IN4325: Core IR Report

Karel van Doesburg (4149335)
TU Delft
Delft, The Netherlands
k.g.h.vandoesburg@student.tudelft.nl

Ruben Bes (4227492)
TU Delft
Delft, The Netherlands
r.l.bes@student.tudelft.nl

Thomas Kolenbrander (4353137)
TU Delft
Delft, The Netherlands
t.j.kolenbrander@student.tudelft.nl

ABSTRACT

Many posts on social media use clickbait titles to trick users into viewing an article. This paper is an attempt to reproduce the paper *Detecting Clickbait in Online Social Media: You Won't Believe How We Did It* by Elyashar et al.[2] which aims to classify Twitter posts as clickbait or normal. Using XGBoost, we show that (insert results) and that the features that are best for predicting clickbait posts are (insert features).

INTRODUCTION

According to Potthast et al., clickbait messages are messages in a social stream that are designed to exploit cognitive biases to increase the likelihood of readers clicking an accompanying link[4]. While clickbait articles are not necessarily harmful, the main issue with this type of content is that it conveys little (factual) information and is mostly aimed at serving advertisements to users, thus can be seen as a type of noise in online social media. The clickbait challenge[1] invites researchers to participate in the ongoing challenge to develop a classifier that rates how click baiting a social media post is.

This paper is an attempt at reproducing the efforts of Elyashar, Bendahan and Puzis in the 2017 clickbait challenge. They test a number of classifiers on a set of features extracted from the official 2017 clickbait challenge dataset[2].

This paper is structered as follows, we will first give an overview of the background information on clickbait and algorithms used in section 2. Next, in section 3 we will discuss our approach to reproducing the paper by Elyashar et al. After this, we outline our experiments and discuss our results in section 4, and finally we finish with a conclusion on this research in section 5.

BACKGROUND

In 2017 the clickbait challenge [3] was organized in order to encourage research in the field of clickbait recognition. The paper we try to reproduce also participated in this challenge but they did not manage to outperform the baseline implemented by the organizers of the challenge. The best-performing approach was based on a neural network created by Zhou et al. [6].

XGBoost

The original paper uses four machine learning methods to classify the clickbait posts: XGBoost, ADABoost, Random Forest and Decision Tree [3]. All of these classifiers are a kind of decision tree classifier. The classifier that scored the best in almost all of the metrics is the XGBoost classifier. This is the classifier that we will also use for our reproduction paper. The XGBooster classifier is recognized as one of the state-of-the art machine learning classifiers [1]. An example, which shows the popularity of the classifier, is the machine learning competition site Kaggle. In this competition 17 out of 29 winning machine learning approaches used XGBoost. XGBoost is based on the more general method of gradient tree boosting, which is a method that generalizes many different (weaker) decision trees into a strong classifier. The strength behind the XGBoost classifier is its scalabitlity. The XGBoost outperforms existing solutions by running ten times as fast. XGBoost also allows use of multiple processors, which is perfectly suitable for the laptops on which we will test our data.

APPROACH

Packages and version numbers

This sections lists the software versions we use, to make replication/verification easier.

- Python 3.6
- XGBoost 0.82
- info-gain 1.01
- NLTK 3.4, WordNet 3.0 corpus

Preprocessing

Our data consists of about 19,500 JSON objects, each representing a tweet linking an article. Each tweet has a number of fields, which can be found in table 1. The original authors make no mention of any filtering steps, but we noticed a number of odd characters, such as hexadecimal, unicode and emoji characters, which we expect to have influence on the results. We therefor filter the textual data using a regular expression to replace all characters that are a hexadecimal non-breaking space(\xa0), Unicode or non-alphabetic by a regular space.

[1] https://www.clickbait-challenge.org/
[2] https://www.clickbait-challenge.org/#data

ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Field	Type	Information
postMedia	Textual Link	Relative link to image(s) in post
postText	Text	Text used in post
id	Integer	Identifier for post
targetCaptions	Text	List of image captions in article
targetParagraphs	Text	List of paragraphs in article
targetTitle	Text	Title of article
postTimestamp	Integer	Timestamp of post
targetKeywords	Text	Keywords of article
targetDescription	Text	Description of article

Table 1. Fields in the JSON objects

We also remove leading and trailing spaces and changed all letters to lowercase.

We aim to investigate the effects of stemming as well, but have not implemented this yet at the time of writing this intermediate report.

Feature extraction

In order to use the data, we first need to extract the features from the dataset. We have categorized the features into three categories: Basic, linguistic and image-related. The image-related features have not (yet) been implemented due to the focus on language processing in this course. With the implementation of these features we had to make some assumptions, which will be discussed below:

- Words: The function *words(p)* returns a set of the words in a given content *p*. The authors do not explain their definition of a word and how exactly the words are extracted from the content. Our approach consists of first removing all punctuation and numbers from the text and then splitting the text on white-spaces in order to create individual words. Finally, we replace all uppercase letters with their lowercase counterparts for an easy comparison between words. Here we assume that words are equal despite having differences in upper/lowercase characters.

- Formal dictionary: To determine whether a word is a formal or not, a dictionary is required. The original authors make use of the PyDictionary[3] library. We choose not to do so as the library does not support a local corpus and instead sends a request to the Princeton server for every word. This slowed down our process and we therefore choose to use the Natural Language Toolkit (NLTK)[4] which does support a local corpus. The difference between these two is that PyDictionary makes use of WordNet 3.1 where NLTK uses WordNet 3.0.

A full list of features can be found in appendix A.

We are considering adding the following features:...

Machine learning

As mentioned before we make use of the XGBoost classifier to classify our posts as clickbait or as non-clickbair [1]. To be able to make suboptimal use of the XGBoost classifier we must optimize some of its parameters. Because the scope of this paper is to reproduce part of the original paper where we preprocess the data and analyze the linguistic analysis feature extraction, we do not aim to optimize the classifier to its fullest. The most important parameters to optimize were the learning rate (eta), gamma, maximum tree depth and the minimum child weight [5], extra features would lead to a significant increase in runtime to find the optimal parameters. These resulted in the following parameters displayed in the format [learning rate, gamma, maximum tree depth, minimum child weight]:

- Accuracy: [0.6, 0.1, 3, 3]
- Precision: [0.4, 0.9, 5, 2]
- Recall: [1.7, 0.1, 9, 1]
- AUC: [1.7, 0.1, 9, 1]

Furthermore we have set the scale_pos_weight attribute to 1 and colsample_bytree and subsample attribute to 0.8.

EXPERIMENTS

Information gain

To analyze the importance of features we will calculate the information gain of the features like in the original paper. The information gain gives us a value which tells us in which degree a feature is able to reduce the disorder in the original dataset when the dataset is split into different classes using this feature as classifier [5]. Or in other words how useful is this feature in classifying the objects in a dataset. Information gain is based on entropy, or more on how much the feature reduces the entropy. To calculate the information gain first the entropy of the classes have to calculated using the following formula:

$$E(x) = -\sum p(x)\log_2 p(x) \tag{1}$$

Here $p(x)$ is the probability distribution of a class. The gain can be found in table 2.

Machine learning/XGBoost results

CONCLUSIONS

From the results in the previous section we can conclude that the ratio of characters between two pieces of content are most effective in telling clickbait posts from non-clickbait posts. After this list of ten features, we first see the percentage of informal words and then the ratio of words in content. Interestingly, these findings do not align with the original authors' results, and our machine learning results are different as well. As we have no idea how Elyashar et al. preprocessed the data, we assume this is where the differences arise.

Repository

Our repository is available at https://github.com/ruteben/IN4325-Applied-NLP.

[3] https://pypi.org/project/PyDictionary/

[4] https://www.nltk.org/

[5] https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

Feature	Gain
Ratio of characters between targetParagraphs and targetDescription	0.379
Ratio of characters between targetParagraphs and targetCaptions	0.307
Ratio of characters between targetParagraphs and targetTitle	0.295
Ratio of characters between targetParagraphs and postText	0.293
Ratio of characters between targetCaptions and targetDescription	0.275
Ratio of characters between postText and targetDescription	0.265
Percentage of formal words in targetParagraphs	0.259
Ratio of characters between targetTitle and targetKeywords	0.255
Ratio of characters between targetParagraphs and targetKeywords	0.243
Ratio of characters between targetDescription and targetKeywords	0.231

Table 2. Caption

REFERENCES

1. Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794. DOI: http://dx.doi.org/10.1145/2939672.2939785
2. Aviad Elyashar, Jorge Bendahan, and Rami Puzis. 2017. Detecting Clickbait in Online Social Media: You Won't Believe How We Did It. *arXiv preprint arXiv:1710.06699* (2017).
3. Martin Potthast, Tim Gollub, Matthias Hagen, and Benno Stein. 2018. The clickbait challenge 2017: towards a regression model for clickbait strength. *arXiv preprint arXiv:1812.10847* (2018).
4. Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. 2016. Clickbait detection. In *European Conference on Information Retrieval*. Springer, 810–817.
5. J. R. Quinlan. 1986. Induction of decision trees. *Machine Learning* 1, 1 (01 Mar 1986), 81–106. DOI: http://dx.doi.org/10.1007/BF00116251
6. Yiwei Zhou. 2017. Clickbait detection in tweets using self-attentive network. *arXiv preprint arXiv:1710.05364* (2017).

APPENDIX

FULL LIST OF FEATURES

Table 3 contains the full list of 101 features currently in use.

Type	Feature	#Features	Extracted from
Basic	Number of characters	6	Each text field
Basic	Number of words	6	Each text field
Linguistic	Difference number of characters	15	Each text field vs each other text field
Linguistic	Ratio number of characters	15	Each text field vs each other text field
Linguistic	Difference number of words	15	Each text field vs each other text field
Linguistic	Ratio number of words	15	Each text field vs each other text field
Linguistic	Words in common with keywords	5	Each text field except keywords
Linguistic	Number of formal words	6	Each text field
Linguistic	Number of informal words	6	Each text field
Linguistic	Percentage of formal words	6	Each text field
Linguistic	Percentage of informal words	6	Each text field

Table 3. Caption