Top [n] ways to detect Clickbait posts

An NLP approach to Clickbait Detection

Alina Bianca Iancu, Gabriele Mazzola, Matteo Pocchiari, Ana Šemrov
Technische Universiteit Delft
{a.iancu-1, g.mazzola, m.pocchiari, a.semrov}@student.tudelft.nl

ABSTRACT

In this work we aimed to reproduce the paper written by one of the participating teams at the 2017 Clickbait Challenge, where the goal was to successfully detect clickbait posts on social media. To that end, we engineered and selected a number of features based both on clickbait posts and the content they lead to, and trained a number of machine learning classifiers in order to determine the differences between clickbait and non-clickbait posts. The best performance in terms of the official Clickbait Challenge metrics, a mean squared error of 0.044, was obtained by a Random Forest classifier.

Author Keywords

Clickbait, NLP, Classification, Twitter, Machine Learning

INTRODUCTION

Online news platforms offer multiple benefits to the modern user in contrast with traditional news outlets. For example, news can be aggregated from multiple sources and the selection of content personalised, users can stay up-to-date by being provided with real-time updates, and the services are predominantly free. However, not charging a subscription fee drives such online portals towards advertisement-based revenue models. The users' attention and clicks thus represent a direct source of revenue, which makes the content creators and news portal administrators employ a variety of psychological tricks to lead readers to their news pages.

One of the most popular ways of encouraging readers to visit an online media portal is by advertising through social media using *clickbait* posts. These short teaser posts spark curiosity among the readers by playing on emotions and cognitive biases [19] at the expense of informativeness, all in order to lead the readers to follow the attached link to the news portal. In addition to sensationalism and exaggeration (usually seen in tabloid journalism), both the clickbait posts themselves and the content they lead to have been seen as contributing to the rapid spread of false news, rumors, and misinformation on the Internet [8].

To encourage research in the direction of automated clickbait post detection, a competition called *Clickbait Challenge*[1] was organised in 2017 by Bauhaus-Universität Weimar. In this work we aimed to reproduce the paper written by one of the participating teams - team Salmon [10], where the authors extracted a variety of content related features (text- and image-based) both from clickbait posts and the content they lead to, and trained a number of machine learning classifiers to determine the differences between *clickbait* and *legitimate* posts.

[1] `https://www.clickbait-challenge.org/`

RELATED WORK

When researching the literature, we have observed that the clickbait issue has been discussed from multiple perspectives.

The effects of clickbait on users have been analyzed from a psychological perspective. For example, in [5] the authors discuss the forward-referring technique usually used in clickbait for inducing curiosity and luring the users to click.

When it comes to the technical perspective, a major step in the direction of clickbait detection was made by the authors of [19]. They created the first clickbait corpus of articles extracted from Twitter and developed a clickbait model based on a Random Forest classifier that achieved 0.76 precision and recall.

Concerning the approaches taken when classifying clickbait and non-clickbait articles, machine learning and deep learning are widely used techniques in the literature. Regarding machine learning, in [4] the authors used a variety of features and employed Gradient Boosted Decision Trees for the classification, achieving good performance and showing that informality is an essential indicator of clickbait content. Moreover, in [7] the authors implemented a model for automatically detecting clickbait and they created a browser extension for warning the users about the possible clickbait cases. They obtained a classification accuracy of 93% using a Support Vector Machine with Radial Basis Function kernel. Furthermore, the research carried out in [13] focused on classifying clickbait on Instagram based on three different types of features (text, meta, and image). Using Random Forest classifier the authors achieved an accuracy of 86.3%. When it comes to deep learning techniques, Convolutional Neural Networks [2] and models based on variational autoencoders [21] have proven to achieve good performance.

Additionally, we have observed different approaches being combined in literature in order to combat limitations specific to clickbait. For example, in [11] the authors combine POS tagging with Named Entity Recognition to overcome the problem of title casing.

Interesting to notice was the fact that clickbait is also analyzed in relation with other areas. For instance, in [6] the authors go from clickbait to fake news and stance detection in article headlines, attempting to separate related and unrelated headlines. Furthermore, they classify the related articles, achieving an accuracy of 89.59%.

DELIVERABLES

Our code is publicly available on GitHub and can be found at: `https://github.com/GabrieleMazzola/NLP_Group22_TUDelft`.

The Appendix contains the list of packages we used, together with their version number.

APPROACH

Dataset description

The data have been collected from the 2017 Clickbait Challenge website[1], where the participants were given access to different datasets of data crawled from the web (mainly Twitter given the large presence of Twitter-specific elements, like hashtags, '@' to reference another user and *RT* which indicates a retweet).

We used the labeled training set with 2459 elements (containing 1697 non-clickbait and 762 clickbait elements) to get an initial insight into the properties of the dataset. Then we used the labeled training/validation set with 19538 elements (with a split of 14777 non-clickbait and 4761 clickbait), divided as follows: 80% used for training/tuning and 20% for testing. Each element in the datasets is uniquely identified with an *id* and refers to a post found on social media. Along with post-specific information (*Post Title, Post Timestamp, Post Media*), each element also contains information about the article the post links to (namely *Article Title, Article Description, Article Keywords, Article Paragraphs, Article Captions*).

There is also a distinct folder available where it is possible to find the media (images in this case) that are present in the post and in the article. Each *id* is linked to a list of media that can be found in the media folder (empty list if the post or the article do not contain any image). The two datasets come as JSONL files, which we readily converted into normal JSON format.

Feature Extraction

Average Lin similarity between words in Post Title

We chose the Lin similarity [16] metric because it takes into consideration not only how similar two words are, but also how different. In this case, it would have been possible to use the Resnik method [20], considering that with the preprocessing of the title we do not find any duplicate words, but we decided to stick with the Lin one because we think it is meaningful to consider the difference between words as well.

We preprocess the original Post Title as follows: first, we removed the words starting with '@', '#' and words present in a predefined set of not useful words [2]. We created this set analyzing some of the titles we had problems to deal with. Later, we tokenized the title with NLTK Tokenizer [3] and perform Named Entity Recognition using the NER developed by Stanford NLP group [4] in order to remove all the tokens recognized as 'person', 'organization' and 'location'. We did this because such entities would not be recognized as words in the WordNet dictionary, reducing in this way the number of valid tuples of two words for which we should compute the similarity. After, we removed the punctuation and the stopwords present in the title, then applied case folding (lower case) to every character. Next step was lemmatizing the remaining words using again the NLTK [5] python library. Finally, we removed the numbers and all the words with less than two characters that are still present in the title. Here we removed duplicate words as well.

After these steps, we have a list of n unique words. We compute all the possible K combinations of 2 words out of this list, thus $K = \binom{n}{2}$. For a given tuple composed of two words (w_1, w_2) we look both for w_1 and w_2 on WordNet: if the word is present in it, a list with all the meanings of the word is returned, otherwise the returned list is empty. In case we have a non-empty list, we always consider the first element of the list to be the candidate word for which we will calculate the similarity. We compute the similarity for all the tuples and then we average the final result dividing by K. The used formula is:

$$lin_{avg} = \frac{\sum_{i=0}^{K} lin_i(w_1, w_2)}{K}$$

We think that non-clickbait posts should have (slightly) more similar but also more relevant words, where relevant here means not being cut down in the preprocessing part: this means that the number of 2-words combinations will be higher yielding a lower lin_{avg}. The opposite can be said for the clickbait posts. The downside of this procedure is that we blindly accept the first meaning of a word to be the candidate one, while we should consider in each case which meaning could be the more appropriate to the context.

The number of words in the list after preprocessing could be used on its own as feature because, from a quick analysis of the results obtained, clickbait posts have (on average) a smaller number of words in the final list compared to non-clickbait. We decided though to combine this with the average similarity and see if there is still a pattern there.

Presence of common phrases

Thanks to [7], we had access to a list of common phrases that are typical of clickbait posts, such as *"totally blew my mind"*, *"will make you want"*, *"you won't believe"*, etc. These phrases are directly aiming at the curiosity gap to push the reader towards the news. We noticed that most of them are phrases with an explicit reference to the reader or to the writer, trying to diminish the distance between them. The usage of a more "friendly" lexicon will be studied further in details under the perspective of POS tagging in the dedicated section.

Normalized number of capital characters in Post Title

We decided to design this feature because from our experiences on social networks most of the clickbait posts present a capitalized version of normal words in order to catch the attention of the reader in the homepage among all the other posts. However, we perform some preprocessing to avoid

[2] RT, 's, 're, u, http, https

[3] `https://www.nltk.org/api/nltk.tokenize.html`

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

[5] `https://www.nltk.org/_modules/nltk/stem/wordnet.html`

miscounting properly capitalized letters (for example, correctly capitalized names of people, locations and organizations) as clickbaity capitalization.

The preprocessing steps examined for this feature are very close to the ones employed in the average Lin similarity calculations: indeed we started by splitting the string into tokens using NLTK Tokenizer [6] and apply again Stanford NER [7] tool to retrieve the entities. Then, for each token, if it is recognized as a person, an organization or a location, it is transformed to lower case, otherwise it is left like it is. Finally, we normalized the number of capital characters by dividing by the number of tokens found in the string.

Number of words in Post Title and number of characters related features

From the original paper [10] we decided to keep only these features out of all the ones designed by the authors because they are the only ones really meaningful for us. As number of words in Post Title we considered the number of tokens obtained in the calculation of the normalized number of capital characters. For the number of characters, the name speaks for itself. As the original paper is doing, we considered the number of characters in *Post Title, Article Title, Article Description, Article Keywords, Article Captions, Article paragraphs*.

Average word length in Post Title and Article Title

Before calculating the average word length, we again preprocess the strings by removing the punctuation and stopwords that are not adding any information to the feature. We opted for this feature because we believe clickbait posts use shorter words, especially due to the higher presence of shortenings (also discussed in [7]).

We tested this hypothesis with our data and we can confirm that this is indeed the case. As in [7], we checked the percentage of post titles that contain shortened forms of words, such as *I'm*, *you'd*, *they'll*, *didn't* and so on, both in the case of clickbait and non-clickbait articles. We found that 10.1% of the clickbait articles contain shortenings of words, while this is the case for only 5.4% of the non-clickbait articles. Thus, as clickbait posts are expected to have a higher value than non-clickbait ones, we additionally considered the normalized number of shortenings for each post title as a feature as well.

Stopwords

Stopwords represent the most common words in a particular language [1]. As specified in [7], the percentage of stopwords is expected to be higher in clickbait headlines than in non-clickbait headlines. The motivation is based on the fact that in clickbait headlines the stopwords need to complete the headline structure, whereas in non-clickbait more content words are used.

We performed this analysis with our dataset as well and our results indeed support the conclusion from the paper: we identified 33% of stopwords in the clickbait headlines, compared to 27% for the non-clickbait headlines. Thus, we considered the normalized number of stopwords for each post title as a feature.

Slang words

In [7] the authors discuss the fact that slang words are another type of words commonly present in clickbait headlines. They composed their own list of slang words to check against the headlines. In our analysis, we used the same list, as we contacted one of the authors who then provided us access to this slang words list. It is important to specify that from this list we eliminated 'RT' (which stands for 'retweet') as it was highly present in both clickbait and non-clickbait headlines. As a feature, we used the normalized number of slang words for each post title.

Sentiment features

Clickbait headlines are meant to spark the curiosity of the readers, hence they need to convey the information in a way that is capturing more sentiment, rather than neutral formulation. Considering this motivation, we generated the sentiment scores, negative and positive, as well as the polarity of each of the post titles. It is important to clarify that by polarity we define how non-neutral a post headline is, so we compute it as the complement of the neutral score.

In order to compute these sentiment scores we used VADER [14], which is part of NLTK library in Python[8]. VADER works by using a combination of opinion lexicons and rules to determine the positivity/negativity of a certain text instance. It quantifies the level of positivity, negativity, and neutrality in the sentence by assigning a score, rather than just classifying it in one of the categories.

Part-of-Speech Tagging

Although this was not one of the features used in the paper that we are reproducing, we performed Part-of-Speech tagging on both the Post Title and the Target Title for the two classes 'clickbait' and 'no-clickbait', using the standard POS tagging tool provided by the NLTK library[9]. As in [7], we found some suspiciously high counts for some of the tags. In particular, we found the **NNP** (Proper Noun) tag to be the highest one in both clickbait and non-clickbait posts.

When investigating the possible reasons behind this, we noticed that many titles are written using capital letters at the beginning of each word. As an example, we found the following title in the dataset: "*The Dude Who Followed His Girlfriend Around The World Just Photographed Their Wedding Perfectly*".

[6] `https://www.nltk.org/api/nltk.tokenize.html`

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

[8] `https://www.nltk.org/_modules/nltk/sentiment/vader.html`

[9] `https://www.nltk.org/book/ch05.html`

No Lowercase	The	Dude	Who	Followed	Christine	Around	The	World	Just	Photographed	Their	Wedding	Perfectly
	DT	**NNP**	**NNP**	VBD	NNP	IN	DT	**NNP**	**NNP**	**NNP**	**NNP**	**NNP**	RB
NER Lowercase	the	dude	who	followed	Christine	around	the	world	just	photographed	their	wedding	perfectly
	DT	**NN**	**WP**	VBD	NNP	IN	DT	**NN**	**RB**	**VBN**	**PRP$**	**NN**	RB

Table 1: POS tagging before and after NER lowercasing

To address this issue we followed the idea proposed in [7]: doing case folding by reducing all letters to lowercase while maintaining the original casing for named entities. In order to do this, we made use of the Stanford Entity Recognizer (NER)[10]. This tool is written in Java but the NLTK library provides a simple Python wrapper for it[11].

To understand how this preprocessing technique affects the Part-of-Speech tagging algorithm, we ran the following experiment:

1. Create a sentence similar to the ones found in the titles. We manually explored the dataset to find a title sentence (the one mentioned earlier in this section) and we replaced "*His Girlfriend*" with the named entity "*Christine*", yielding the final sentence: "*The Dude Who Followed Christine Around The World Just Photographed Their Wedding Perfectly*".
2. Run the NLTK POS tagger on the obtained sentence from point 1), without any preprocessing steps. Results are shown in the top part of Table 1.
3. Run the NLTK POS tagger after applying the proposed technique: First, we use the Stanford NER to detect named entities. Subsequently, we transform all the words except the ones detected as named entities by the NER to lowercase. In this case, we obtain the sentence: "*the dude who followed Christine around the world just photographed their wedding perfectly*". As said, this is the input for the NLTK POS tagger. Results are shown in the bottom part of Table 1.

The comparison shown in Table 1 indeed shows how the transformation of the original sentence led to more satisfactory results. For example, '**Just**', '**Photographed**', '**Their**', and '**Wedding**' are no longer tagged as proper nouns (NNP) but instead they are tagged as '**Adverb**', '**Verb Past Participle**', '**Possessive Pronoun**', '**Noun**', respectively.

We then computed the percentage of each tag in a certain title, and averaged over the different samples. Figure 1 shows the results for both clickbait and non-clickbait Post Titles, computed on approximately 2500 samples. It is possible to notice how certain tags show interesting differences across the two groups, motivating our choice to use these POS tags as features for the classification. A note must be made regarding the fact that this step is extremely time demanding, due to the NER tagger: more than 10 hours are required to tag the titles of the entire dataset of approximately 19000 samples.

There are 45 possible tags in total[12], and using all of them

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

[11] `https://www.nltk.org/api/nltk.tag.html#nltk.tag.stanford.StanfordTagger`

[12] `https://stackoverflow.com/questions/15388831/what-are-all-possible-pos-tags-of-nltk`

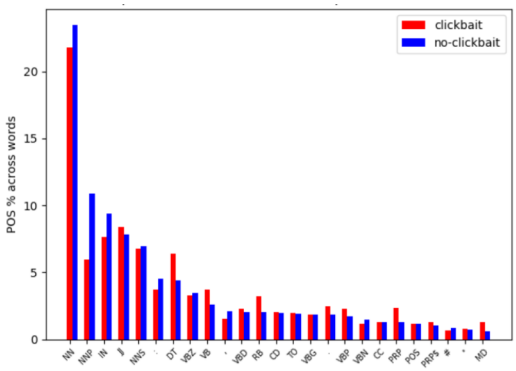

Figure 1: POS tagging - Post Titles - NER lowercase preprocessing

would mean having a total of 90 features (45 for the post text and 45 for the target title). However, we noticed that not all of these tags were useful features. For example, some of these tags (such as **LS**) were not found even once across the whole training set. Therefore, we used an estimation[13] of the mutual information between each one of these tags and the label of the training sample to retain the top 70% most important tags (after excluding the ones assuming a single unique value across the whole dataset). This value is an estimation of how much each of these features correlates with the target variable (which, in this case, is the true label): a high value of correlation suggests that such feature might be important for the classification. Also, this helps us reduce the final number of features, thus combating the well-known curse of dimensionality [15].

Post Text N-grams

For the purposes of this project, we considered N-grams as sequences of N contiguous words from each post text in the dataset. An analysis of the distribution of N-grams in a corpus can reveal some of the common patterns of phrases used throughout the texts and was also done in [7], as it is especially relevant for the case of clickbait detection due to clickbait posts using repetitive, "typical" phrases to catch the readers' attention. We therefore opted for N-gram analysis even though the original paper [10] did not do it.

For a preliminary analysis, we extracted all the 1-, 2-, 3-, and 4-grams from the entire corpus, including both the clickbait and non-clickbait posts. When extracting unigrams, we explicitly left out stopwords as they would have just polluted

[13] `https://scikit-learn.org/stable/modules/generated/sklearn.feature\_selection.mutual_info_classif.html`

the unigram collection; however, inspired by [7] we still included them in the analysis of the higher level N-grams as we thought they could reveal more informative, longer patterns of phrases which lack content words and tend to appear often in clickbait. Furthermore, to capture the general occurrences of digits, quotes, and hashtags, we replaced each matched digit, quote, and hashtag with $\langle d \rangle$, $\langle quote \rangle$, and $\langle hashtag \rangle$, respectively. We used frequency analysis to find a threshold determining the minimum number of occurrences required for an N-gram to be considered, finally retaining all the N-grams that appeared in the corpus at least 5 times.

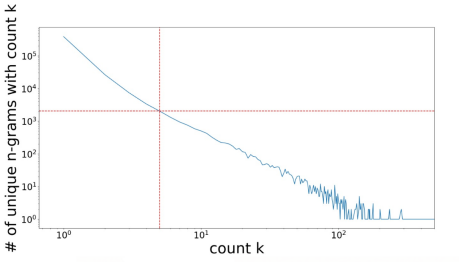

Figure 2: Frequency analysis of the n-grams

Looking at the N-gram distributions in Figure 2, it is possible to observe that there are a lot of unique N-grams occurring only for a small number of times (for example, there are 300K+ N-grams occurring only once). Since our underlying assumption was that an N-gram has to appear a sufficient number of times in order to be relevant for the classification, we selected the N-grams that occurred in the corpus at least 5 times, finally obtaining approximately 10.000 N-grams. Had we not pruned the N-gram list, we would have captured very rare, non-representative N-grams in the analysis as well and setting 5 as threshold gave us the possibility to reduce the number of computations and still have a reliable set for further analysis.

We then used the selected N-grams to build a post – N-gram matrix, counting the total number of occurrences of each N-gram in each of the corpus posts. Similar to our POS tag approach, we computed the estimates of mutual information between each of the N-grams and the true label, retaining the top 1% of the highest scoring N-grams. This left us with a final total of about 100 N-grams, which we used to compute a new set of features - the occurrence counts of each of the N-grams in the post text. Some notable examples of the highest scoring N-grams based on mutual information included: "$\langle d \rangle$ *things*", "*top* $\langle hashtag \rangle$ *stories*", "*heres what*", "*you need to*". Some highly specific N-grams appeared among the selected 100 as a result of the Clickbait Challenge dataset being biased towards certain (temporally relevant) topics; for example, the top scoring N-gram was "*trump*". For the purposes of this project, we limited ourselves to detecting clickbait in the context of this dataset and this period of time. If we wanted to generalize our features past the context of the Clickbait Challenge, however, we probably would have had to (manually) filter such biased N-grams out and also consider introducing an upper threshold in terms of mutual information scores.

Formal / Informal Words

Although we agree on the motivation given in the reproduced paper as to why formal and informal words might be useful for clickbait detection, we do not agree on the method with which the authors decided to engineer such a feature. In particular, they used PyDictionary to determine whether or not a particular word is formal by simply checking if that word was found in that dictionary. We tested the sentence "*we dunno how to do this*" and the result with using PyDictionary is as follows:

- The words found in the dictionary (and thus **formal**) are: *do*.
- The remaining words (*we - dunno - how - to - this*) are not found, and thus they would be counted as **informal**.

As far as we are aware, there is no rigorous way of determining whether a word is formal or informal in a programmatic fashion. Therefore, we implemented our own algorithm: (1) transform the sentence to lowercase, (2) tokenize the sentence removing the punctuation[14], (3) lemmatize the tokens using the NLTK WordNet Lemmatizer, (4) for each token, if it is found in the NLTK 'words' corpus, then count it as **formal**, otherwise **informal**. Using this algorithm on the above-mentioned sentence, we classify all the words as **formal** except for the word '*dunno*', which is classified as **informal**.

By analysing the percentage of formal words in clickbait and non-clickbait Post Titles we found average percentages of 88.2% and 82.4%, respectively. This shows that, even though our method seemed to work on the specific example of "*we dunno how to do this*", the result does not confirm the motivation which led us to compute this feature: instead of having a higher percentage of "informal" words in clickbait titles, it seems clickbait titles have more "formal" words. However, this might have been caused by our algorithm, and it does not necessarily imply the initial motivation was wrong.

Image Presence - Text in Image

In the paper we reproduced, the authors used the information about images related to the post to generate two features. Inspired by previous work (in [9] and [8]), they authors (i) use the image presence as a binary feature, and (ii) the presence of text[15] in such image as another binary feature. However, after having conducted our own analysis on the data, we found unsatisfactory results: 63% of clickbait posts have an image and 67% of non-clickbait posts have an image; 13% of clickbait related images have text[16] and 14% of non-clickbait related images have text.

This shows that, as far as our dataset is concerned, having an image (or text in it) is not a discriminative feature which can

[14] `http://www.nltk.org/api/nltk.tokenize.html?highlight=regexp#module-nltk.tokenize.regexp`

[15] By means of Tesseract: `https://opensource.google.com/projects/tesseract`

[16] We applied a threshold of 10 characters, as a noise-filtering parameter: Tesseract OCR sometimes generates string characters as output although the image does not have any text in it.

help differentiate between clickbait and non-clickbait posts. Therefore, we did not employ these features for the classification.

EXPERIMENTS

Machine Learning Algorithms and Evaluation Metric

Our goal in this research was to analyze whether is possible to classify clickbait and non-clickbait articles. Based on the features that we have previously described, we experimented with the following classifiers: Naive Bayes, Maximum Entropy, Random Forest, as well as AdaBoost.

When it comes to **Naive Bayes**, we are aware of its simplicity and that its assumption of independence between features does not hold in real situations. However, it has proven to work well when it comes to text categorization [17], so we decided to include it in our experiments.

Just like Naive Bayes, the **Maximum Entropy** classifier belongs to the family of probabilistic models. However, unlike Naive Bayes, it does not assume that the features are conditionally independent of each other. This is definitely true for text-based classification, which is the case we are analyzing. Additionally, when it comes to Natural Language Processing applications, Maximum Entropy has been used in literature [3] [17]. Based on these motivations, we decided to include it in our experiments.

Regarding **Random Forest**, it is an ensemble classifier based on decision trees. We think it is appropriate for our defined problem because many of our features are discrete (that is, we check, measure and count the presence of certain properties, as presented in the Feature Extraction section). We can imagine that finding a good threshold for these features, especially in the binary case that we are considering, can lead to promising results. Since this is the philosophy behind the way decision trees work, and since during the training phase they are computing the right features and thresholds to define the separation between the two classes, we considered that the Random Forest classifier was a good fit for our problem. However, if only a small number of features are proven to be relevant for determining the outcome, Random Forest might result in a rather weak classifier. That is why we also considered **AdaBoost**, which can be seen as an improved version of Random Forest - it adapts in order to weight relevant predictors by tweaking the subsequent learners in the favor of the previously misclassified instances.

Regarding training, we have split the data in 80% training set and 20% testing set. We performed **Grid Search** on the 80% training set in order to optimize the parameters of each of the chosen classifiers. Grid Search performs an exhaustive search over the parameter values for each classifier and evaluates them, in our case, using stratified k-fold cross validation. It returns the parameters that perform the best given a chosen metric. For the parameter values we have performed Grid Search on and for details regarding the classifiers, refer to the Appendix. It is important to note that we decided to use stratified k-fold cross validation instead of the basic k-fold cross validation because we have observed that our dataset was highly imbalanced. Stratified k-fold cross validation preserves the percentages of samples for each class in the original data. Since we wanted to train the models for the actual, real situation in our data, we decided to employ this approach.

When it comes to the evaluation metric, we argue that accuracy is not a reliable metric to use, given the high class imbalance of the dataset. On the other hand, metrics such as precision, recall, and F1 are more useful in such setting. We decided to employ **precision**. This was the metric based on which the optimization with Grid Search was performed as well. We based this choice on the decision that in the case of our classification, False Positives, FP, (classifying as clickbait articles that are actually non-clickbait) are a worse case than False Negatives, FN, (classifying non-clickbait articles that are actually clickbait). Since a browser extension for blocking clickbait articles could be implemented based on this classification, we think it is better to still show users content even if there is a chance of it being clickbait (FN), rather than hiding relevant content from them (FP). Since the formula of the precision is

$$Precision = \frac{TP}{TP+FP}$$ (where TP = True Positives),

we can see that it is maximized when the value of FP is minimized.

Once the best parameters for each estimator were obtained using Grid Search, we tested the tuned classifiers on the remaining 20% test set. In addition to precision, we also calculated the Mean Square Error (MSE) using the mean judgment of the annotators and the soft probabilities predicted by our models. We decided to calculate this metric as well because it was used to evaluate the submitted models in the Clickbait Challenge [17] and we wanted to have, as much as possible, an accurate comparison between our models and the ones submitted to the competition. We are aware that this comparison is not completely perfect however, in the sense that we did not evaluate the models on the true test dataset that was kept hidden by the organizers, but on a part of the training set held aside by us.

RESULTS

Classifier	Acc	Prec	Rec	F1	MSE
Random Forest	0.824	0.772	0.397	0.525	**0.044**
MaxEntropy	**0.832**	0.726	**0.501**	**0.593**	0.049
Naive Bayes	0.800	0.651	0.387	0.486	0.131
AdaBoost	0.795	**0.802**	0.217	0.341	0.060

Table 2: Comparison of the classifiers used

Table 2 shows the results of the experiment for the selected classifiers. As said, these values are obtained by testing the classifiers on the held-out fraction of the dataset (20% of the original dataset). We did not use this test set for the hyperparameter tuning of the classifiers nor any other test.

Since the Mean Squared Error (**MSE**) is the official metric used to rank models participating the Clickbait Challenge

[17] `https://www.clickbait-challenge.org/`

2017, we decided to further analyze the results obtained with the classifier with the lowest **MSE**: Random Forest. Although we are aware of the fact that we did not test our classifier on the TIRA machine, it is important to remember that we left out almost 4000 samples for testing purposes, which could have been otherwise used as part of the training set. Therefore, we believe the results we get are to some extent comparable with the ones published in [18]. In particular, we match the value of **MSE** of the baseline adopted for such challenge.

In order to get a better insight into the results, we have computed and plotted the confusion matrix, which is shown in Figure 3. Having been pointed out multiple times in this report already, the class imbalance can be observed also by looking at the confusion matrix: indeed, also in the test set we have clickbait and non-clickbait posts at percentages of 25% and 75%, respectively.

Since our setting is a binary classification task with highly imbalanced classes, if features were not discriminative the classifier would end up classifying samples as **non-clickbait** 100% of the time, in order to minimize the classification error (non-clickbait is the class with the biggest percentage in the dataset). However, by looking at the confusion matrix in Figure 3 we can observe the number of True Positives and False Positives being 380 and 112, respectively. This shows that the features we engineered are actually helpful for the classifier to partially distinguish between the two classes. For further investigation, we also wanted to verify the change in our classifier in an experiment with balanced classes.

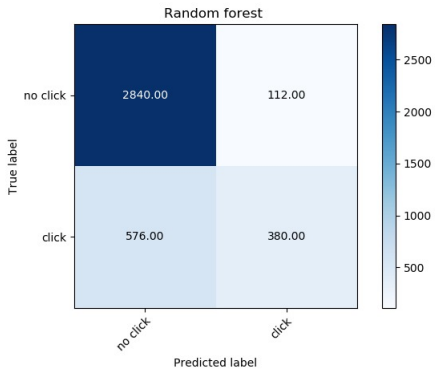

Figure 3: Confusion Matrix for the Random Forest classifier

Class-balanced setting

We downsampled the number of non-clickbait posts in our dataset, in order to match the number of clickbait posts, thus obtaining a perfectly balanced dataset. Precisely, we obtained 4761 samples per class. Again, we trained on 80% of the balanced dataset and tested on the remaining 20%. The results we obtain are shown in table 3, followed by the confusion matrix in figure 4.

In the confusion matrix we can observe that the complementary categories (TP with TN and FP with FN, respectively)

Classifier	Acc	Prec	Rec	F1	MSE
Random Forest	0.9	0.765	0.762	0.764	0.048

Table 3: Results for the balanced dataset

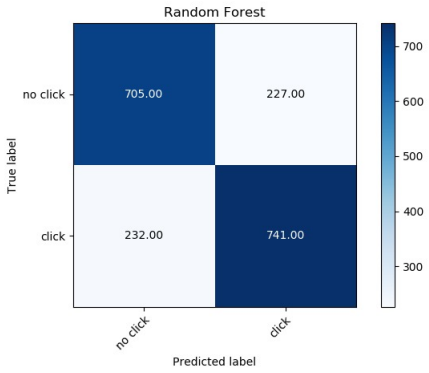

Figure 4: Confusion Matrix for the Random Forest classifier with the balanced dataset

have very close values. This situation shows that the classifier has been trained in a balanced manner when differentiating between clickbait and non-clickbait. This is especially reflected through the fact that the number of FP and FN is almost equal, implying that the classifier is not biased in predicting any of the categories more when misclassifying. However, it clearly still has some problems in differentiating between clickbait and non-clickbait in some of the situations. We think this could be caused by some specific characteristics of the two classes that we did not capture by means of the features we have selected. Yet, to some extent this result can also be related to conflicting annotations, as we will discuss in the next section.

Error analysis

In order to understand the failure cases of our classifier, we have performed a manual analysis of the misclassified instances. We analyzed the truth judgments provided by the human annotators and we found many instances with conflicting annotations. Further, we computed the means and standard deviations of the human judgments for the misclassified instances and we observed a mean of **0.59** and a standard deviation of **0.17**. From these results we can infer that, in most of the cases, the classifier was unsure of the nature of the post (clickbait or non-clickbait) when human annotators made conflicting judgments as well. This observation is also supported by the fact that the same measures computed on the whole dataset revealed a different scenario: a mean of **0.32** and a standard deviation of **0.25**. Considering these findings, we believe that there is a connection between the behavior of our classifier and the decisions made by the human annotators. By looking at the complete data, we observed that it is not generally the case that human annotators strongly disagree. However, when that happens, the classifier seems to perform worse as well, perhaps showing a not completely clear distinction between clickbait and non-clickbait news. Additionally, it is important to also take into account

that only five annotators per sample were responsible for the class labels. Perhaps, having more human annotations would help us draw a more reliable conclusion.

Feature Importance Scores

In order to get a better insight regarding the features that have the most impact when classifying clickbait articles, we have computed the feature importance scores using the Random Forest classifier. The results for the top 10 most important features for the classification are presented in table 4.

Feature name	Importance score
No. of characters - post title	0.056317
NNP Tag - post title	0.049675
Stopwords count - post title	0.046521
No. of characters - article description	0.038956
No. of words - post title	0.038809
Average word length - post title	0.037134
No. of characters - the article captions	0.034948
NNP tag - target article	0.033673
No. of characters - article paragraphs	0.033261
No. of characters - article title	0.031053

Table 4: Feature importance scores computed using the Random Forest classifier

We find it surprising that the most important feature when classifying between clickbait and non-clickbait is the number of characters in the post title. This is one of the features that we reproduced from the original paper. Intuitively, we would expect the number of characters to be relevant, but not at such a significant level. Rather, we would expect the difference in the post title structure for the clickbait and non-clickbait articles to be reflected through more structural or content-oriented features. However, we are aware and taking into account the possibility that this might be a particular characteristic of the specific data we have worked with (data extracted from social media). It might be the case that this does not generalize to data from other social media or news platforms.

Additionally, we can observe that the second most important feature is the number of proper nouns (**NNP** tag) in the post title. This is a sensible result and it matches our previous feature analysis, visualized in figure 1. We can observe that the **NNP** POS tag is the one with the highest discrepancy between clickbait and non-clickbait articles, the percentage of proper nouns present in non-clickbait articles being approximately 5% higher than the one present in clickbait articles. Thus, it can be expected that it would have such a high impact on differentiation between the two classes.

The third most important feature for the classification is the number of stopwords in the post title. As already discussed in the Feature Extraction section, it is expected that the clickbait titles will have more stopwords, as they are required for completing the structures of the titles. On the other hand, since the non-clickbait titles contain more content words, a high number of stopwords is not required anymore.

Regarding the remaining top 10 most important features, we can see that most of them are counts of words or characters. Additionally, we can observe the importance of the proper nouns once again, in this case for the content of the target article.

CONCLUSIONS

Clickbait news detection revealed itself as an interesting topic of research. As we observed in the Clickbait Challenge 2017 submissions, there are different approaches the teams adopted to solve this classification problem. In this paper, we reproduced part of the work done in [10], but also developed our own approach and methodologies for solving this task. Unfortunately, because of time constraints, we did not manage to officially test our work using the truly held-out test set of the Clickbait Challenge. However, we did our best to assess the performance of our classifier and compare it with the published rankings of the challenge [18] in a fair way. We obtained a Mean Square Error of 0.044 using Random Forest as classifier. This value matched with the published baseline for such challenge.

Future work might include the usage of tools specifically tuned for the Web, which is where Clickbait articles are found. As an example, it would be interesting to investigate whether using POS taggers developed for the social networks [12] would improve our performance. Based on our findings, it seems there is still room for improvement and therefore additional features would need to be engineered which are more discriminative between clickbait and non-clickbait articles.

REFERENCES

1. Stop words. `https://en.wikipedia.org/wiki/Stop_words`. Accessed: 2019-04-02.

2. Agrawal, A. Clickbait detection using deep learning. In *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*, IEEE (2016), 268–272.

3. Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. A maximum entropy approach to natural language processing. *Computational linguistics 22*, 1 (1996), 39–71.

4. Biyani, P., Tsioutsiouliklis, K., and Blackmer, J. " 8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality. In *Thirtieth AAAI Conference on Artificial Intelligence* (2016).

5. Blom, J. N., and Hansen, K. R. Click bait: Forward-reference as lure in online news headlines. *Journal of Pragmatics 76* (2015), 87–100.

6. Bourgonje, P., Schneider, J. M., and Rehm, G. From clickbait to fake news detection: an approach based on detecting the stance of headlines to articles. In *Proceedings of the 2017 EMNLP Workshop: Natural Language Processing meets Journalism* (2017), 84–89.

7. Chakraborty, A., Paranjape, B., Kakarla, S., and Ganguly, N. 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)*, IEEE (2016), 9–16.
8. Chen, Y., Conroy, N. J., and Rubin, V. L. Misleading online content: Recognizing clickbait as false news. In *Proceedings of the 2015 ACM on Workshop on Multimodal Deception Detection*, ACM (2015), 15–19.
9. Ecker, U. K., Lewandowsky, S., Chang, E. P., and Pillai, R. The effects of subtle misinformation in news headlines. *Journal of experimental psychology: applied 20*, 4 (2014), 323.
10. Elyashar, A., Bendahan, J., and Puzis, R. Detecting clickbait in online social media: You won't believe how we did it. *arXiv preprint arXiv:1710.06699* (2017).
11. Finkel, J. R., Grenager, T., and Manning, C. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, Association for Computational Linguistics (2005), 363–370.
12. Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., and Smith, N. A. Part-of-speech tagging for twitter: Annotation, features, and experiments. Tech. rep., Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2010.
13. Ha, Y., Kim, J., Won, D., Cha, M., and Joo, J. Characterizing clickbaits on instagram. In *Twelfth International AAAI Conference on Web and Social Media* (2018).
14. Hutto, C. J., and Gilbert, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media* (2014).
15. Keogh, E., and Mueen, A. *Curse of Dimensionality*. Springer US, Boston, MA, 2017, 314–315.
16. Lin, D., et al. An information-theoretic definition of similarity. In *Icml*, vol. 98, Citeseer (1998), 296–304.
17. Pang, B., Lee, L., and Vaithyanathan, S. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, Association for Computational Linguistics (2002), 79–86.
18. Potthast, M., Gollub, T., Hagen, M., and Stein, B. The clickbait challenge 2017: towards a regression model for clickbait strength. *arXiv preprint arXiv:1812.10847* (2018).
19. Potthast, M., Köpsel, S., Stein, B., and Hagen, M. Clickbait detection. In *European Conference on Information Retrieval*, Springer (2016), 810–817.
20. Resnik, P. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of artificial intelligence research 11* (1999), 95–130.
21. Zannettou, S., Chatzis, S., Papadamou, K., and Sirivianos, M. The good, the bad and the bait: Detecting and characterizing clickbait on youtube. In *2018 IEEE Security and Privacy Workshops (SPW)*, IEEE (2018), 63–69.

APPENDIX

Packages and Tools Version

We list the packages and tools we used, together with their version number.

- NLTK: version 3.4
- Stanford Named Entity Recognizer: version 3.9.2
- PyDictionary: version 1.5.2
- Vadersentiment: version 3.2.1
- Numpy: version 1.16.2
- Scikit-learn: version 0.20.1
- Pandas: version 0.24.1
- Pytesseract: version 0.2.6
- Tesseract OCR: version 3.05.01
- Matplotlib: version 3.0.2

Grid Search Parameters

For each of the classifiers applied during our experiments, we present the parameters that were tested during Grid Search.

Moreover, for each of the parameters of the classifiers, an explanation will be provided.

It is important to specify that for the Naive Bayes classifier there were no parameters to tune. The prior probabilities were automatically computed according to the distribution of the data. Thus, the parameters were searched and presented for Maximum Entropy (table 5), Random Forest (table 6) and AdaBoost (table).

Set of parameters Maximum Entropy Classifier

Parameter	Set of values
'penalty'	'l1', 'l2'

Table 5: Set of tested parameters for the Maximum Entropy Classifier

Parameters:

- **penalty**: the type of norm that is used for penalization

Set of parameters Random Forest Classifier

Parameters:

- **n_estimators**: the number of decision trees composing the Random Forest
- **max_depth**: The maximum depth of the decision trees.
- **max_features**: The maximum number of features to consider when assessing the best split for each node.
- **criterion**: The criterion measured when calculating the quality of a split at a node in the decision tress.

Parameter	Set of values
n_estimators	100, 200, 400
max_depth	100, 125, 150, 175, 200
max_features	'auto', 'sqrt', 'log2'
criterion	'gini', 'entropy', None

Table 6: Set of tested parameters for the Random Forest Classifier

Set of parameters AdaBoost Classifier

Parameters:

- **n_estimators**: the number of estimators when boosting is terminated
- **learning_rate**: The values by which the contribution of each classifier is shrinked.

Parameter	Set of values
n_estimators	50, 100, 150
learning_rate	0.01,0.05,0.1,0.3,1

Table 7: Set of tested parameters for the AdaBoost Classifier

Generated Features Overview

In table 8 an overview of the types of features that were generated, as well as their count is presented. We did not specify each feature individually, but grouped them per category. For example, in the case of the POS tags, 24 of them are from the post text, while the remaining 21 are from the target title.

Feature type	Number of features
POS tags	45
Formal words	2
Similarity Lin	1
Sentiment features	3
Stopwords	1
Short versions of words	1
Slang words	1
N-grams	101
Characters count	6
Average word length	2
Capital letters	1
Number of words	1
Common phrases	1

Table 8: Features overview