Heuristic Feature Selection for Clickbait Scoring

The Icarfish Clickbait Detector at the Clickbait Challenge 2017

Matti Wiegmann Michael Völske Benno Stein Matthias Hagen Martin Potthast

Bauhaus-Universität Weimar and Leipzig University
<first name>.<last name>@uni-weimar.de and martin.potthast@uni-leipzig.de

ABSTRACT

We study feature selection as a means to optimize the baseline clickbait detector employed at the Clickbait Challenge 2017 [6]. The challenge's task is to score the "clickbaitiness" of a given Twitter tweet on a scale from 0 (no clickbait) to 1 (strong clickbait). Unlike most other approaches submitted to the challenge, the baseline approach is based on manual feature engineering and does not compete out of the box with many of the deep learning-based approaches. We show that scaling up feature selection efforts to heuristically identify better-performing feature subsets catapults the performance of the baseline classifier to second rank overall, beating 12 other competing approaches and improving over the baseline performance by 10%. This demonstrates that traditional classification approaches can still keep up with deep learning on this task.

1. INTRODUCTION

Because of the wide-spread usage of social media in politics, online news publishing, and public relations, the detection of clickbait, hate speech, and fake news has been a topic of increasing relevance, also from a research perspective. Given the high frequency with which new posts are spread on social media, their publishers compete for user attention: a hunt for clicks. This competition incentivizes the use of clickbait headlines or clickbaiting language to generate page visits. Especially in the news domain, publishers seem to balance a certain degree of clickbaitiness against perceived credibility. This balance is a central problem for automated detection: since there is no closed definition that distinguishes clickbait headlines from others, regression approaches are developed to determine the degree of "clickbaitiness."

Preceding the Clickbait Challenge 2017, Potthast et al. [5] suggested an approach for binary clickbait classification using a range of different features at the tweet and at the document level. We reimplemented a regression model version in Python as close as possible to the full feature set of the original publication to serve as a baseline for the Clickbait Challenge 2017.[1] This baseline achieved the 7th rank among 13 participants (cf. Table 2). In their original study, Potthast et al. could increase the performance by 10% when selecting the 1,000 best features (χ^2 selction) from the full feature set. But simply adapting the same feature selection for the regression task of the Clickbait Challenge 2017 did not yield the same improvement, since χ^2 feature selection does not seem to work as well for scoring clickbaitiness as it does for classification.

The performance of many machine learning models heavily relies on a good selection of features since the features determine the information that can be used by the learning algorithm. Features can be discriminative (reducing the prediction error), confusing (increasing the prediction error), or redundant (no direct impact). Also feature combinations can have similar properties: they can be supportive if their combination helps to discriminate better, or they can be inhibiting if their combination causes confusion. Therefore, a good feature selection approach should identify confusing features and inhibiting combinations, and reduce redundancy to avoid the curse of dimensionality.

Guyon and Elisseeff [2] distinguish three basic strategies for feature selection:

- *Filter methods* estimate the value of a feature by statistical analyses like correlation, mutual information, etc. Such techniques are effective, if the data supports such analyses.
- *Wrapper methods* are meta-heuristics that apply a search strategy to find a subset and evaluate it by training a model (e.g., simulated annealing or genetic algorithms with the prediction error as their fitness function).
- *Embedding methods* integrate filter methods within a machine learning algorithm (e.g., random forest regression).

Since the Clickbait Challenge 2017 data is rather sparse, filter methods and embedding methods are not very effective: Statistics like correlation or mutual information cannot provide meaningful information for sparse data as indicated by the rather high prediction errors of a lasso regression (considering covariance to select features) or a random forest regression (building decision trees) shown in Figure 1. Heuristic wrapper methods are thus the only alternatives left, incurring a comparably higher computational complexity. In this paper, we demonstrate how to improve the performance of the Clickbait Challenge 2017 baseline via such heuristic feature selection.

2. RELATED WORK

As already mentioned, our approach is a derivation of the clickbait detection classifier developed by Potthast et al. [5]. This approach utilized 215 different features and feature types, grouped into the following 5 categories: tweet-based features (e.g., character and word n-grams), Downworthy rule sets, General Inquirer word lists, features based on the linked web page, and meta information (e.g., publisher). It was trained on the Webis Clickbait 2016 corpus [5], a manually annotated 3000 tweet corpus constructed alongside the classifier, and we reimplemented it as a baseline for the Clickbait Challenge 2017 as well as retrained on the challenge's training dataset.

The training dataset of the Clickbait Challenge 2017 comprises nearly 20,000 tweets, each annotated by at least 5 people recruited at Amazon's Mechanical Turk [7]. Another 20,000 tweets have been annotated the same way, but withheld for testing.

[1] See https://github.com/clickbait-challenge/clickbait17-baseline.

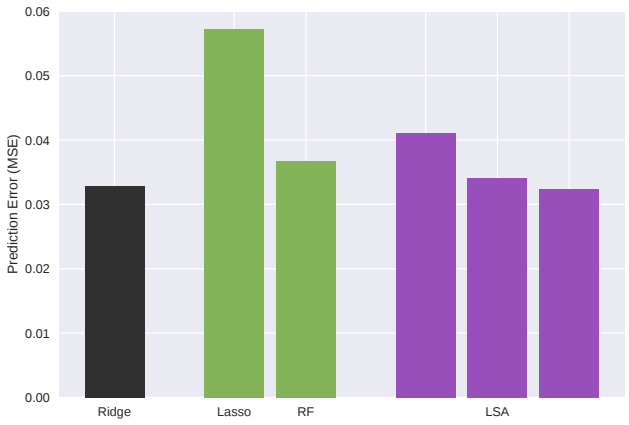

Figure 1: Comparison of prediction errors of different out-of-the-box feature selection techniques. Shown are ridge regression (black), lasso regression (green), and random forest regression (green) on the full feature set, and latent semantic analysis as dimensionality reduction followed by ridge regression with 150, 300, and 600 categories (purple, left to right).

A complete overview of all other approaches submitted to the Clickbait Challenge 2017 can be found in [6], which also includes a review of other related work published elsewhere. We therefore omit a further review of related work here.

3. ICARFISH CLICKBAIT DETECTOR

The approach of Potthast et al. [5] was reimplemented to serve as baseline for the Clickbait Challenge. In our reimplementation, if necessary for a given feature, tweets are pre-processed using NLTK's WordNetLemmatizer and TweetTokenizer.[2] Some of the originally proposed features were omitted to render the approach compatible with the challenge's training dataset and the TIRA evaluation platform [4]. For instance, the publisher and retweet information originally used are not present in the challenge's datasets, and the Imagga image tagging service used cannot be reached from within the challenge's evaluation platform. Furthermore, the Downworthy clickbait rules were omitted, since they are redundant.

3.1 Features

The following feature types are used:

1. *Tweet character n-grams*. All character 1-, 2-, and 3-grams that occur more than twice, resulting in a total of 12,471 distinct character n-gram features weighted by tf-idf. We believe that n-grams occurring only once or twice within the 19,538 tweets of the training dataset are prone to overfitting.

2. *Tweet word n-grams*. All word 1-, 2-, and 3-grams that occur more than twice, resulting in a total of 24,861 distinct word n-gram features weighted by tf-idf.

3. *Engineered features*. In total, twelve engineered features are implemented. Three features encode character counts: average word length, length of the longest word, and total character length. Five features encode character occurrences: the occurrence frequency of '@' (mentions), '#' (hashtags), and '.' (dots), whether a tweet starts with a number, and the occurrence frequency of abbreviations following the Oxford abbreviations list. Two features encode meta-data, namely whether the tweet has media attachments and the part of day (as quarters of a day, 1 to 4) the tweet was issued. The last two features encode the tweet's sentiment polarity as per the VADER sentiment detector implementation in NLTK [3], and the Flesh-Kincait readability score.

4. *Word lists*. Using 181 General Inquirer word lists,[3] the Terrier stop word list,[4] the Dale-Chall easy words list [1], and the Downworthy common clickbait phrases,[5] each word list is used as a feature indicating whether a word form the list occurs in a tweet, and how often.

Altogether, we obtain 37,528 distinct features. Note that we do not use any features from the web pages linked in a tweet to reduce the complexity of this initial study, to speed up the experimentation process, and for practical reasons: a clickbait scorer which does not have to download the linked web page of a teaser message is more scalable than one which does.

3.2 Feature Selection Background

The fitness function of heuristic feature selection approaches is typically the prediction error; in case of the Clickbait Challenge 2017, the mean squared error between the predicted clickbaitiness scores and the ones in the ground-truth is used. In our study, we use the mean squared error as the fitness function for training a ridge regression model,[6] since it is a fast linear model that does not preselect features based on their properties (like lasso, elasticnet, and decision tree regressors), and since it provides better results using the full feature set compared to a linear support vector regression or a gradient descent regression.

Several local search algorithms and combinatorial optimization algorithms can find good approximations of the "optimal" feature subset (e.g., simulated annealing, genetic algorithms, and ant colony optimization). The major downside is that they usually identify a good subset but give no information about how good the features are and how they interact. Determining an individual feature's quality and taking into account feature relationships will yield more insights about the task at hand and might even help to engineer better features. We therefore do not employ simulated annealing, genetic algorithms, or ant colony optimization in our study.

As noted earlier, feature selection can significantly increase the performance of predictive models by generalizing and reducing computational load for further experiments and analysis. Established strategies usually use statistics such as χ^2 or correlations between a feature and the target vector. When working on really short texts, such as social media posts or comments, all features based on occurrence frequencies (like n-grams or word lists) become really sparse. These sparse vectors on their own do not contain a lot of information, since most entries are zero, and therefore the statistical metrics become indifferent. This can be seen well when considering the performance of learning algorithms that fit using these metrics (like lasso or elasticnet). This also implies that a feature selection strategy is needed which considers combinations (or interactions) of features.

Considering that statistical metrics are not helpful, applying a search strategy is the logical next step. These strategies add and/or remove features at each iteration, based on how the prediction error changes. Established strategies are:

[2]We used the NLTK 3.2.4 library for python

[3]http://wjh.harvard.edu/~inquirer/

[4]https://github.com/terrier-org/terrier-core/blob/4.2/share/stopword-list.txt

[5]https://github.com/snipe/downworthy/blob/master/Source/dictionaries/original.js

[6]We used the scikit-learn 0.18.1 library and Python 3.5.2.

- *Exhaustive Search*, which is exponential in the number of features.
- *Forward Selection*, which does not find interactions.
- *Compound Selection*, which adds k features and removes r, $k > r$. This strategy is computationally feasible for some hyper-parameters and does find interactions. If these interactions involve many features and if multiple interactions are considered (which is very likely for sparse text data), k has to become very large (to capture interactions and change the prediction error in a meaningful way) and r has to become dynamic (large if many redundant features are added, small if a useful interaction was added). This changes the compound selection to be approximately equal to the next strategy.
- *Backward Selection* starts with a full feature set and iteratively removes features that change the prediction error below a certain threshold. This strategy is linear in the number of features but does not find multiple interactions.

None of these strategies is natively perfect, but backward selection is a good starting point once modified to account for the aforementioned specifics of the data. If one feature contributes equally to confusing as well as discriminatory interactions (which is to be expected given the data), removing it would not change the prediction error. To reliably resolve those interactions, it is necessary to remove features in a way most confusing interactions are resolved and most discriminatory ones are still intact. This requires a way to judge each feature's influence on the whole set.

The most promising strategy to determine the value of a feature is the leave-one-feature-out error minimization strategy [8]. This strategy compares the prediction error (i.e. MSE) of two models. The first one trains and predicts on all features. The second one trains and predicts on all all features except the one whose value is to be determined. The difference between both models (current MSE minus old MSE) hints at the feature's contribution in the prediction. Here, a positive value indicates that, if a feature is removed, the prediction error rises and vice versa. The idea of this study is to extend backward selection with leave-one-feature-out error minimization to determine reliable feature values.

To infer useful information about the value of a feature from its leave-one-out error, it is necessary to first remove redundancy and multiple interactions. The primary assumption is that smaller subsets contain less problematic redundancies and interactions and the leave-one-out error over a subset of features is more clear. Considering the inherent ambiguity of language, it is likely unnecessary to use a strategy to construct a subset. It is also likely that, if the subsets are to small, to much information about the interactions is removed. Since it is not possible to infer useful subsets for leave-one-out, parallel backward selection is used to determine the subsets, resulting in the leave-many-out feature selection strategy.

3.3 Leave-Many-Out Feature Selection

The leave-many-out strategy performs a backward selection search: starting with the full feature set, features are randomly removed one at a time until a minimum subset size is reached. For each removed feature, the leave-one-out error is recorded. This procedure is repeated many times, resulting in a series of leave-one-out errors for each feature. The average over all errors recorded for a given feature can be considered a score of its overall usefulness. This score captures how frequently and by how much the removal of a feature improves or reduces the prediction error, or if its removal has no impact. The score also reflects if a feature is more important for some interactions, or if it is redundant. Finally, ranking features by their scores and selecting the top k ones will result in a highly discriminatory subset of features.

3.4 Hadoop-based Implementation

Our feature selection approach is implemented within the Hadoop framework,[7] consisting of preprocessing, mappers, and reducers. Preprocessing prepares all data that only need to be computed once, the mappers direct the reducers by passing feature subset changes to them, and the reducers conduct the train-validate steps in parallel on given subsets, recording the MSE changes.

The preprocessing step includes parsing the provided data set, deriving the aforementioned features into a feature matrix, as well as filling a vector with the ground-truth clickbaitiness scores. This data was stored as two npz-files (numpys data storage format). Afterwards, the map reduce input file is generated. This file dictates how many models (1000 in our case) are analyzed in parallel: each line consists of a unique ID and the size of the full feature set (37528). Both npz-files and the input file are then fed into a Hadoop-streaming job.

The mapper reads the input file one line at a time, generating a bit set of the given length with all bits set to 1, where the i-th bit corresponds to the i-th feature, its truth value indicating whether the feature is to be included in a to-be-analyzed feature subset. It then repeatedly (1000 repetitions in our case) flips one of the 1-bits to 0, passing the bit set resulting from each bit flip to a reducer, keyed by the line's ID. Altogether, for each of the 1000 input lines, our mapper emitted 1000 bit sets to the reducers, ordered by ID and progressive feature removals, resulting in an average 26 removals of each feature overall.

We adjusted the Hadoop job to spawn one reducer per line ID. Since reducers are typically spawned only once per line ID, they need to initialize the regression model and the supplied npz-files only once before handling the series of bit sets for a given ID. To ensure generalizability and to foreclose overfitting, each reducer randomly splits the training data into 70:30 train-test sets. It then proceeds to handle the bit sets one at a time, computing the MSE of the regression model after removing the respective features indicated by a bit set, and emitting the recorded prediction error along the bit set as a reference which feature subset was used to compute it, keyed by the input line's ID.

4. EXPERIMENTAL RESULTS

We did conduct a large-scale experiment to select a "best" feature subset that should then be tested on the Clickbait Challenge 2017 test data and then further analyzed the individual features to gain insights about the impact of the best and worst features.

4.1 Selecting the "Best" Feature Subset

In total, we processed one million remove-train-predict iterations for our feature assessment (i.e., an average of 26 subsets a feature was part of). After all the outputs of these iterations are accumulated, the average error change per feature for the subsets a feature was part of is computed and used to rank the features. From this ranking, we then choose feature subsets to examine the effect of removing the lowest ranking features and training a model on the remaining ones. We constructed 49 subsets including the top-ranked 100%, 98%, 96%, ..., 2% of the features (i.e., steps of 2%) and three other subsets containing only the top-ranked 1.5%, 1.0%, and 0.5% of the features. For every subset, we trained a ridge regression model on 2/3 of the training data and validated it on the remaining 1/3 of the

[7] We employ the Hadoop-streaming library in version 2.7.2; the code is available at https://github.com/clickbait-challenge/icarfish

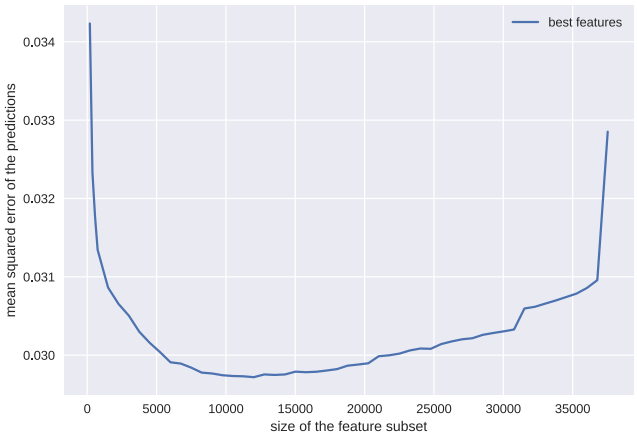

Figure 2: Prediction error of the trained ridge regression models for the 52 examined feature subsets on a constant 2/3-1/3 train-validate-split of the Clickbait Challenge 2017 training data.

Table 1: Prediction error of some selected models on a constant 2/3-1/3 train-validate-split of the Clickbait Challenge 2017 training data.

Feature Subset Size (absolute)	(percentage)	Mean Squared Error
37,528	100.0%	0.0328
12,008	32.0%	0.0297
375	1.0%	0.0323
187	0.5%	0.0342

training data (the split was kept constant for all the 52 feature subsets). The results on the 1/3 of the training data used for validation are plotted in Figure 2.

The best performing model in this experiment series achieves a prediction error of 0.0297 with 12,008 features (32.0% of the original features). This corresponds to an about 10% performance improvement over the full feature set (cf. Table 1). The smallest subset size that does not perform worse than the full feature set uses 375 features (1.0%).

We choose the performance-wise best among the validated models as our final approach, tested it after the challenge finished, and achieved a prediction error of 0.0351 on the Clickbait Challenge 2017 test data (participant icarfish in Table 2). This means that icarfish with its heuristic feature selection achieves a performance gain of about 20% over the baseline (that is identical except using the full feature set) and would have been placed second in the competition.

4.2 Impact Analysis

To gain more insights on the different feature subsets, we did further analyze the validation experiment on the challenge's training data with the 52 different feature subsets. Comparing the result of the 52 subsets some interesting observations can be made. Apparently, removing the first 2% of the most "confusing" features provides the by far highest performance gain among any neighboring pair of feature subsets. These "worst" features could be features that are confusing in themselves or features whose removal

Table 2: Results of the Clickbait Challenge 2017.

Participant	Rank	Mean Squared Error
zingel	1	0.0332
icarfish (this paper)	-	0.0351
emperor	2	0.0359
carpetshark	3	0.0362
clickbait17-baseline	7	0.0435

resolves inhibiting interactions. It further seems as if about 68% of the features have an at least slightly confusing impact; in most cases probably due to inhibiting interactions between some features that are reduced by removing one of them. And the reverse also seems to hold: There are some, but far fewer, features with beneficial interactions among them, slightly increasing the performance when included.

Figure 3 shows the features with high impact (absolute value of performance gain or loss larger than 10^{-5}) split by feature types (cf. Table 3 for the exact features and scores). One can observe that basically all feature types include good and bad features; with differences in strength and frequency. Most of the 12 engineered features have a high impact that even is about one order of magnitude larger than that of any other feature. However, two of the engineered features harm prediction performance (the average word length and whether a tweet starts with a number). One reason for the relatively high impact of the engineered features probably is that they are not as sparse as the n-grams or word lists. Interestingly, word n-grams are the only feature type without high-impact confusing features. An interesting side note is that the question mark seems to be discriminatory as a word token but confusing as a character token, while the exclamation mark is discriminatory in both categories.

From Figure 3 and Table 3, it seems as if the engineered features from the baseline approach are among the most valuable for scoring clickbaitiness. Maybe more compact models could thus be constructed if the information of individually higher impact features could be combined into higher impact engineered features. An example for a new engineered feature could be something like the "number of words ending with -ly or -ing" (counting adverbs, gerunds, etc.) since these suffixes seem to be rather discriminating character-based features. However, the hypothesis of creating good engineered features from individual other features is only partially supported by a qualitative analysis of the high impact features. While the engineered "number of stopwords" feature is very discriminatory and most of the high impact individual word unigram features actually are stop words, the engineered "number of mentions" feature (counting the number of @'s in a tweet) is not in the table of the high impact engineered features even though 3 of the 18 most discriminatory character tokens involve the '@' character.

5. CONCLUSION

Our study has demonstrated that an iterated leave-many-out strategy can help to identify good feature subsets from the ones employed in the Clickbait Challenge 2017 baseline. The challenge poses clickbaitiness scoring as a regression task on social media posts in form of tweets and thus typically comes with rather sparse features—a particularly challenging scenario for feature selection. The best feature subsets that our approach identified achieve a performance gain of 10–20% over the baseline with respect to the prediction error. At the same time, our technique is able to hint at possible feature interactions and further feature engineering possibilities.

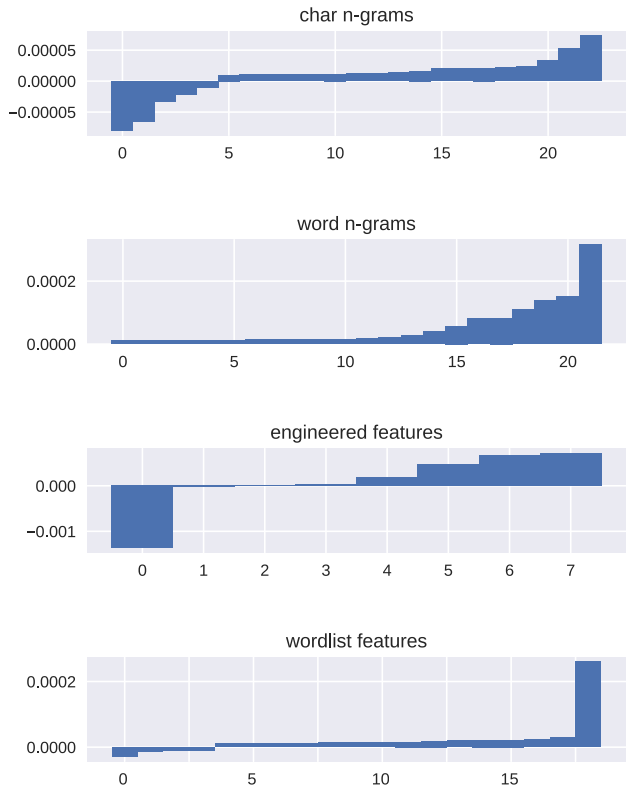

Figure 3: Feature impact values by category, sorted by performance. The height of the bar corresponds to the average leave-many-out error of the subsets including the feature. A high positive bar indicates discriminative features. Features with average error below 0.0001 are omitted.

Concerning the actual subset selection, open questions for future research still are how many different random feature subsets are needed to draw valid conclusions about the individual feature's potential and how to best cover potentially interacting feature groups in the selection of the removed subsets. Furthermore, our initial observations on different feature's strengths may hint at some possible engineered features that combine the individual features' potential—integrating more of such engineered features in a regression model could also be an interesting future direction. Finally, also ensemble techniques for feature-based learning or combinations with deep learning are rather obvious promising directions.

Table 3: Features with an absolute error impact value $> 10^{-5}$. All values are times 10^{-4}

Engineered	Value	Word lists	Value
Mean Word Length	-1.367	RcLoss	-0.290
Starts with Number	-0.303	Human collect.	-0.145
Abbreviation Count	0.106	PowCon	-0.115
Sentiment Polarity	0.249	EnlTot	-0.112
Longest Word Length	1.896	Exprsv	0.113
Part of Day	4.738	You	0.115
Character Sum	6.699	Virtue	0.117
Number of Dots	7.177	PowLoss	0.120
		Quan	0.142
		Dist	0.146
		Vice	0.148
		PowDoct	0.153
		Polit	0.190
		Finish	0.196
		Academic	0.220
		Eval	0.221
		PowEnds	0.228
		Stop Words	0.290
		Easy Words	2.600

Word n-grams	Value	Character n-grams	Value
how to	0.106	thi	-0.799
as	0.108	his	-0.662
sex	0.109	?	-0.338
dies	0.112	hi	-0.212
5	0.114	th	-0.105
that	0.125	hes	0.100
an	0.131	ing	0.102
to	0.135	wh	0.107
...	0.144	!	0.107
in	0.149	how	0.109
here	0.153	ly	0.116
things	0.178	ly␣	0.117
heres	0.201	s	0.118
is	0.278	j	0.135
at	0.404	n	0.162
a	0.574	.@	0.202
!	0.817	..	0.208
how	0.828	s␣	0.212
these	1.101	a	0.215
?	1.376	␣@	0.236
.	1.500	.	0.338
this	3.157	␣	0.530
		@	0.732

References

[1] J. S. Chall and E. Dale. *Readability revisited: The new Dale-Chall readability formula*. Brookline Books, 1995.

[2] I. Guyon and A. Elisseeff. An Introduction to Variable and Feature Selection. In *Journal of Machine Learning Research 3*, pages 1157–1182. JMLR, 2003.

[3] C. J. Hutto and E. Gilbert. VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text. In *ICWSM*, 2014.

[4] M. Potthast, T. Gollub, F. Rangel, P. Rosso, E. Stamatatos, and B. Stein. Improving the Reproducibility of PAN's Shared Tasks: Plagiarism Detection, Author Identification, and Author Profiling. In *CLEF*, pages 268–299. Springer, 2014.

[5] M. Potthast, S. Köpsel, B. Stein, and M. Hagen. Clickbait Detection. In N. Ferro, F. Crestani, M.-F. Moens, J. Mothe, F. Silvestri, G. Di Nunzio, C. Hauff, and G. Silvello, editors, *Advances in Information Retrieval. 38th European Conference on IR Research (ECIR 16)*, volume 9626 of *Lecture Notes in Computer Science*, pages 810–817, Berlin Heidelberg New York, Mar. 2016. Springer. .

[6] M. Potthast, T. Gollub, M. Hagen, and B. Stein. The Clickbait Challenge 2017: Towards a Regression Model for Clickbait Strength. In *Proceddings of the Clickbait Chhallenge*, 2017.

[7] M. Potthast, T. Gollub, K. Komlossy, S. Schuster, M. Wiegmann, E. Garces, M. Hagen, and B. Stein. Crowdsourcing a Large Corpus of Clickbait on Twitter. In *(to appear)*, 2017.

[8] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature Selection for SVMs. In *NIPS 13*, 2000.