Clickbait Detection
Applied NLP Project - Group 11

Stavrangelos Gamvrinos
Delft University of Technology
Delft, South Holland
s.gamvrinos@student.tudelft.nl

Kyriakos Psarakis
Delft University of Technology
Delft, South Holland
k.psarakis@student.tudelft.nl

Panagiotis Soilis
Delft University of Technology
Delft, South Holland
p.soilis@student.tudelft.nl

ABSTRACT

Nowadays, a very common phenomenon that can be observed in online social media is "clickbaiting". This term refers to the use of catchy phrases and psychological tricks at the social media posts and article titles, with the purpose of attracting more user traffic to the corresponding platform. This way, websites attract more readers and increase their revenues via advertising. However, these posts are often a nuisance for the readers and frequently lead to fake or low-quality content.

In this study we examine the possibility of detecting clickbait posts using Machine Learning techniques. In particular, we use a variety of different features based on linguistic analysis, n-grams, sentiment analysis and others to classify social media posts and article combinations as clickbait or non-clickbait. Moreover, we evaluate the performance of four classifiers that have been found to perform well in previous Clickbait detection studies. Our experiments showed **...TODO CONCLUDE**

KEYWORDS

Clickbait, Classification, Automated deception detection.

1 INTRODUCTION

The past decade has seen the rise of online news media. According to [1], this emerging form of media is characterized by two differences compared to traditional ones. To be more specific, the Web offers a vast amount of options to the users, thus breaking down the trend that people are loyal to one media source. Moreover, the main revenue source of online media is via their website advertisements. As a result, there is enormous competition to grab user attention and generate website clicks. The former is especially important given that ad spending for digital advertisements is expected to overtake that of traditional ones in the USA for the first time in 2019.[1]

The aforementioned trends have led the majority of online media to employ various techniques to generate clicks. However, among those there is an unwanted trend that has emerged, that of Clickbait titles. In particular, Clickbait refers to the generation of catchy headlines that draw the user's attention to click on an article. The official definition provided by the Oxford English Dictionary[2] defines Clickbait as "content whose main purpose is to attract attention and encourage visitors to click on a link to a particular web page". Such titles can be frustrating for online users and undermine their online experiences since the target articles are usually of low quality[2]. A characteristic example of such a title is available in Figure 1.

Figure 1: Example of non-clickbait vs clickbait headline presented by Y. Chen et al.[3]

In this paper, we propose a machine learning technique that would enable automated clickbait detection on social media. Our work was particularly inspired by the approach of [4] and combines other best practices found in the literature with ideas of our own. More specifically, we propose a list of features that could potentially discriminate clickbait from non-clickbait titles. These features are then evaluated on four different machine learning classifiers. Finally, we propose future research opportunities related to the topic of Clickbait detection. In short, the following research questions are tackled:

(1) Which features separate clickbait headlines from the non-clickbait ones?
(2) Which machine learning techniques are more suited to the clickbait detection task?

[1] https://techcrunch.com/2019/02/20/emarketer-digital-ad-forecast/

[2] https://en.oxforddictionaries.com/definition/clickbait

(3) Which are possible future research opportunities regarding the topic of automatic clickbait detection?

The rest of the paper is structured as follows: Section 2 summarizes related work on the topic of clickbait detection. Section 3 details the features and classifiers that were evaluated within the scope of this paper. Finally, the results of our experiments are presented in Section 4.

2 BACKGROUND

One of the first papers in literature addressing the topic of automatic clickbait detection is [3]. More specifically, they performed a literature review on the possible textual and non-textual clickbaiting cues, and suggested possible machine learning methods that could be applied to this task. Since then, there have been several research attempts to develop machine learning classifiers that perform reasonably well on the topic.

In 2016, three approaches with common characteristics were proposed. Firstly, [2] formalized the concept of clickbait by defining eight distinct clickbait types. Moreover, they proposed a machine learning approach which uses novel informality features which proved to be a strong indicator of clickbait. Following that, [5] created the first publicly available clickbait corpus composed from Twitter tweets. In addition to that, they developed a clickbait detection model with 215 features which can be divided into three categories, namely teaser message features, linked web page features, and meta information features. The third approach proposed that year was [1] who built a classifier that detects clickbait headlines and developed a browser extension, called "Stop Clickbait", which assists users in avoiding clickbait headlines.

The following year, [6] proposed a novel approach that accounts for the user behavior, using metrics such as the Click-Through Rate (CTR), to improve clickbait classification performance. Over and above, [7] constructed the Webis Clickbait Corpus 2017 which is a new corpus of 38,517 annotated Twitter tweets. This dataset was used to evaluate the clickbait detectors submitted in the Clickbait Challenge 2017[3]. One of the latest attempts was the Ant Colony Optimization (ACO) proposal by [8]. ACO is a Swarm Intelligence (SI) based technique which uses human understandable rules allowing to interpret and modify them accordingly. This is in stark contrast to machine learning approaches which are often viewed as a black box, the internals of which cannot be properly understood.

In this paper, we aim to reproduce part of the clickbait machine learning features proposed by [4]. This was one of the twelve detectors proposed in the Clickbait Challenge 2017. In particular, the method proposed in this paper takes advantage of the most promising features described in [4] coupled with other interesting features proposed in the literature. It has to be underlined that possible deep learning approaches were not evaluated as they were not within the scope of this paper. The effectiveness of this proposal is then evaluated on four separate machine learning classifiers.

[3]https://www.clickbait-challenge.org

3 APPROACH

In this section, we detail the implementation of the proposed clickbait classification system, originally inspired by [4]. An overview of the system pipeline can be seen at Figure 2.

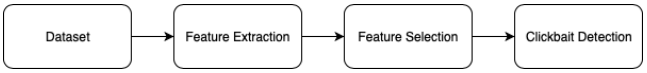

Figure 2: Clickbait Detection Pipeline

The code developed for this project is publicly available on GitHub[4]. Moreover, the versions of the software libraries used to extract the features and classify the objects are detailed in Software versions.

Dataset Overview

The data that was used to develop the system proposed were two publicly available datasets from the Clickbait Challenge[5] competition. In particular, two labeled datasets are provided that contain 2,495 and 19,538 samples respectively. The dataset structure is such that every sample consists of a unique post-article combination with multiple data fields related to each of the two components. More specifically, the post comes with a title field and an image while the article contains five fields, namely title, keywords, description, paragraphs and captions.

The proposed system takes advantage of five of those fields. In particular, the post title and the article title are used in the majority of the features since they are the main content elements that the users view before deciding whether to click on an article or not. Furthermore, the post image field was also considered since its presence could affect the user behavior. Last but not least, the article keywords and description were taken into account when designing features since they might be an indicator of the article quality associated with a specific post.

Features

The features extracted for each sample in the two datasets were divided into four categories, namely Linguistic Analysis, N-grams, Sentiment Analysis and Other Features. The details of each feature are described below:

[4]https://github.com/agamvrinos/NLP-Clickbait-detection
[5]https://www.clickbait-challenge.org

Linguistic Analysis This category of features includes semantic and syntactic analysis of text to spot patterns that separate non-clickbait from clickbait posts.

Number of characters/words: The length of the post title has been found to be a feature that separates the two classes by [1] and [4]. However, these two papers reach contradicting conclusions on whether clickbait or non-clickbait posts have larger titles. As a result, we decided to calculate the length of four content fields both in terms of number of words as well as characters. More specifically, these are the post title, article title, article description and article keywords. It has to be underlined that the content was tokenized using the space character as a delimeter in the case of the length count and that whitespaces were also included in the character count. Moreover, if the content item contains more than one elements the feature is calculated using the average number of characters/words. An example of the length based extraction is found in Table **??**. In total, eight features were extracted, four word length and four character length ones.

Difference between number of characters/words: Another feature that [4] found to separate the two types of posts is the difference in length between two of the content elements available in the dataset. Therefore, we created one feature for each of the six combinations of content elements used in the aforementioned length based features. These difference features were extracted both for the word count and the character count features, resulting in twelve features in total. It is important to note that these features were extracted both for the words and the characters individually resulting in twelve features in total.

Number of characters/words ratio: Similarly to the difference features, [4] propose the use of the ratio between the number of words/characters of two separate content elements. As a result, twelve more features were created.

Begins with interrogative: The authors in [2] propose a binary feature which checks whether the content element begins with an interrogative word[6]. To be more exact, we check whether the post title and the article title begin with one of the following: *"Who"*, *"What"*, *"When"*, *"Where"*, *"Why"* or *"How"*. Based on our intuition and personal experience, clickbait post/article titles should begin with interrogative words more often than non-clickbait ones.

Common clickbait phrases: Relatively recent studies in clickbait detection [5][8] observed that common phrases such as *"Mind-Blowing"* and *"Unbelievable"* appear very often in clickbait posts. This observation is also intuitive given that the phrases that were found to attract a lot of user traffic are being recycled over and over again. To that end, as in [5] we used a dictionary[7] of 73 common phrases taken from the Downworthy browser plugin and looked for these in the title of each individual post.

Slang words/phrases: Another phenomenon that can frequently be observed in clickbait posts is the use of slang words. In particular, words like *"OMG"*, *"LOL"* and *"LMAO"* can very commonly be seen as part of the title of clickbait posts. This was also mentioned in previously conducted studies which also focused on clickbait detection [1][8]. In our experiments, we used a publicly available corpus of slang words and abbreviations[8] and extracted the features by looking at the presence of at least one of these words. In case at least one occurrence is detected the post is tagged as clickbait, otherwise it is considered as non-clickbait.

Part-Of-Speech tags: In [1], [9] and [8] the use of Part-Of-Speech (POS) tags as features has been proven to yield good results in the clickbait detection problem. For example, these features can either be the count or the presence of proper nouns, adverbs, determiners and in general all the categories and subcategories of the parts of speech. From the related research we have observed that especially the count of these tags and more specifically the count of the proper nouns, abbreviations and pronouns are good features for our task. Consequently, this lead to our decision to take the counts of the POS tags in our implementation. Additionally, we decided to take only the 16 POS tags mentioned in [1] due to the others' bad performance in classifying posts in the clickbait setting. The aforementioned performance comparison has also been done in [9]. Finally, as shown in [9] we created the two common patterns features that indicate the presence of some of the most common clickbait title patterns. These two patterns are number + noun phrase + either verb of the word "that". For the POS tags we used the NLTK library[9].

Determiners: Clickbaits often use words such as *"their"*, *"my"*, *"when"* and *"there"* in order to refer to other individuals and pique the curiosity of the readers. As mentioned in [1], these words, also known as determiners, were found to be extremely discriminative when it comes to clickbait detection. In our study, the detection of such words is again being done by using the NLTK's POS Tagger and looking for those tagged with *"DT"* in the article title of each post.

Possessives: Many clickbait posts address the reader in a more personal way using possessive words such as *"I"*, *"You"* and *"He"*. On the other hand, non-clickbait posts are almost always expressed in the third person. This was also observed in [1] and was considerd to be essential in distinguishing

[6] https://en.wikipedia.org/wiki/Interrogative_word

[7] https://github.com/snipe/downworthy/blob/master/Source/dictionaries/original.js[*Accessed: 2/4/2019*]

[8] https://gist.github.com/Zenexer/af4dd767338d6c6ba662[*Accessed: 2/4/2019*]

[9] http://www.nltk.org/

between clickbait and non-clickbait posts. Similarly to the determiners, we looked for *"PRP"* and *"PRP$"* POS tags in each article title.

N-grams Some other features that are widely used in the literature [1] [5] [8] are the n-gram features that can either be word or character n-grams. In short that means that they are either a continuous sequence of n characters/words. Moreover, a feature representation of a sentence by n-grams is the amount of times that the sentence specific n-gram appears in comparison to the n-grams extracted from the dataset. In our implementation we chose to implement only the word n-grams due to them being more intuitive. For instance, some tri-grams can be very good indicators of clickbait content, such as "10 things that", "you cant believe" and "what happens when". Similarly, they can be indicators of non-clickbait content: "man charged with" and "national emergency as".

Our implementation generated over X n-grams, that led us to using two different thresholds by checking their frequency distribution in the entire dataset, as shown in Ngram Distributions. The first threshold was applied to remove the least influential nodes with very low appearance. The other threshold is the exact opposite meaning that if an n-gram appears a lot of times in the entire dataset it cannot be a good feature to separate the two classes. Both of the aforementioned thresholds are derived from the n-gram frequency distribution. Finally, we acknowledge that the n-gram features can be too many, leading the classifiers to fall in the curse of dimensionality. Therefore, we use the information gain feature selection method prior to the classification, as described in the following parts of this section.

Sentiment analysis

Hyperbolic features: Clickbait post titles often contain particular words that are carefully chosen such that they attract the reader's attention. A common characteristic of all these words is hyperbolism. Examples of such words are for instance *"awesome"*, *"amazing"* etc. To that end, this is a reasonable characteristic to distinguish between clickbait and non-clickbait posts. In our case we followed the same approach as mentioned in [1] in order to identify posts that contain hyperbolic words. More specifically, we used the CoreNLP tool offered by Stanford and checked for *"Very positive"* and *"Very negative"* words in the post's title. As opposed to other existing Sentiment analysis tools that use a three-category approach (negative, neutral, positive), CoreNLP includes the "Very positive" and "Very negative" classes, something that allows the detection of hyperbolism. That being said, the extracted features were based on the presence of hyperbolic words in the title of each individual post.

Sentiment Polarity features: Apart from detecting hyperbolic words, we also performed Sentiment analysis in order to identify the polarity of each post based on its title. To do so, we used the *Sentiment Intensity Analyzer* offered in VADER[10] which measures the intensity of the sentiment (also known as "compound score") in the range [-1,1]. This feature was also used in [5] but with the use of Stanford's NLP library. The reason why we selected VADER instead, was mainly because Stanford's Python wrapper does not support intensity analysis. The intensity was again extracted from each post's title.

Other Features While the linguistic and sentiment analysis features are one of the most common ones used for Natural Language Processing (NLP) problems, there are also other features that can be specifically designed for the problem of clickbait classification.

Has image: Research performed in [10] has showed that a larger percentage of clickbait posts contain images compared to legitimate ones. There, we create a binary feature has_image that simply checks whether an article is accompanied by an image.

Begins with number: An interesting feature proposed by [5] is to check whether the post title begins with a number or not. This feature intuitively makes sense since a lot of clickbait post/article titles start with "10 ways/things that ...". As a result, we chose to extract two binary features for the post and article titles.

Punctuation count: A number of papers [5][6] has already experimented with features concerning the use of punctuation within the clickbait classification task. Thus, we designed features that capture the number of exclamation marks, question marks, abbreviations, dots and ellipses for the post and article titles which we consider as the most important content elements. Moreover, since the dataset is exported from Twitter, two more features related to Twitter posts were created. More specifically, we counted the number of sings and hashtags in a post title, an idea inspired by [4].

Post creation hour: Another point worth looking into is the claim of [11] that crowdturfing campaigns are performed within a specific time interval. Therefore, we decided to encode the post creation hour with an integer value in the range 1-24 to check whether clickbait posts are posted at a specific time per day. That said, given that content websites nowadays post their articles on social media at specific hours every day, we expect non-clickbait posts to also follow a specific time pattern that may make the class separation based on this feature tricky.

[10] https://github.com/cjhutto/vaderSentiment

Feature Selection

The previous part of our implementation results into more than 10,000 features. The reason behind this is that the n-gram features for $n \in [1, 3]$ are taking most of the feature space as the non n-gram features are 72. Furthermore, just a fraction of the aforementioned n-gram features are useful for our classification while such a high number of features also leads to a dimensionality problem. Following these two observations we decided that performing feature selection is an essential part of our approach. Similarly to [4], we selected the information gain algorithm or, in Information Theory terms, mutual information that uses the Kullback-Leibler (KL) divergence to calculate the relative entropy. This method has been used in the past for feature selection by [12]. Its calculation can be summarized as follows:

$$I(X;Y) = H(X) - H(X|Y) = D(P(X,Y)||P(X)P(Y))$$

Where $I(X;Y)$ is the mutual information, $H(X)$ quantifies the amount of information that each observation of X provides and $H(X|Y)$ the amount of information that each observation of X provides when Y is known. In short, this means that the mutual information measures the reduction in uncertainty/information gained for X when Y is known. The formula $D(P(X,Y)||P(X)P(Y))$ corresponds to the entropy calculation using the KL divergence. For the mutual information we have used scikit-learn's implementation[11] for binary classification, as our problem is binary (clickbait/non-clickbait).

Clickbait Detection

The Clickbait detection task can be modeled as a machine learning binary classification problem. In particular, each post/article combination can be considered as an observation that has to be classified in one of two categories, clickbait or non-clickbait. While there is a wide range of machine learning classifiers available, we chose the following four for this task:

Naive Bayes: Naive Bayes refers to a set of simple probabilistic classifiers which are based on Bayes theorem. That means that for the classification of a test sample, the posterior probability for each class is calculated and the sample is assigned to the class with the highest probability as follows:

$$P(\omega_i|X) = \frac{P(X|\omega_i)P(\omega_i)}{P(X)}, \forall i = 1, ..., N$$

The main assumption behind Naive Bayes is that all the features are independent with each other. Although simple, Naive Bayes is widely used in the field of Natural Language Processing [3][5], especially in text classification, and it is often used as a baseline classifier. Furthermore, Naive Bayes is significantly fast and reliable while its accuracy can often be compared with those of more sophisticated and complex classifiers such as SVM. In our study, we use the Gaussian Naive Bayes of the scikit-learn library, which assumes that the likelihood of the features is Gaussian.

Max Entropy: Maximum Entropy (ME) is another classifier that is also commonly used in Natural Language Processing and text classification [13]. Unlike Naive Bayes, ME does not make the assumption that the features are conditionally independent of each other. That makes it suitable for text classification problems where words (unigrams, bigrams), which are not independent, are often used as features. More specifically, the main principle behind ME is that the correct model/distribution that fits the training data, is the one that maximizes the entropy and still adheres to the constraints set by the provided training set. The respective distribution can be calculated as:

$$P(c|d) = \frac{1}{Z(d)} \exp(\sum_i \lambda_i f_i(d,c))$$

where λ_i corresponds to parameters to be estimated, f_i refers to features and $Z(d)$ is the normalization factor:

$$Z(d) = \sum_c \exp(\sum_i \lambda_i f_i(d,c))$$

In this paper, we used the LogisticRegression classifier offered in scikit-learn which is just another name for the Maximum Entropy classifier. Given its observed performance in previous studies as well as the use of n-gram features, we expect Maximum Entropy to yield better results compared to Naive Bayes.

Random Forest:

XGBoost:

4 EXPERIMENTS

This section begins by describing the assumptions under which the experiments were performed, the datasets and the classifier hyperparameters that were used. Following that, the feature selection process and its results are presented, with the classification results being detailed straight after that. The section concludes with the findings that arise from the tests conducted.

Experimental setup

To begin with, there were two labeled datasets available as already mentioned in Section 3. These contained 2,495 and 19,538 samples, and will be referred to as small and

[11] https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html#sklearn.feature_selection.mutual_info_classif

large from now on respectively. The large dataset was used as a training/validation set to train the classifiers and tune their hyperparameters. In particular, the k-fold stratified cross-validation provided by the scikit-learn[12] library was used in order to make the best use of the data available. The stratified version of the cross-validation was selected over the simple one to account for the dataset imbalance. More specifically, the large dataset corresponds to a prior distribution of around 25%/75% for clickbait/non-clickbait. As a result, by using the stratified version we were able to maintain a similar class distribution for each of the folds created during the cross-validation process. Regarding the number of folds, the proper value of k is highly related to the size of the dataset. In particular, alterations in the size of k result in models with different bias and variance. In our case, we decided to set k to 10 since it has been observed to provide a good bias-variance tradeoff [14] and our dataset is large enough. However, for smaller datasets, a decrease in the number of folds would be inevitable. Last but not least, the small dataset was used as a left out dataset to test only the best pre-trained and optimized classifier on true unseen data to obtain an unbiased estimate of the error.

Moving on to the features used, all the features described in Section 3 were extracted resulting in an output of 6,859 dimensions in total. Given the large number of features, we used the info-gain feature selection to filter out the desired number of features that maximize the data separability. It has to be underlined that the features were split into discrete and continuous groups in order to run the info-gain algorithm with the "discrete_features" parameter set to true and false respectively. As for the classifier hyperparameters, extensive tuning was performed the details of which can be found in Classifier hyperparameter optimization.

Another important thing to note is that the class labels used were 0 for the "non-clickbait" posts and 1 for the "clickbait" ones. Finally, five evaluation metrics were chosen to analyse the classification results obtained, namely accuracy, precision, recall and F_1 score. The first four metrics were selected since they are the most common ones used in the NLP literature to evaluate such systems while the F_1 score was chosen to allow for a reasonable classifier evaluation given the inherent class imbalance in the dataset.

Feature selection results

98 non-gram features

[12]https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html

Clickbait classification results

Default hyperparameters. XGBoost specified binary classification *Feature size selection*

Selected top 160, since 160 dimensions were the features with the highest accuracy for all four classifiers. Furthermore, best F_1 score for 2 of them, one very small difference from its highest value. The only one with a large decrease in Naive Bayes.

+ Naive: not so great performance but great speed!

Final results for x features

The values selected to obtain the results mentioned in this section are mentioned below:

- Naive Bayes: prior probabilities of 0.24-0.76 manually added
- Maximum Entropy: C = 1, penalty = l2, solver = liblinear
- Random Forest:
- XGBoost:

With optimized parameters:

Discussion

5 CONCLUSIONS

Summary of paper and results

Future research opportunities:

- Semi-supervised learning with unlabeled data (80000)
- Patterns with POS tags and n-grams
- PCA feature extraction
- use cost matrix to assign more cost for classifying non-clickbaits as clickbaits (or the other way around)
- Swap nltk with Stanford NLP
- Clickbait classification using website usage statistics (CTR, Bounce Rate, Avg time on page). User behavior is useful in clickbait detection similar to [6]

REFERENCES

[1] Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. Stop clickbait: Detecting and preventing clickbaits in online news media. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 9–16. IEEE, 2016.

[2] Prakhar Biyani, Kostas Tsioutsiouliklis, and John Blackmer. "8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[3] Yimin Chen, Niall J Conroy, and Victoria L Rubin. Misleading online content: Recognizing clickbait as false news. In *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*, pages 15–19. ACM, 2015.

[4] Aviad Elyashar, Jorge Bendahan, and Rami Puzis. Detecting clickbait in online social media: You won't believe how we did it. *arXiv preprint arXiv:1710.06699*, 2017.

[5] Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. Clickbait detection. In *European Conference on Information Retrieval*, pages 810–817. Springer, 2016.

Classifier\Feature Size	20	40	60	80	120	160	200	No ngrams
Naive Bayes	0.727	0.777	0.770	0.773	0.777	**0.783**	0.778	0.767
Maximum Entropy	0.793	0.805	0.801	0.807	0.805	**0.808**	0.806	0.804
Random Forest	0.803	0.820	0.822	0.824	0.826	**0.826**	0.825	0.823
XGBoost	0.811	0.824	0.829	0.830	0.831	**0.831**	0.831	0.830

Table 1: Accuracy score of difference feature sizes

Classifier\Feature Size	20	40	60	80	120	160	200	No ngrams
Naive Bayes	0.502	**0.523**	0.502	0.491	0.475	0.441	0.431	0.468
Maximum Entropy	0.371	0.462	0.458	0.483	0.477	**0.486**	0.485	0.476
Random Forest	0.476	0.538	0.542	**0.544**	0.543	0.540	0.537	0.535
XGBoost	0.476	0.541	0.564	0.568	0.567	**0.568**	0.567	0.564

Table 2: F_1 score of difference feature sizes

Algortithm	Accuracy	Precision	Recall	F_1 score
Naive Bayes	0.783	0.592	0.352	0.442
Maximum Entropy	0.837	0.732	0.524	0.610
Random Forest				
XGBoost				

Table 3: Clickbait classification results

[6] Hai-Tao Zheng, Xin Yao, Yong Jiang, Shu-Tao Xia, and Xi Xiao. Boost clickbait detection based on user behavior analysis. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data*, pages 73–80. Springer, 2017.

[7] Martin Potthast, Tim Gollub, Kristof Komlossy, Sebastian Schuster, Matti Wiegmann, Erika Patricia Garces Fernandez, Matthias Hagen, and Benno Stein. Crowdsourcing a large corpus of clickbait on twitter. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1498–1507, 2018.

[8] Deepanshu Pandey, Garimendra Verma, and Sushama Nagpal. Clickbait detection using swarm intelligence. In *International Symposium on Signal Processing and Intelligent Recognition Systems*, pages 64–76. Springer, 2018.

[9] Xinyue Cao, Thai Le, et al. Machine learning based detection of clickbait posts in social media. *arXiv preprint arXiv:1710.01977*, 2017.

[10] Abhijnan Chakraborty, Rajdeep Sarkar, Ayushi Mrigen, and Niloy Ganguly. Tabloids in the era of social media?: Understanding the production and consumption of clickbaits in twitter. *Proceedings of the ACM on Human-Computer Interaction*, 1(CSCW):30, 2017.

[11] Gang Wang, Christo Wilson, Xiaohan Zhao, Yibo Zhu, Manish Mohanlal, Haitao Zheng, and Ben Y Zhao. Serf and turf: crowdturfing for fun and profit. In *Proceedings of the 21st international conference on World Wide Web*, pages 679–688. ACM, 2012.

[12] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Robotics: Science and Systems*, volume 2, pages 65–72, 2005.

[13] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, pages 61–67, 1999.

[14] Max Kuhn and Kjell Johnson. *Applied predictive modeling*, volume 26. Springer, 2013.

A APPENDIX

Software versions

Library/Tool	Version
Stanford CoreNLP	3.9.2
vaderSentiment	3.2.1
NLTK	3.4
scikit-learn	0.20.2
XGBoost	X.X
bla	X.X

Classifier hyperparameter optimization

Naive Bayes

The prior probabilities for each class were manually added. These correspond to 0.76 for non-clickbait and 0.24 for clickbait posts. The resulting accuracy remained unaffected.

Maximum Entropy

Hyperparameters	Values
solver	["liblinear", "lbfgs"]
C	[100, 10, 1, 0.1, 0.01, 0.001, 0.0001]
penalty	["l2"]

Random Forest

Hyperparameters	Values

XGBoost

Hyperparameters	Values

Ngram Distributions

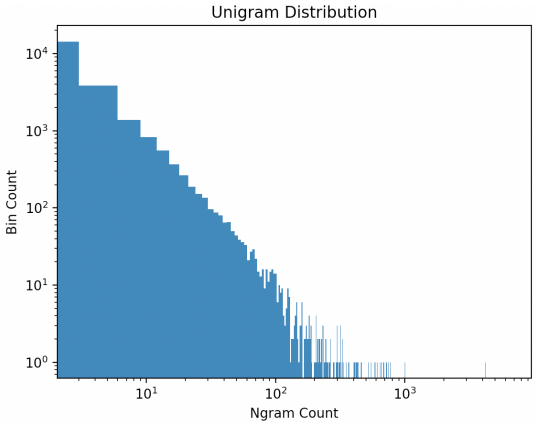

Figure 3: Unigram Log-Log Distribution

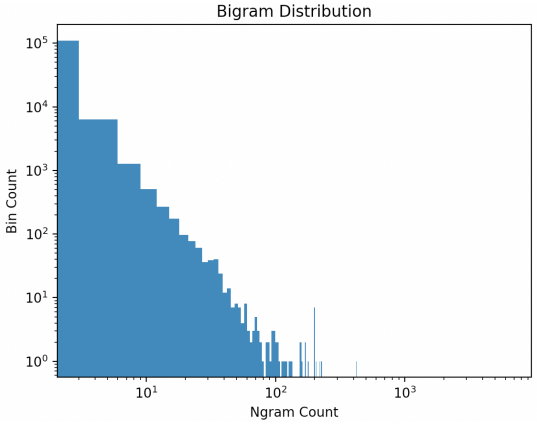

Figure 4: Bigram Log-Log Distribution

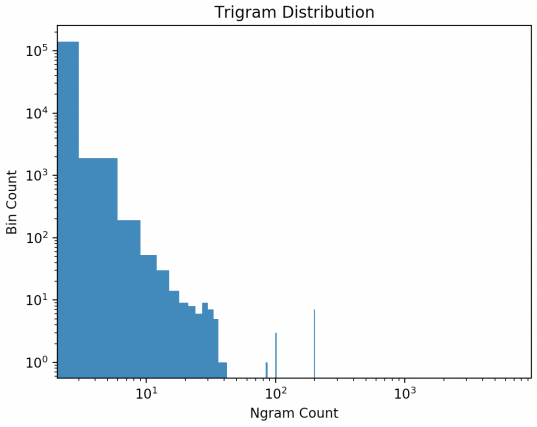

Figure 5: Trigram Log-Log Distribution