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 on 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 advertisements. 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 post-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 that clickbait detection can indeed be automated by applying machine learning classifiers to carefully designed features.

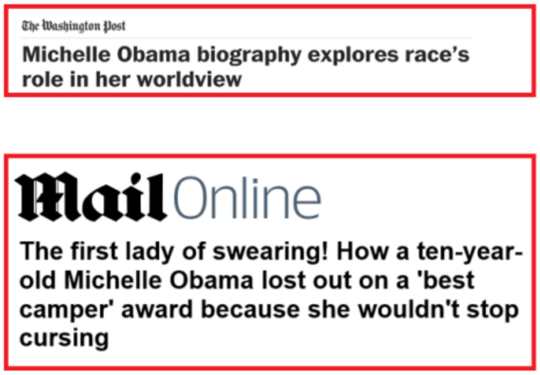
KEYWORDS

Clickbait, Classification, Automated detection.

1 INTRODUCTION

The rise of online news media over the past decade has upset the media ecosystem. 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 moving away from the trend were people remain loyal to one media source. Moreover, the main revenue source of online media is website advertisements. As a result, there is enormous competition in order 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 experience since the target articles are usually of low quality [2]. A characteristic example of such a title against its non-clickbait counterpart is available in Figure 1.



The Washington Post

Michelle Obama biography explores race's role in her worldview

MailOnline

The first lady of swearing! How a ten-year-old Michelle Obama lost out on a 'best camper' award because she wouldn't stop cursing

Figure 1: 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 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 discriminate clickbait from non-clickbait titles. These features are then evaluated on four different machine learning classifiers, namely Naive Bayes, Max Entropy, Random Forest and XGBoost. 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?

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

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

(2) Which machine learning techniques are more suited to the clickbait detection task?
(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, in this study the authors 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 that 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 study. The effectiveness of this proposal is then evaluated on four separate machine learning classifiers.

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 viewed at Figure 2.

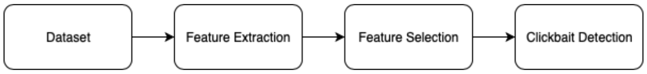


Figure 2: Clickbait Detection Pipeline

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 Appendix A.

Dataset Overview

The data used to develop the system proposed were the publicly available datasets from the Clickbait Challenge 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.

Feature Extraction

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. It has to be underlined that the features that were inspired from

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

[4] https://github.com/agamvrinos/NLP-Clickbait-detection

the original paper [4] chosen for reproduction are marked with an asterisk(*). Moreover, there was no preprocessing performed on the entire corpus with the exception of some preprocessing steps (e.g. lowercase) before the extraction of particular features. In the latter case, these are mentioned explicitly. The details of each feature are described below:

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 [4] and [1]. However, these two papers reach contradicting conclusions on whether clickbait or non-clickbait posts have larger titles. In this paper, we decided to calculate the length of four content fields both in terms of words and characters. More specifically, these features were extracted for the post title, article title, article description and article keywords. It has to be stressed that the content was tokenized using the space character as a delimiter in the case of the length count and that whitespace was included in the character count features. Moreover, if the content item contains more than one element, the feature is calculated using the average number of characters/words. An example of the length based extraction is found in Table 1. In total, eight features were extracted, four word length based and four character length based.

Content	Character Length	Word Length
Apple's iOS 9 new feature	25	5
Kate Moss 'kicked-off flight'	29	4
['Media', 'CEOs', 'business news']	7.33	1.33

Table 1: Content character/word length examples

Difference between number of characters/words*: Another feature that [4] found to separate the two types of posts is the absolute 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. The values of this feature are calculated as follows:

$$diff\_len(cont_a, cont_b) = |len(cont_a) - len(cont_b)|$$

where $cont_a$ and $cont_b$ refer to the two content elements that are compared in each case.

Number of characters/words ratio*: Similarly to the difference features, [4] proposes the use of the ratio between the number of words/characters of two separate content elements. As a result, twelve more features were created.The values of this feature are calculated as follows:

$$ratio\_len(cont_a, cont_b) = \frac{len(cont_a)}{len(cont_b)}$$

where $cont_a$ and $cont_b$ the two content elements.

Begins with interrogative: The authors in [2] propose a binary feature which checks whether the content element begins with an interrogative word[5]. To be more exact, we check whether the post title and the article title begin with one of the following substrings: *"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 attract user attention are being used over and over again. To that end, as in [5] we used a dictionary[6] of 73 common phrases taken from the Downworthy browser plugin and looked for these in the title of each individual post and article to create two extra features.

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"* are commonly seen as part of the title of clickbait posts and articles. 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[7], and extracted binary features by looking at the presence of at least one of these words.

Part-Of-Speech tags: In [1],[9] and [8] the use of Part-Of-Speech (POS) tags as features has been proven to yield interesting results in the clickbait detection problem. For example, these features can either be the count or 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 the count of the proper nouns, abbreviations and pronouns are features well-suited to our task. Consequently, this lead to our decision to only consider the POS tag counts. Our implementation only considers the 16 POS tags mentioned in [1] since they were the most intuitive ones. Additionally, we created the two pattern features proposed by [9]. These are: "number + noun phrase + verb"

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

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

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

and "number + noun phrase + that". The library used for the POS tag extraction is NLTK[8]. Regarding preprocessing, we tokenized the sentence using whitespace as a delimiter and gave it as input to the POS tagger. However, we observed that the NLTK tagger had the issue of tagging every noun in the middle of the scentence that starts with a capital letter as a proper noun. Thus, what we did to combat this issue was to analyze every word separately making it appear as a first word in a sentence, enabling NLTK to assign it the correct tag. In hindsight the Stanford NLP POS tagger would have been a better option due its better performance in the aforementioned scenario. Finally, since we only wanted to check the presence of a proper noun regardless of whether it is in plural or not, the "NNP" counter feature was also incremented when the plural form "NNPS"appeared. The same was done for the 'NN'/'NNS' tags.

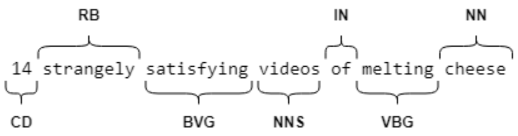


Figure 3: A post title annotated by the NLTK POS Tagger

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 in the clickbait detection task. In our study, the detection of such words is being performed with the use of NLTK's POS Tagger. In particular, we are searching for words tagged with *"DT"* in the post/article title fields. The preprocessing, setbacks and fixes mentioned in the "Part-Of-Speech tags" features were also performed here.

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 by [1] and was considered essential in distinguishing between clickbait and non-clickbait posts. Similarly to the determiners, we looked for *"PRP"* and *"PRP$"* POS tags in each post and article title to extract their occurence counts. Once again, the same preprocessing and fixes as in "Part-Of-Speech tags" features were applied.

N-gram Features

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 or 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 trigrams can be very good indicators of clickbait content, such as *"10 things that"*, *"you can't believe"* and *"what happens when"*. Similarly, they can be indicators of non-clickbait content: *"man charged with"* and *"national emergency as"*. The preprocessing done in the n-gram feature generation is to make every character lowercase and to remove all the special characters without replacing them with whitespace.

Our implementation generated tens of thousands of n-grams, that led us to use two thresholds(lower, upper) that filter n-gram counts below or above them. The selection of these was done by checking their frequency distribution in the entire dataset, as shown in Appendix C. The thresholds selected for the unigrams, bigrams and trigrams were (6, 1000), (6, 200) and (6, 100) respectively. The lower threshold was applied to remove the least influential n-grams. On the contrary, the upper threshold was used for the exact opposite reasons. In particular, if an n-gram appears too many times in the entire dataset, the classifiers will be unable separate the two classes. This led to the n-gram feature size to shrink to 6,759. Furthermore, we realized that the named entity n-grams are dataset specific and that would lead to overfitting on the provided dataset. Thus, we decided to remove them by searching for the POS tags 'NNP' or 'NNPS' using the NLTK POS tagger. That said, the Stanford's Named Entity Recognizer[9] would have been a better choice for this task. Finally, we acknowledge that the n-gram features can be too many, leading the classifiers to fall in the curse of dimensionality. Therefore, we decided to use a feature selection method prior to the classification, as described in the following parts of this section.

Sentiment analysis

Hyperbolic features: Clickbait post and article titles often contain particular words that are carefully chosen to attract the reader's attention. A common characteristic of all these words is hyperbolism. Examples of such words are *"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 titles 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 and article title. As opposed to other existing Sentiment analysis tools that use a three-category approach

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

[9] https://nlp.stanford.edu/software/CRF-NER.html

(negative, neutral, positive), CoreNLP includes the "Very positive" and "Very negative" classes, something that allows for the detection of hyperbolism. It has to be mentioned that we tokenized the post/article titles and tested each of the resulting words against Stanford's tool to identify the presence of hyperbolic words.

Sentiment Polarity features: Apart from detecting hyperbolic words, we also performed Sentiment analysis in order to identify the polarity of each post/article 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 and article title separetly to create two numeric features.

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. Therefore, we created 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 [4][5][6] have 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.

Feature Selection

The previous part of our implementation results into 6,857 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 98. 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 used the scikit-learn 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 for this task, we chose two probabilistic (Naive Bayes, Maximum Entropy) and two ensemble based on decision trees (Random Forest, XGBoost) classifiers:

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

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

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 is often used as a baseline classifier. Furthermore, Naive Bayes is fairly 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.

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 Logistic Regression 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: In the field of NLP one of the most prominent ensemble classifiers is Random forest [14][15]. Random Forest is an ensemble of decision trees that are created from a random sub-sample of the training set. The final classification decision is made by aggregating all of the individual decision tree votes. A strength of the Random Forest is the addition of randomness at the node-splits. This allows the node to search for the best feature among a random subset of features. As a result, it generalizes better and avoids overfitting on the training set. In our study, we used the Random Forest classifier offered in scikit-learn.

XGBoost: In recent years XGBoost [16] has been used extensively in NLP research [17][18], winning multiple awards[12] [13]. In particular, it is an ensemble classifier that uses gradient boosted decision trees. This means that instead of updating the weights of the training instances, it optimizes the model's loss function. In our implementation we use the hinge loss $l(y)$ as the model's loss function:

$$l(y) = max(0, 1 - t * y)$$

where y is the classifier prediction and $t = \pm 1$ is the true label. Our implementation uses, the XGBoost classifier offered by the Distributed Machine Learning Community (DMLC)[14].

4 EXPERIMENTS

This section describes the assumptions under which the experiments were performed, and the datasets/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 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[15] 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 24%/76% for clickbait/non-clickbait. 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 trade-off [19] and our dataset yields large

[12] http://stat-computing.org/awards/jmc/winners.html

[13] https://higgsml.lal.in2p3.fr/prizes-and-award/award/

[14] https://xgboost.ai/

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

enough folds. However, for smaller datasets, a decrease in the number of folds would be inevitable. Lastly, the small dataset was used as a holdout dataset to test the best 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 6,857 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 classifiers, different setups were used to tune the related hyperparameters, the details of which can be found in Appendix B.

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, four evaluation metrics were chosen to analyze the classification results obtained, namely accuracy, precision, recall and F_1 score. The first three 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

The fact that 6,857 features were extracted led to the need for a feature selection method to avoid falling into the curse of dimensionality and to keep the computational time in check. A detailed breakdown of the number of features extracted can be found in Appendix D. Therefore, we decided to take advantage of the info gain feature selection method also used by [4]. The top 20 features as obtained from the info gain run on the large dataset are available in Table 2.

Rank	Feature	Info Gain Value
1	Post Title Word Length	0.0627
2	Post Title Character Length	0.0627
3	Post Title NN	0.0591
4	Ratio Characters Post Title/Article Title	0.0484
5	Ratio Words Post Title/Article Description	0.0476
6	Ratio Characters Post Title/Article Description	0.0430
7	Ratio Words Post Title/Article Title	0.0409
8	Difference Characters Article Title/Article Description	0.0408
9	Difference Characters Post Title/Article Keywords	0.0400
10	Characters Article Description	0.0360
11	Words Article Description	0.0360
12	Ratio Words Post Title/Article Keywords	0.0342
13	Difference Words Post Title/Article Keywords	0.0320
14	Ratio Characters Article Title/Article Description	0.0320
15	Post Title NNP	0.0318
16	Ratio Words Article Title/Article Description	0.0315
17	Difference Characters Post Title/Article Description	0.0304
18	Ratio Characters Post Title/Article Keywords	0.0265
19	Difference Words Article Title/Article Description	0.0257
20	Difference Words Post Title/Article Description	0.0165

Table 2: Info gain output - Top 20

The info gain values for all the features can be found on GitHub[16]. The feature selection outputs allows us to conclude the following:

- The length based features proposed by [4] seem to be very discriminative as they dominate the rankings. In particular, 18 out of the top 20 features belong to this feature group.
- The word and character length based features are not independent and their values are frequently correlated. This is noticeable by the fact that they achieve comparable info gain values in a number of cases such as the post title word/character length features.
- The POS tag features are very reasonable with most of them appearing in the top 50 features. The best performing ones that belong to this category are the NN/NNP tags which count the nouns/entities in a post/article title.
- The n-gram features are also worth mentioning since a number of them appear in the top 100. The one with the highest info gain value is the unigram "you" which appears in position 21. Some of the n-grams that appear afterwards are "your", "this", "things", "these", "why" and "what". A quick look at clickbait titles on the Web validates that these n-grams commonly appear at such articles.
- Both of the "begins with number" features appear in the top 25 which validates the claim of [5] and our personal experience on clickbait post/article titles.

[16] https://github.com/agamvrinos/NLP-Clickbait-detection/blob/master/info_gain.csv

Classifier\Feature Size	20	40	60	80	120	160	200
Naive Bayes	0.727	0.777	0.770	0.773	0.777	**0.783**	0.778
Maximum Entropy	0.793	0.805	0.801	0.807	0.805	**0.808**	0.806
Random Forest	0.803	0.820	0.822	0.824	0.826	**0.826**	0.825
XGBoost	0.811	0.824	0.829	0.830	0.831	**0.831**	0.831

Table 3: Accuracy of difference feature sizes

Classifier\Feature Size	20	40	60	80	120	160	200
Naive Bayes	0.502	**0.523**	0.502	0.491	0.475	0.441	0.431
Maximum Entropy	0.371	0.462	0.458	0.483	0.477	**0.486**	0.485
Random Forest	0.476	0.538	0.542	**0.544**	0.543	0.540	0.537
XGBoost	0.476	0.541	0.564	0.568	0.567	**0.568**	0.567

Table 4: F_1 score of difference feature sizes

- The interrogative and possessives features also appeared in the top 55 validating previous findings in the literature [1][2] that these features can be discriminative.
- Another observation is that the punctuation related features are scattered around in various positions outside the top 40. An alternative feature that could be designed would be to group the count of punctuations in one feature. As a result, such a feature would check whether clickbait posts use more punctuations compared to non-clickbait ones.

Following the info gain output, the next step was to decide on the optimal number of features that should be used. In order to perform that selection, we run all four classifiers using their default parameters on a number of difference feature sizes, namely 20, 40, 60, 80, 120, 160 and 200. Then based on the accuracy and F_1 score of these tests, the optimal size was selected. The results obtained during our tests are available in Tables 3 and 4.

Analyzing the results in the aforementioned tables, it becomes clear that the optimum dimension size is using the top 160 features. To be more exact, all four classifiers achieve their highest accuracy with that size value. Moreover, two out of four classifiers achieve their higher F_1 score at 160 features and the Random Forest yields a value which is only 0.004 smaller than its best overall value. The only classifier that has a considerably smaller F_1 value at 160 dimensions is the Naive Bayes Classifier. However, since this classifier was not expected to be the benchmark of our study, we decided to sacrifice some of its performance by opting to go for the top 160 features.

Clickbait classification results

Having decided on the number of features that will be used, the next step was to optimize the hyperparameters of each classifier. The values that were found to yield the best performance are mentioned below:

- *Naive Bayes*: prior probabilities of 0.24-0.76 were manually inserted
- *Maximum Entropy*: C = 1, penalty = l2, solver = liblinear
- *Random Forest*: n_estimators = 500, max_depth = 30, bootstrap = False
- *XGBoost*: objetive = binary:hinge, the rest of the hyperparameters were set to their defaults

Moreover, the values of all the different hyperparameter combinations that were tried out can be found in Appendix B. The four classifiers that were tested in this paper, were evaluated based on four metrics:

- *Accuracy*: calculate the percentage of predictions that the model correctly classified
- *Precision*: out of all the posts classified as clickbait, how many are indeed clickbait?
- *Recall*: out of all the clickbait samples in the dataset, how many have been classified as clickbait?
- *F_1 score*: this metric is calculated as a function of precision and recall. It is usually used in the case of uneven class prior distributions as is the case for the large dataset used for this application.

Following the hyperparameter optimization, the classifiers were run using their best settings on the large dataset using 10-fold stratified cross-validation. The results obtained from these tests are detailed in Table 5.

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	0.827	0.746	0.442	0.555
XGBoost	0.831	**0.754**	0.456	0.568

Table 5: Clickbait classification results

Based on the results, we can conclude the following:

- All four classifiers obtain accuracy values higher than the dataset bias. What is meant by that is that classifying all the objects to the dominant class (non-clickbait) would result in an accuracy of around 76% which is lower than the results obtained. This is a first indication that the features designed enable the discrimination between the two classes. Moreover, by examining the corresponding confusion matrices during cross-validation, we observed that indeed the used features allow the classifiers to be discriminative.
- Maximum Entropy obtains the highest values in three out of four metrics. To be more exact, it has the highest accuracy, recall and F_1 score, while XGboost yields the highest precision.
- Naive Bayes is the classifier with the worst performance in all four metrics. That said, its training time was the smallest among these options by a large margin. Therefore, it could still be a reasonable choice in the case of a very large training set.

According to our test results on the large dataset, the Maximum Entropy classifier should be considered as the best model trained using the features designed in this paper. That said, we were curious to test how well that specific classifier generalizes to new unseen data. For that reason, we used the small dataset provided and obtained the results found in Table 6.

Algorithm	Accuracy	Precision	Recall	F_1 score
Maximum Entropy	0.725	0.612	0.306	0.408

Table 6: Maximum Entropy results on holdout dataset

Analyzing the results obtained, it becomes clear that its performance is significantly reduced compared to its results from the 10-fold cross-validation. Furthermore, the confusion matrix obtained via the test run on the holdout set is available in Figure 4.

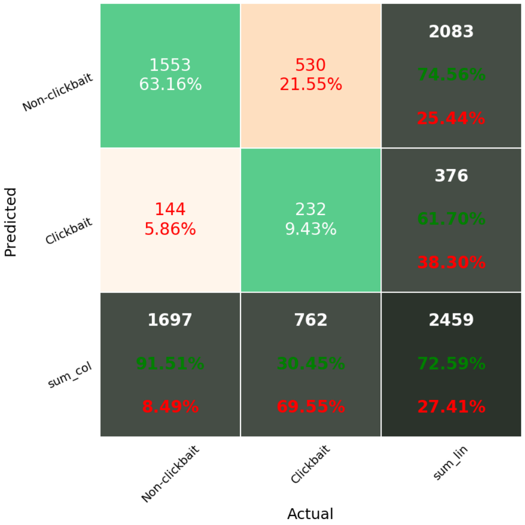


Figure 4: Maximum Entropy confusion matrix

The confusion matrix clearly shows that the classifier makes more mistakes when classifying clickbait posts as non-clickbait (false positive) than the other way around (false negatives). One of the contributing factors could be the imbalance of the large dataset and its shortage of clickbait samples. Last but not least, we made sure to visually inspect samples that are missclassified. While some of the post-article combinations were mistakenly categorized due to inefficiencies of our system, we also noticed three issues in the dataset. More specifically, there are numerous samples in the training/validation data that are mistakenly annotated as non-clickbait when the content is clearly clickbait. Similarly, some post-article combinations are very ambiguous on whether the content is actually clickbait or not, even for a human annotator. What is more, we noticed that some of the missclassified objects contain forms of irony. This leads us to conclude than the inclusion of some irony detection features might have improved the performance of our system. A few examples of those issues are available in Table 7.

5 CONCLUSIONS

This study attempted to tackle the issue of clickbait detection with the use of machine learning techniques. More specifically, we used a rich set of features (linguistic, n-grams, sentiment, others) covering different aspects of the underlying task and assessed their discriminability using the Information Gain algorithm. Over and above, we evaluated the performance of four different classifiers namely, Naive Bayes, Maximum Entropy, Random Forest and XGBoost. Judging by the performance of each classifier during our cross-validation procedure, we selected Maximum Entropy to predict clickbait/non-clickbait posts on our holdout set.

The experiments conducted within the context of this study provided some interesting insights. Firstly, we identified that clickbait detection can indeed be automated, although the features used must be carefully selected. That said, we observed that particular types of features such as length-based, POS tags and n-grams are the most discriminative when it comes to clickbait post classification. Furthermore, Maximum Entropy showed the best performance during cross-validation and its performance on the test set can be characterized as reasonable given the large imbalance of the dataset. Lastly, by diving deeper into the missclassified posts we came to realize the importance of the corpus

Missclassification	Sample Field		
Reason	**Post Title**	**Article Title**	**Annotated Label**
Dataset Annotation	27 faces everyone who's ever done a poo will recognize	27 Faces Everyone Who's Ever Done A Poo Will Recognize	no-clickbait
	Here's what we think Pokemon would taste like	Here's What We Think Pokemon Would Taste Like	no-clickbait
Sample Ambiguity	Obama actually writes back to people who call him an idiot	Obama Actually Writes Back To People Who Call Him An Idiot	no-clickbait
	Man dies when car plunges from parking garage	Man Dies When Car Plunges from New Orleans Parking Garage	clickbait
Irony Detection	Nobel-winning sexism in the lab via @RoomForDebate	Nobel-Winning Sexism in the Lab - Room for Debate - NYTimes.com	clickbait
	Russian teenagers 'set on fire' while trying to take the 'ultimate selfie' on train roof	Russian teenagers 'set on fire' while trying to take the 'ultimate selfie' on train roof	clickbait

Table 7: Issues with missclassified samples

collection/anotation procedure and how biased and/or low-quality datasets can influence the results obtained in an NLP problem.

In conclusion, we would like to point out some future research opportunities that could potentially increase the performance of clickbait detection. Firstly, since misclassifying non-clickbait posts to clickbait is generally assumed to be worse than the other way around, the use of a cost matrix which would penalize this could presumably lead to better results. What is more, features related to website usage statistics (e.g. CTR, Bounce Rate, Avg. time on page) and irony-detection could prove valuable in clickbait detection and should be researched further. Lastly, the application of semi-supervised learning techniques could also lead to improved results. In particular, the Clickbait Challenge 2017 provides a publicly available dataset of 80,012 unlabeled samples. This dataset could be used to provide additional artificially labeled data to the classifiers, thus potentially increasing their performance.

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.

[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] Łukasz Kobyliński and Adam Przepiórkowski. Definition extraction with balanced random forests. In *International Conference on Natural Language Processing*, pages 237–247. Springer, 2008.

[15] Yi Su, Frederick Jelinek, and Sanjeev Khudanpur. Large-scale random forest language models for speech recognition. In *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[17] Matej Martinc, Iza Skrjanec, Katja Zupan, and Senja Pollak. Pan 2017: Author profiling-gender and language variety prediction. In *CLEF (Working Notes)*, 2017.

[18] Michael Sungjun Kim, Jiwei Liu, Xiaozhou Wang, and Wei Yang. Connecting devices to cookies via filtering, feature engineering, and boosting. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 1690–1694. IEEE, 2015.

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

A SOFTWARE VERSIONS

Library/Tool	Version
Stanford CoreNLP	3.9.2
vaderSentiment	3.2.1
NLTK	3.4
scikit-learn	0.20.2
XGBoost	0.82

B 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
n_estimators	[100, 200, 300, 400, 500, 1000]
max_depth	[20, 30, 40, 50, 60]
min_samples_split	[2, 4, 6]
min_samples_leaf	[1, 2, 4]
bootstrap	[True, False]

XGBoost

Hyperparameters	Values
n_estimators	[100, 200, 300]
eta	[0.05, 0.1, 0.2, 0.3]
gamma	[0, 0.1, 0.2, 0.3, 0.4, 0.5]
reg_lambda	[1e-5, 1e-2, 0.1, 1, 100]
max_depth	[4, 5, 6]
min_child_weight	[4, 5, 6]

In the case of XGBoost, the tested hyperparameter combinations did not result in any increase in the performance of the classifier. Thus, the default values were used for the final XGBoost classifier.

C N-GRAM DISTRIBUTIONS

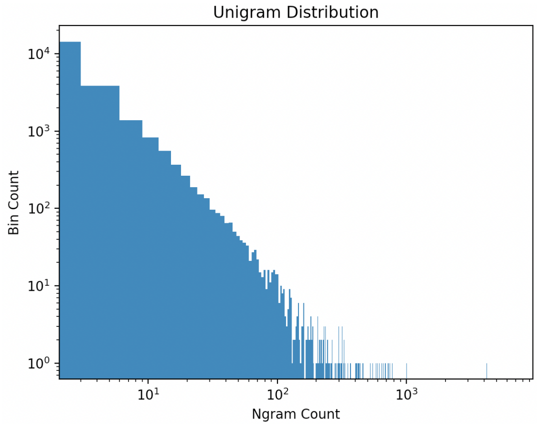


Figure 5: Unigram Log-Log Distribution

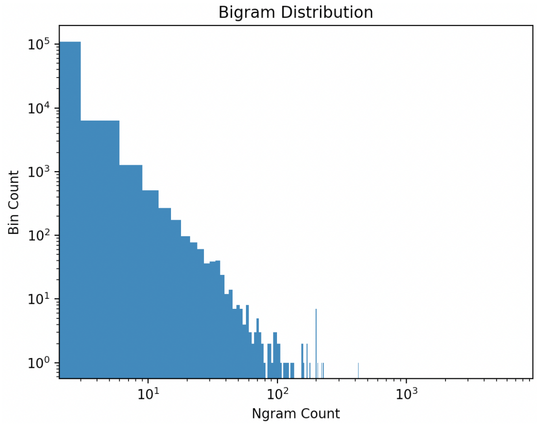


Figure 6: Bigram Log-Log Distribution

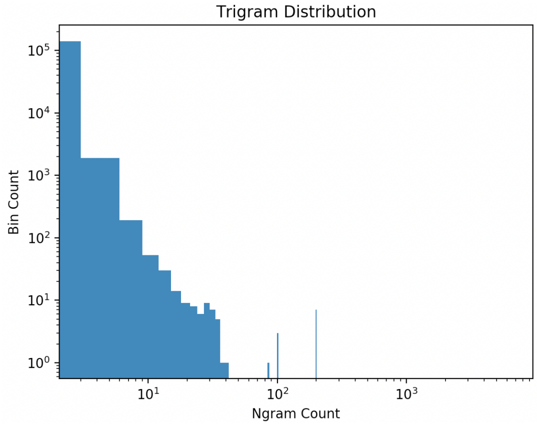


Figure 7: Trigram Log-Log Distribution

D FEATURE OVERVIEW

Feature Name	Features	Description	Type[17]
Word Length	4	Number of words in post title, article title, article description and article keywords.	N
Character Length	4	Number of characters in post title, article title, article description and article keywords.	N
Diff. between no. of words	6	Difference in the number of words of all the possible combinations of the post title, article title, article description and article keywords fields.	N
Diff. between no. of characters	6	Difference in the number of characters of all the possible combinations of the post title, article title, article description and article keywords fields.	N
No. of words ratio	6	Ratio of the number of words of all the possible combinations of the post title, article title, article description and article keywords fields.	N
No. of characters ratio	6	Ratio of the number of characters of all the possible combinations of the post title, article title, article description and article keywords fields.	N
Begins with Interrogative	2	Presence of Interrogatives (Who, What, How, etc.) at the beginning of the post title and article title.	B
Common clickbait phrases	2	Presence of common clickbait phrases (e.g. Unbelievable, Mind-blowing) in the post title and article title.	B
Slang words/phrases	2	Presence of slang words phrases (e.g. OMG, LOL) in the post title and article title.	B
Part-Of-Speech tags	2*(16+2)	For the post title and article title, we extract 16 features that correspond to the number of occurrences for each individual tag (e.g. NN, IN) and 2 features that refer to the number of occurrences of particular combinations of the tags (CD + NN + VB, CD + NN + that).	N
Determiners	2	Presence of Determiners (e.g. their, my) in the post title and article title.	B
Possessives	2	Presence of Possessives (e.g. I, You) in the post title anad article title.	B
Unigram	4354	Count of occurrences for each unigram. Note that the extraction of unigrams is corpus-dependent. The mentioned number refers to the extraction of unigrams from our training dataset.	N
Bigrams	2153	Count of occurrences for each bigram. Similarly to unigrams, the number refers to the training dataset that was used.	N
Trigrams	252	Count of occurrences for each trigram. Similarly to unigrams, the number refers to the training dataset that was used.	N
Hyperbolic features	2	Presence of hyperbolic words (e.g. awesome, frustrated) in the post title and article title.	B
Sentiment polarity features	2	Intensity of the sentiment extracted from the post title and article title. The returned values lie in the range [-1,1].	N
Has image	1	Presence of an image in the article	B
Begins with number	2	Presence of a number at the beginning of the post title and article title	B
Punctuation count	10+2	Count of *!,?,’,...,.* occurrences at the post title and article title & count of #,@ at the post title	N
Post creation hour	1	The creation hour of the particular post. The returned value is in the range of [1,24]	N

[17] N: Numeric, B: Binary