Crowdsourcing a Large Corpus of Clickbait on Twitter. In *(to appear)*, 2017.

[12] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[13] D. H. Wolpert. Stacked generalization. *Neural networks*, 5(2): 241–259, 1992.