CLICKBAIT DETECTION

Riya Maan
4719913
r.maan@student.tudelft.nl

Aishwarya Shastry
4743016
a.shastry@student.tudelft.nl

Jos Smalbil
1362933
p.j.smalbil@student.tudelft.nl

ABSTRACT

UPDATED—June 14, 2019. The purpose of a clickbait is to make a link so appealing that people click on it. However, often at the end the reader leaves 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 partly reproduce the 3rd ranked solution from Clickbait Challenge 2017 [8]. We use a similar ensemble of linear SVM model and the dataset used in the Clickbait Challenge. The major difference in our approach is we use feature extraction and selection for our machine learning model. The Python code of our project is available on GitHub [2].

INTRODUCTION

The increasing use of social media in today's society for online news reporting, politics and public relations has given rise to misuse of the internet by few. People publish fake news, hate speech and clickbait in a hunt for clicks, to grab users' attention. Clickbait is an effective method to increase page visits as it offers enough information to entice curiosity in users, but not sufficient information to satisfy their curiosity without clicking on the linked content [22]. It exploits the "curiosity gap" of users. If the tweet withholds information required to understand what the content of the article is and it exaggerates the article to create misleading expectations for the user, then it is a clickbait [17]. The content of clickbait articles is often not related to the title, shows a poor quality, and at the end leaves the reader unsatisfied [8]. Particularly on Twitter, there has been an increasing trend to use clickbait headlines to drive the readers to click on links mentioned in the tweets. As mentioned in [24], the top twenty most prolific publishers on Twitter used clickbait on regular basis and the percentage of publishing clickbait tweets reached to an astounding 26% among all the tweets.

Clickbait tweets do not reveal important information and exaggerate the actual information. Some of the examples of clickbait are:

- Why most real estate agents never get rich?
- Find out if your home in the Orlando area is increasing in value. Learn what you could sell for today.
- Man tries to hug a wild lion. You won't believe what happens next!
- 21 stars who ruined their face due to plastic surgery. Talk about regrets!
- Only the people with an IQ above 160 can solve these questions. Are you one of them? Click to find out...

Clickbaits mention unbelievable results, mysterious stories, challenge your IQ and use tricky stuff to lure users into clicking the post. This has become a major issue and needs to be solved.

In order to save time of readers it is possible to detect if a title is a clickbait and warn them or even hide the link altogether. To detect the "clickbaitness" of articles, in the clickbait challenge participants were asked to build a classifier to score clickbaitness of a post. The classifiers give the clickbait score in the range [0,1] where 1 indicated high clickbaitness. Their performance was measured against a crowdsourced test set. The posts in the training and test set were judged on a 4 point scale [0, 0.3, 0.66, 1] by at least five annotators.

In this paper we propose to solve the problem of detecting clickbaitness of social media posts using a machine learning model, specifically we apply a linear SVM classification model based on a set of linguistic features from our training data (Figure 1).

RELATED WORK

This section provides a brief overview of recent work on Clickbait detection.

In the past few years, there has been a lot of study on Clickbait detection. Bram Vijgen [21] first studied articles that have lists often referred as listicles. Listicles are often suspected as clickbaits due to their titles which are often shared by teaser messages. He collected a total of 720 articles by Buzzfeed in Jan 2014. He found that these titles contained a cardinal number same as the number of items listed. These titles also contained nouns and adjectives that conveyed authority.

Facebook attempts to kill clickbait as announced by El-Arini and Tang [5]. Little is known about Facebook's clickbait filtering approach. The announcement mentions only that context features such as dwell time on the linked page and the ratio of clicks to likes are taken into account to detect and stop clickbait.

Gianotto [7] implements a browser plugin to detect clickbait

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

[2] https://github.com/riya-maan/Clickbait-Detection

CHI'16, May 07–12, 2016, San Jose, CA, USA

ISBN 123-4567-24-567/08/06…$15.00

DOI: http://dx.doi.org/10.475/123_4

on the rule set. The plugin is open source and one could check the source code of the plugin to confirm it's authenticity.

Potthast et al. [13] first used machine learning approach to detect clickbaits. They collected tweets and used crowdsourcing to solve the problem. Chakraborty et al. [3] used an ML classifier for automatic clickbait detection and created a browser extension STOP CLICKBAIT to block clickbaits. Their model achieved an accuracy of 0.93 for detecting clickbaits and 0.89 for blocking them.

Chen et al. [23] suggested a hybrid approach for automatic clickbait detection. Their approach used lexical, semantic and syntactic analysis as well as image and user behaviour analysis which merged both the methods to give better results.

Agrawal A. [1], in his work, uses a convolutional neural network to solve the clickbait problem. He creates a corpus using various social media platforms and utilizes deep learning for learning features rather than undergoing the tedious process of feature engineering. They received an accuracy of 0.90, a precision of 0.85 and a recall of 0.88 on the clickbait class.

Elyashar, A. et al. [6] uses classification approach to distinguish between a clickbait and legitimate posts. Their classifier is based on a variety of features: image related features, linguistic analysis and methods for abuser detection. The best performance obtained by their ML classifier was an AUC of 0.8, an accuracy of 0.812, precision of 0.819 and recall of 0.966. Thomas, P. [16] uses a fusion of neural networks. Their model does not require any linguistic preprocessing and achieved an accuracy of 0.826, and an F1 score of 0.564 and mean squared error of 0.0428. Zhou, Y. [24] uses a self-attentive neural network to solve Clickbait challenge. They reformatted the regression problem into a multi-classification problem and performed token-level, self-attentive mechanism on the hidden states of bi-directional gated recurrent units resulting in a mean square error of 0.033. Their model was ranked first in the Clickbait Challenge 2017.

Cao, X., Le, T. and Zhang, J. [2] in their approach filter out features to avoid over-fitting and improve running time of learning and selected 60 most important features for the final model. They achieved a Mean squared error of 0.035, an accuracy of 0.82, and F1-score of 0.61 using random forest regression.

Biyani et al. [19] use a machine learning model to detect clickbaits. They use various features and show that degree of informality is a strong indicator for clickbait for webpages. Their model achieves high performance (0.749 F-1 score) in predicting clickbaits.

Ferro et al. [15] and Wiegmann et al. [14] utilize features which are grouped into 4 categories. They show that scaling up feature selection efforts to heuristically identify better-performing feature subsets improves the performance of the baseline classifier. Their work improves the baseline performance by 20 percent showing that that traditional classification approaches can still keep up with deep learning on this task.

A good feature selection is important as the features determine the information that can be used by learning models. Grigorev [8] used linear SVM to train each feature. These models were trained to predict the mean clickbaitness score for each tweet.

APPROACH

Our approach in clickbait detection is basically applying a machine learning model. First, we pre-process the tweet data and then we extract the features. After applying 5-fold cross-validation we split our data into a training and a test set. We then train a linear SVM regression model after applying feature selection with PCA. The resulting classifier can predict whether a post is clickbait or not. The overview of our approach is shown in Figure 1. For this project, we are focusing more on language-based features. Our system is trained on Clickbait 2017 training corpus. The training set contains 19,538 tweets (4761 are clickbait and 14777 are no-clickbait posts) each annotated by a minimum of 5 people recruited at Amazon's Mechanical Turk [13]. 18,979 tweets annotated in the same manner are used for testing. Our training set contains 19538 posts.

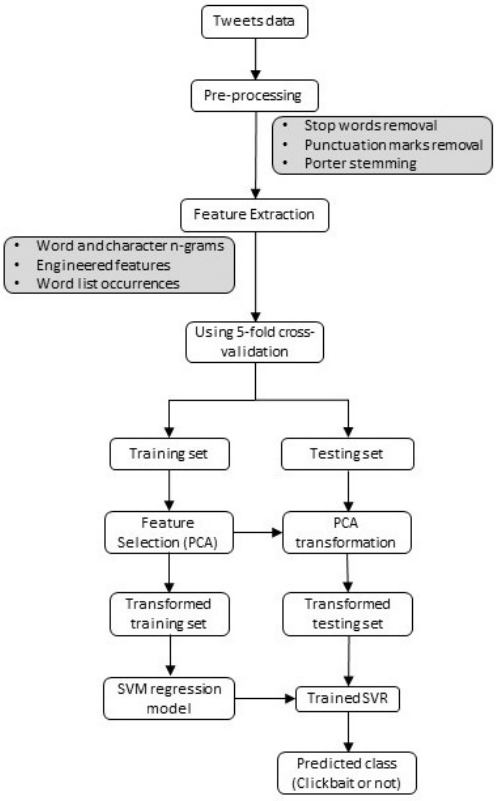

Figure 1. Flowchart of algorithm for clickbait detection

Preprocessing

We begin by removing stopwords from our obtained tweet text. We used stopwords from the NLTK package and filtered our tweets to obtain tweets without English stopwords. Next, we performed Porter stemming to obtain stems of words [18]. After obtaining the new text with stems, we further preprocessed it by removing punctuation marks. This obtained text is further used for feature extraction.

Table 1. Comparison of Results

Method	Accuracy	MSE	F1 Score	Precision	Recall
Agarwal[1]	0.90	-	-	0.85	0.88
Elyashar[6]	0.812	-	-	0.819	0.966
Thomas[16]	0.826	0.0428	0.564	-	-
Zhou[24]	-	0.033	-	-	-
Cao[2]	0.82	0.035	0.61	-	-
Biyani[19]	-	-	0.749	-	-

Feature Extraction

The next step is to obtain features from our preprocessed tweet text. Features mentioned in the paper 'Heuristic Feature Selection for Clickbait Detection' [14] are extracted:

1. *Tweet character n-grams*- We extracted 1, 2 and 3 character n-grams from tweets. Only the n-grams which occurred more than twice were taken into further consideration and weighted by tf-idf; resulting in 18,285 character n-gram features.

2. *Tweet word n-grams*- We extracted 1, 2 and 3 word n-grams from tweets. Word n-gram features occurring more than twice are weighted by tf-idf. This gave us 12,617 features in total.

3. *Engineered features*- We extracted a total of 12 engineered features.

   - Character count features: We calculated three character count features, that is, average word length, length of the longest word and total character length.
   - Character occurrence features: Five character occurrence features are calculated: occurrence frequency of @ (mentions), # (hashtags) and. (dots), whether a tweet starts with a number and the occurrence frequency of abbreviations following the Oxford abbreviations list.
   - Metadata: The metadata of the tweets are also used to indicate if tweet contains media attachments and part of the day.
   - Sentiment polarity: Sentiment polarity of the tweet is determined by VADER sentiment detector. In order to calculate the VADER sentiment, we used the Natural Language Toolkit (NLTK) for Python with the VADER sentiment analysis tool and the VADER lexicon as also done in [9]. VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis Python library. To calculate the sentiment of a tweet it uses both qualitative and quantitative methods, the VADER lexicon is used as an empirically validated *gold-standard* list of lexical-features and these features are combined with some general rules that use grammatical and syntactical conventions for sentiment intensity.
   - Readability score: Tweets' Flesh-Kincaid readability score is calculated. The Flesch-Kincaid Grade Level score can be seen as the number of years of education generally required to understand this text. The grade level is calculated with the following formula:

     $$0.39\left(\frac{\text{total words}}{\text{total sentences}}\right)+11.8\left(\frac{\text{total syllables}}{\text{total words}}\right)-15.59$$

     The lowest grade level score for a text, in theory, is -3.40. In order to obtain the pronunciation and tokenization of texts we again used the NLTK Python package.

4. *Word Lists*- We used following word lists:

   - General Inquirer word lists- The General Inquirer wordlist is freely available on the web and contains 1915 positive words and negative 2291 negative words.
   - Terrier stop word list- Terrier stop word list is retrieved from Terrier Search Engine. This list contains 732 stopwords.
   - Dale-Chall easy words list- The Dale- Chall words contains 2950 easy words. We check our text for these words. These words are understood by more than 80% of 4th graders.
   - Downworthy common clickbait phrases- This list contains few common phrases that have been identified as a clickbait. A list of 73 phrases has been used here.

   The indication of how often any word from these lists occur as a tweet is used as a single feature for each word list.

After feature extraction, we obtained a total of 30,918 distinct features.
All the features collectively give a feature matrix with more than thirty thousand features for each tweet. We cannot use the whole list of features because of the limited dataset. So, we propose to use PCA for feature selection from given feature set.

Principal Component Analysis

Principal Component Analysis (PCA) converts set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. These set of uncorrelated variables are called principal components. Using PCA, we can identify our most dimensionalities and use them to explain the variance in data. It is mostly used for dimensionality reduction. Our model uses PCA with the value of components as 0.99 to preserve 99% variance.

After feature selection, we will get a set of features that we can use for further procedures. We intend to use linear SVM as in

[8] for classifying the given tweets into two classes: clickbait and no-clickbait. A linear SVM works well with huge data and a lot of features.It is faster than other kernels [11].

SVM Regression model

In SVM regression model, input is mapped to n-dimensional feature space using nonlinear mapping. A linear model is then constructed in the feature space. The linear model could be described as

$$f(x,w) = \sum_{i=1}^{n} \omega_i \times g_i(x) + b \quad (1)$$

The SVM regression performs linear regression in high dimensional space using εwhich is the insensitive loss function. We could obtain different εas mentioned in Table 2.

SVM Kernels

Kernels can be defined as a way of computing dot product of two vectors. Kernels operate in high dimensional feature space without calculating the coordinates of the data in the space and instead use dot product of the two vectors. This operation being computationally inexpensive is popularly known as Kernel trick[10].
We use the below three types of Kernels in our experiment.

Linear Kernel

In a Linear Kernel, we calculate the inner product of two vectors <x,y> and a constant c. A Linear kernel function could be given by:

$$K(X,Y) = X^T Y + C \quad (2)$$

A linear kernel finds the largest possible margin between regions and separates them. A linear kernel performs better for text classification and with data with lots of features[11].

RBF Kernel

The Radial Basis Function (RBF) kernel is commonly used in SVM classification [4]. The RBF kernel on two samples X and Y, represented as feature vectors in some input space, is defined as:

$$K(X,Y) = \exp\left(\frac{||X-Y||^2}{2\sigma^2}\right)$$

[20]. In this equation $||X-Y||^2$ can be seen as the squared Euclidian distance between the two feature vectors. The RBF kernel can be interpreted as similarity measure since the value of the RBF kernel decreases with distance and ranges between zero and one.

Sigmoid Kernel

The Sigmoid kernel is also known as the Hyperbolic Tangent kernel or the Multilayer Perceptron (MLP) kernel. The Sigmoid kernel comes from the neural network field and it is interesting to note that a SVM model using a Sigmoid kernel function is equivalent to a two-layer, perceptron neural network. The kernel is defined as:

$$K(X,Y) = \tanh(\gamma X^T Y + r)$$

There are two adjustable parameters in the sigmoid kernel, the slope γ and the intercept constant r. A common value for γ is $1/N$, where N is the data dimension. More details on Sigmoid kernels can be found in a study by Lin et al. [12].

Kernel	ε	Error rate
Linear	0.5	0.68
	0.1	0.3
	0.01	0.3
RBF	0.5	0.69
	0.1	0.37
	0.01	0.377
Sigmoid	0.5	0.69
	0.1	0.51
	0.01	0.49

Table 2. Error rate for different kernels for different values of epsilon

RESULTS AND DISCUSSION

We worked on Clickbait Detection Challenge 2017 dataset. They provided one dataset with 19538 tweets and one with 2459 tweets data. We tried to run the project on 8GB RAM system with Intel Core i7 processor, 64-bit operating system. Due to system constraints, we were not able to work with the larger dataset. Hence, we used the smaller dataset to execute our project.
On using the smaller dataset, we got a set of 14525 features. But we just had 2459 tweets, so, either we could have increased dataset or selected few features. We used the latter alternative and selected features using PCA so as to preserve 99% variance. After performing feature selection, we trained SVM regression model. Then, we transformed testing set using PCA and classified using trained SVR. The results are averaged over 5-fold cross-validation to remove any bias.
We found out that linear kernel works best for this dataset, with epsilon value of 0.1 and 0.01, as can be seen in table 2. However, if we increase the epsilon value to 0.5, error rate increases. For other two kernels, that is RBF and sigmoid, similar trend is observed. Epsilon value of 0.5 results in worst performance.
This is in line with the methodology used by Alexey Grigorev in [8]. He used linear SVM regression model as a classifier. His approach was ranked third among all approaches submitted to the challenge. Our reproduction of their experiment lead to similar findings. However, we did not use tree based SVM model. This was because we did not divide our features in different categories and used mostly language based features that depend on tweets' text. This was done to make the project more in line with natural language processing.
Even after changing the feature set, linear kernel worked best among other kernels. Hence, we can say that we effectively reproduced Alexey's paper [8] to get similar results with different set of features.

CONCLUSION AND FUTURE WORK

In this project, we evaluated tweets whether they are clickbait or not. We developed a system that pre-processes tweets by removing stop-words, punctuation marks and porter stemming. After this, features are extracted. We used four kinds of features: word n-grams, character n-grams, engineered features and occurrence of word lists. As we had fewer tweets than the

number of features, so, we used feature selection using PCA. After features are selected to conserve 99% variance, they are passed into trained SVR system and the clickbaitness score will be evaluated. If the clickbaitness score is more than 0.5, it is predicted as clickbait, else as no-clickbait.
Results from this system were evaluated using 5-fold cross-validation and different kernels and epsilon values. We found out that for our set of features, the linear kernel performs best among other kernels for the value of epsilon 0.1 and 0.01. These results were found to be in line with the kernel used by Grigorev in [8].
Usage of clickbait articles has seen a rapid increase in the past few years. To prevent internet users from these, we need effective systems can be used to identify clickbait. We can use make systems in the form of browser extensions, that can make users aware of clickbait. To increase the efficiency of these systems, more language-based and media-based features need to be explored, that can identify clickbait.

REFERENCES

1. Amol Agrawal. 2016. Clickbait detection using deep learning. (2016).
2. Le T. Cao, X. and J. Zhang. 2017. Machine Learning Based Detection of Clickbait Posts in Social Media. (2017).
3. Paranjape B. Kakarla-S. Chakraborty, A. and N. Ganguly. 2016. Stop Clickbait: Detecting and preventing clickbaits in online news media. *IEEE/ACM International Conference on Advances in Social Networks* (2016).
4. Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. 2010. Training and Testing Low-degree Polynomial Data Mappings via Linear SVM. 11 (04 2010), 1471–1490.
5. Khalid El-Arini and Joyce Tang. 2014. News Feed FYI: Click-baiting. (2014). http://newsroom.fb.com/news/2014/08/news-feed-fyi-click-baiting.
6. Bendahan J. Elyashar, A. and R Puzis. 2017. Detecting Clickbait in Online Social Media: You Wont Believe How We Did it. (2017).
7. Alison Gianotto. 2014. DownworthyâĂŤA Browser Plugin to Turn Hyperbolic Viral Headlines Into What They Really Mean. (2014). http://downworthy.snipe.ne.
8. Alexey Grigorev. 2017. Identifying Clickbait Posts on Social Media with an Ensemble of Linear Models. (2017).
9. C.J. Hutto and Eric Gilbert. 2015. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. (01 2015).
10. Harish Kandan. 2017. Understanding the Kernel trick. (2017). https://towardsdatascience.com/understanding-the-kernel-trick-e0bc6112ef78
11. Alexandre KOWALCZYK. 2014. Linear Kernel: Why is it recommended for text classification? (10 2014). https://www.svm-tutorial.com/2014/10/svm-linear-kernel-good-text-classification/
12. Hsuan-Tien Lin and Chih-Jen Lin. 2003. A Study on Sigmoid Kernels for SVM and the Training of non-PSD Kernels by SMO-type Methods. (06 2003).
13. K. Komlossy S. Schuster M. Wiegmann E. Garces M. Hagen .M. Potthast, T. Gollub and B. Stein. 2017. Crowdsourcing a Large Corpus of Clickbait on Twitter. (2017).
14. B. Stein M. Hagen M. Potthast M. Wiegmann, M. VÃűlske. 2017. Heuristic Feature Selection for Clickbait Detection. *The Icarfish Clickbait Detector at the Clickbait Challenge 2017* (2017).
15. M.-F. Moens J. Mothe F. Silvestri J. Kekalainen P. Rosso P. Clough G. Pasi C. Lioma S. Mizzaro G. M. Di Nunzio C. Hauff O. Alonso P. Serdyukov N. Ferro, F. Crestani and G. Silvello. 2016. Report on ECIR 2016: 38th European Conference on Information Retrieval. *SIGIR Forum* 50, 1 (2016), 12–27. http://doi.acm.org/10.1145/2964797.2964801
16. Thomas P. 2017. Clickbait Identication using Neural Networks. (2017).
17. Alex Peysakhovich and Kristin Hendrix. 2016. News Feed FYI: Further Reducing Clickbait in Feed. (4 August 2016). https://newsroom.fb.com/news/2016/08/news-feed-fyi-further-reducing-clickbait-in-feed/.
18. Porter. 2018. Porter Stemming. (2018). http://www.nltk.org/_modules/nltk/stem/porter.html/
19. Kostas Tsioutsiouliklis Prakhar Biyani and John Blackmer. 2016. 8 amazing secrets for getting more clicks": detecting clickbaits in news streams using article informality. *AAAI'16 Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (2016), 94–100.
20. J.P. Vert, Koji Tsuda, and B SchÃűlkopf. 2004. A Primer on Kernel Methods. (01 2004).
21. Bram Vijgen. 2014. The Listicle: An Exploring Research on an Interesting Shareable New Media Phenomenon. (2014).
22. Wikipedia. 2018. Clickbait — Wikipedia, The Free Encyclopedia. (2018). https://en.wikipedia.org/wiki/Clickbait.
23. N. J. Conroy Y. Chen and V. L. Rubin. 2015. Misleading online content: Recognizing clickbait as false news. *In Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection, pages 15âĂŞ19. ACM, 2015.* (2015), 15–19.
24. Yiwei Zhou. 2017. Clickbait Detection in Tweets Using Self-attentive Network. *arXiv preprint arXiv:1710.05364* (2017).