A community driven approach for click bait reporting

Darius Bufnea
Department of Computer Science
Babeş-Bolyai University
Cluj-Napoca, Romania
bufny@cs.ubbcluj.ro

Diana Şotropa
Department of Computer Science
Babeş-Bolyai University
Cluj-Napoca, Romania
diana.halita@ubbcluj.ro

***Abstract*—Click baits are primarily used by online content publishers. Their purpose is to allure readers to click on a link and subsequently visit other articles by the same publisher, in order to increase page views and ad revenue. Most of the time click baits are used for pointing to low quality articles or thin content. The user falls into the publishers' trap due to a misleading or incomplete title or content exaggeration. A bait article link might also appear on social network shares or within the search engines result page, the presence of such a link in 3rd party web sites having a negative impact on user experience. Hence, it is important to properly identify and report them.**

In this paper we present an academic research browser extension meant to be used in the click bait reporting process. The extension offers to users the possibility to explicitly report a click bait, a series of details about the bait link being extracted and logged for further analysis. Based on all the gathered data, our goal is to obtain a community driven click bait samples database that may subsequently be used as an input for different supervised learning algorithms for click bait detection.

***Keywords*-click bait, information retrieval, web user behavior, community driven database, fake news, SERP results, academic research plug-in**

I. Introduction

The term click bait has its origins in old media, in television or radio shows, when before a commercial, the audience was urged to "Do not touch the dial!" (i.e. the audience will not believe what happens next ... after the break). Later, click bait was formally defined, in Oxford dictionary[1], as content presented in online media, whose main purpose is to attract attention and encourage visitors to click on a link to a particular web page. While this description covers the function of click bait, it doesn't fully differentiate it from genuine high-quality pages, mainly due to a good click-to-rate. Among their similarities, there are also dissimilarities.

While for a genuine website the content of an article generally justifies the headline, click baits tend to put more effort into attracting the click than in creating valuable content. Since their beginning, click baits exploited the audience's curiosity. On the web, click baits articles present irresistible headlines and eye-catching thumbnail pictures which are teasing the readers with a hint of what the article is about, without giving all the answers away. Usually, these strategies generate enough curiosity among the readers such that they become compelled to click on the link to fill the knowledge gap or to forward the material over online social networks in order to monetize the landing page.

Together with the expansion of the Internet and the migration of users towards online media from classical advertising platforms (newspaper, radio and television), content publishers increased their effort to maximize income, page views and ad revenue. In classical platforms, there was no or little interaction between content publishers and content consumers. By contrast, in online media, content publishers are able to interact with content consumers by collecting real time feedback about consumers' behaviour and about the most successful methods to trap and engage consumers and convert their visits in revenue. Among such methods we can mention: thin and low quality content split over multiple pages, click bait links between websites/domains under the same affiliation, aggressive and grey search engine optimization techniques, aggressive social media presence and advertising. In front of this, the content consumer is most of the time single and disarmed. In this context, 3rd party big actors on the WWW scene such as search engines or social networks should do more to protect the web consumer, but they failed in doing so, mainly due to their duplicity: they also serve as advertising platforms or ads brokers for content publishers. Only recently, being under fire by the public opinion, some search engines and social platforms started to take actions in some aspects such as fake news spreading.

Click bait links are placed in general by content publishers between sites under the same affiliation, their main goal being to keep the web visitor trapped as much as possible. Following such a bait link usually has a negative impact on user experience. But this unhappy and time consuming user experience can also be propagated outside a publisher's network to a 3rd party web site. A catchy but misleading or incomplete title can easily be used as a text link by a search engine in SERP[2] or by a social network in a news feed. Although not directly affiliated with the bait's creator (i.e. the content publisher), the source of the bait in these situations

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

[2]Search Engine Results Page

can be considered the search engine or the social network. This is another reason why the aforementioned should step in and take actions against this type of links and publishers who use them.

In this paper we present an academic research browser extension meant to be used in the click bait reporting process. The extension offers users the possibility to explicitly report a click bait, a series of details about the bait link being extracted and logged for further analysis. Based on all the gathered data, our goal is to obtain a community driven click bait samples database that may subsequently be used as an input for different supervised learning algorithms for click bait detection. At the same time, our goal is to advert to the web and research communities about this type of practice and to urge actions against it.

The rest of the paper is organized as follows. Section II presents different previous researches related to click bait identification, with the main focus on the data samples used in these researches. Section III presents our developed plug-in's architecture, while section IV describes the click bait features logged in our samples database. Section V and VI discuss the advantages and disadvantages of implicit report vs. explicit report of a click bait, taking into account how non bait link data could also be provisioned to the samples database (necessary for further analysis using supervised learning based methods). At the end, we present our conclusions and some further research directions.

II. Previous work

There have been extensive studies on identifying click baits, most of them being based on the machine learning approach. Biyani et all describe in [1] a machine-learning model to detect click baits by using a testing and a training dataset containing news articles collected during late 2014 and 2015. There are 1349 click bait and 2724 non click bait web pages coming from different news sites (such as the Huffington Post, New York Times, CBS, Associated Press, Forbes, etc.) whose pages surfaced on the Yahoo homepage. The articles covered different domains such as politics, sports, entertainment, science and finance. The authors defined 8 categories based on the relation between the title and the content on the landing page. All the classified web pages should fall in one of the following categories:

- Exaggeration: Title exaggerating the content on the landing page;
- Teasing: Omission of details from title to build suspense;
- Inflammatory: Either phrasing or use of inappropriate/vulgar words;
- Formatting: Overuse of capitalization/punctuation;
- Graphic: Subject matter that is salacious or disturbing or unbelievable;
- Bait-and-switch: The thing promised/implied from the title is not on the landing page: it requires additional clicks or just missing;
- Ambiguous: Title unclear or confusing to spur curiosity;
- Wrong: Just plain incorrect article: factually wrong.

In order to obtain their dataset, the authors of [2] focused on Twitter as a social media platform used by many content publishers. Twitter platform is usually used for publishing links to different websites, by specifying a short message (maximum 140 characters), the actual link and a picture. They collected tweets that included links from the top 20 most prolific publishers on Twitter. From tweets published in week 24 of 2015 they randomly sampled 150 tweets per publisher for a total of 2992 tweets (one publisher sent only 142 tweets in that time). There were three assessors who rated each tweet as being click bait or not.

Lockwood presented in paper [3] a dataset of 2136 articles, created as a spreadsheet of every title of every article published in Frontiers in Psychology journals throughout 2013 and 2014. In order to classify click baits, each article was analyzed by three raters by using only subjective factors regarding the title, such as: positive framing ("I know click bait when I see it"), phrase arousal, wordplay potential and the possibility of being a resource for social networks.

During 2017 there was a competition called Clickbait Challenge, which had the purpose of encouraging the development of detection technology for click bait in social media, by developing a classifier that rates how click baiting a social media post is. They propose in [4] a more general approach in which one may consider the content analysis based on natural language processing and image analysis. During the competition there were released three datasets which contain posts from Twitter, two of them being used for training and testing, while the third one was used for evaluating the models of the contestants. The collection of data from the first dataset is presented in [2] and comprises 2,495 posts: 762 click baits and 1,697 non click baits. The data is collected from each of the top 20 most prolific publishers on Twitter. The second dataset is presented in [5] and contains 19,538 posts with 4,761 click baits and 14,777 non click baits. The dataset provides JSON-Objects containing the text and images of the analyzed post as well as the main content of the linked target web page. Each dataset contains the following:

- a line delimited JSON file which contains the information extracted for a specific post and its target article, such as: id, timestamp, the short message, the image which is posted in addition to the short message, the title of the article, the description from the meta tags of the article, the keywords from the meta tags of the article and the content of the article).
- a line delimited JSON file which contains people evaluation considering a "clickbaitness" score. For every tweet, there were five individual evaluators that had the possibility to assign the tweet to one of the following category: not click baiting (0.0), slightly click baiting (0.33), considerably click baiting (0.66) and heavily click baiting (1.0).
- media: A folder that contains all the images posted in addition to the short message.

In order to help the users deal with click baits across

different websites, there were added in Chrome Store some plug-ins that promise to automatically detect click baits and to notify users about low quality websites. From analyzing Chrome Store statistics follows that most of them are not used by a large mass of users and some of them have negative reviews related to click baits detection. Their main disadvantage is that they usually target social media platforms (such as Facebook, Youtube, Twitter) as the main source of links to click baits. Their purpose is to notify users about the possible fake news presented on the platforms or to remove them automatically. Another disadvantage is that their efficiency is not scientifically proven.

Chakraborty et all present in [6] an extensive dataset which contains both click bait and non click bait web pages. They built their dataset from two different sources: 18513 articles from Wikinews, for non click baits and 8069 articles manually chosen from different domains (such as "BuzzFeed", "Upworthy", "ViralNova", "Scoopwhoop", and "ViralStories") for click baits. Then, they built a Chrome plug-in, called 'Stop Clickbait' which warns the users about the existence of click baits in different web pages and provides the facility to block certain click baits whereby it automatically block similar click baits in future visits. One of the biggest disadvantages of this plug-in is that it is too invasive regarding user privacy, due to the fact that it scans and saves all the anchor elements from every analyzed webpage. This is also the reason why we consider that it is not scalable.

III. Plug-in architecture

We chose Google Chrome web browser for implementing our plug-in. This decision is mainly driven by the fact that Google Chrome browser has the largest user community in general, and, at the same time, is by far the most popular web browser within our faculty's students in particular - we involved our students both in testing the plug-in in its development phase, and also in the click bait reporting process. The plug-in consists of two main components: the plug-in itself written in JavaScript programming language that runs in browser space and a back-end web service written in PHP hosted by our university's servers. These two components communicate over https. The back-end service is responsible for generating unique API keys used by the bait reporters (a bait reporter being a user having an instance of the plug-in installed in his/her browser) and also for storing the bait reports in a back-end database. The plug-in architecture is depicted in Figure 1.

A. The back-end service

The back-end service provides two endpoints which are used in asynchronous calls by the front-end part of the plug-in:

- `getNewId`: this endpoint returns a unique SHA256 API key used by a bait reporter in any future reports. In regard to security constraints, this endpoint may be called (i.e. returns a valid API key) no more than once every 5 minutes from a specific IP address. The `getNewId` endpoint is usually called at the plug-in initialization, the API key being stored by the plug-in in the browser's local storage. If the plug-in fails to obtain a valid API key, it will retry to call this endpoint next time when the user starts his/her browser. By setting the 5 minutes limit per IP address for requesting a new API key, we tried to avoid the situation when a malicious reporter means to pollute the bait database with noise (false non bait reports). This strategy can be used in conjunction with other techniques such as the mod_evasive Apache plug-in or some connection limits set via the operating system's firewall. Although, all these techniques cannot guarantee the absence of false non bait reports, they ease the process of cleaning the database of such reports.
- `report`: this endpoint is called by a reporter to actually store a bait report. The reported and logged features are presented in section IV. A valid API key is required for a successful report.

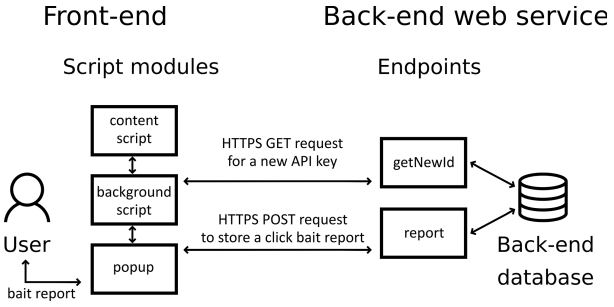

Fig. 1. Click bait report plug-in's architecture

B. Plug-in's front-end

The plug-in itself is written in JavaScript, its architecture following the Google Chrome's API programming and security guidance. It consists of three modules:

- a content script called `page.js` that interacts with the user's visited web page. This content script adds event listeners for different events that lead to new links being opened in the same tab or a new one: click (for catching regular left mouse clicks), mousedown (for catching middle mouse clicks, sometimes called scroll clicks), or contextmenu (for catching right clicks or links being opened in a new tab through the context menu). All these listeners send messages through the Chrome runtime, containing the attributes of a followed link to the second component of our plug-in, the background script.
- a background script called `background.js`. This module stays at the very heart of our plug-in. It is responsible for plug-in initialization, initial API key retrieval, persistence, receiving messages about accessed links from the above content script and storing information about these links.
- a popup page together with `popup.js` script. This module is responsible for user interaction with the browser when reporting a click bait. It retrieves all relevant attributes as stored by the above background script about

the current opened tab such as: how this tab was opened, its URL, which link was used to open this tab and the referrer URL of the current tab. All these features are then logged to the back-end service by calling the `report` endpoint.

C. Batch processing and filling in the bait database with additional data

Although not part of the plug-in itself and neither being necessary on the back-end for its functionality, we provide an additional batch processing tool that helps filling in the bait database with additional data. This tool is aimed to be run at a later time over the bait database entries and enhance them with: full absolute content of the bait destination URL, its excerpt, content description, languages of the source and destination URLs, the image or other media used by the bait link. All these features are extremely important in a further click bait classification but we do not retrieve and store them at report time for two reasons: it will increase the plug-in complexity and at the same time we express copyright infringement concerns in public releasing a click bait database containing these features (concerns mentioned in subsection IV-A).

IV. Logged features

The following features are reported by the plug-in and logged by the back-end service:

- `sourceURL`: the source URL of the bait;
- `destinationURL`: the destination URL of the bait;
- `linkText`: the text that appears within the bait link (i.e. the innerHTML of the anchor);
- `linkMedia`: whatever image or other media that may also link to the bait URL. In order to increase the bait impact, a text link is usually accompanied by an image within the same link (the same anchor tag) or as a different link that points to the same destination URL. In fact, the user might click on either the text or the media image, the browser's behaviour (and our plug-in behaviour too) being the same. In either case, we extract, report and logged both the text and the media. While the link's text is suitable for further text analytics, the media might also be suitable for use in machine learning based frameworks such as TensorFlow in order to detect click bait based on a media link (and also for associating the bait text with the bait media).
- `destinationTitle`: the destination URL's title;
- `isClickBait`: an always true feature indicating that the report is a click bait. This feature assures a future compatibility for allowing non bait reports (that will have this attribute set to false).
- `type`: how the bait link was opened: left click, scroll (middle) click, right click or context menu "Open in a new tab". This feature is not directly used, it is rather logged for debugging purposes.
- `pluginVersion`: the plug-in version;
- `openTime`: the time when the bait URL was opened;
- `reportTime`: the report time of the bait URL. This timestamp together with the previous one will be used in order to detect how long it took the user to detect the bait link's misleading text (or media) relative to the destination URL's content. For a click bait report, the user usually reacts in a few seconds.
- `reporterId`: a unique API key associated on the back-end with a bait reporter id. This key will not be used to identify in any way the reporter, rather it is used due to security concerns in order to prevent and limit the pollution of our baits database with fake reports. Other user privacy concerns and security related problems are also discussed throughout the paper.

One interesting discussion is about the click bait source domain or URL: should it be considered as a click bait feature or not? There are certain situations when a web visitor reaches a site that is part of an affiliated network of websites, or is under the same administration as the website the bait link is pointing to. Site owners use such methods in order to increase page and ad views, the user being almost trapped in this scheme of cross links that usually points to low quality or thin content. At a glance, most of the bait reports logged until now fall in this category, but further analysis is required in this regard.

On the other hand, there are situations when a bait article that uses an extremely catchy title is being indexed by search engines. A user may obtain in the search engine result page (i.e. SERP) a link to this article, the link text provided in the SERP being exactly (or build upon) the article's catchy title. Our plug-in will report the search engine as the source of the bait, but whether the search engine is responsible for the bait or not, or at least partially, remains an open discussion. A similar case is when the bait article is shared on a social network: the plug-in will log the social network as the source of the bait.

Previously provided click bait data samples [2], [3], [5] where taken exclusively from a social network (i.e. Twitter). The source of the bait not being a considered feature, all baits had the same source, i.e. the social network. Being aware of the source of the bait, our plug-in is also logging the source URL of the bait, but, as stated above, whether a search engine or a social network is responsible for the bait and if the source domain of a bait should be used as a feature in a future A.I analysis, remains an open and disputable question. However, the authors opinion is that, both these categories of traffic sources (i.e. search engines and social networks), as dominant and big actors on the WWW scene, should be more aware and responsible in providing links that might fall in the bait category.

A. Post log batch processing retrieved features

A series of features of a click bait may be retrieved at a later time through batch processing. Such features include:

- the bait language: this consists in the language of the text link and the destination URL's content. Although this language can be detected in some situation client side at

report time, we suggest that a back-end detection is more suitable, subsequent to report time. The language can be detected with the help of the html tag lang attribute (but there are frequent situations when this attribute is not properly set) or by looking at the Top Level Domain of the source and destination domains. Another approach is to back-end detect the bait language by calling a 3rd party API which provides languages detection services.

- destination URL's content: this content is absolutely necessary for any future text analytics that would imply click bait detection, the quality of this content being the one who delude the user. We are not logging the destination URL content at report time mainly for two reasons:
  - we do not want to provide the entire (i.e. full) content of the destination URL in any public release of the click bait database, due to copyright concerns (very probably this content is copyrighted by the owner of the domain who published that content). Instead, we suggest to the interested researchers to harvest this content on their own at a later time. Another approach would be to log in the bait database only a limited length abstract of the content, an excerpt or the content description as it appears in the header of the html files. However, this later approach would reduce the quality of any further analysis. It would be useful for the click bait database to also contain any image (media) that might appear within the bait link (there are situations when users click on the bait link being impressed by the graphical image rather than by the text link itself). However, the same copyright concerns mentioned above stand for these images (media) too, so we do not include the media itself in the click bait database - rather we will store only the link to these images, leaving to other researchers the possibility for retrieving them on their own. Another possibility would be to store in the bait database the thumbnail or the image at a reduced resolution in order to avoid copyright issues.
  - security concerns related to any private information that may be displayed to the user at the destination URL if the user is logged in or have any form of a session started. Accessing the destination URL at a later time, assures private information free content (any session variables used to personalize the content of the destination URL at report time will not be available at a ulterior content retrieval time).

V. Implicit or explicit report of a click bait

A click bait report can be trigger either in an explicit or an implicit way.

1) In an explicit way, the user triggers the submission himself/herself by clicking on the report icon of our plug-in when he or she is unsatisfied by the reached content following a link. This is the default and the desired behaviour of our plug-in.
2) An implicit report on the other hand can be performed automatically in the background by the plug-in, if the behaviour of the user suggests so. A simple heuristic for detecting such behaviour is the following one: the user clicks on a link that opens in a new tab a web page hosted on an external domain (i.e. an external link). After a second or so, the user closes this new tab (Google Analytics calls this behaviour "bounce"), most probably the user being unsatisfied by the quality of the reached content. All these actions/events (tab opening, tab closing, timestamp measurements) are already implemented in the current version of the plug-in, an implicit report feature for the plug-in being extremely easy to implement. Observation: instead of opening the external link in a new tab and closing this newly open tab within a few seconds, an external link can be opened through a click bait in the same tab. In this scenario the user would rather push the back button of its browser within a few seconds since the last Tab Update event, the general principle of the implicit detection of a click bait being the same.

However, at this moment, we will not rely on the implicit report approach mainly due to user privacy concerns. A future version of the plug-in might have a user configurable option, which will allow, if explicitly enabled by the user, implicit click bait reports in user behaviour scenarios as the one described above (i.e. the user "bounces", accessing one single webpage of the external domain for a few seconds).

VI. Implicit or explicit report of a non click bait

The developed plug-in could also be used to populate a back-end database with non click bait entries. In order for our collected entries to be suitable for a further supervised learning based analysis, an entry in our database should contain an attribute that will allow training of a classification algorithm that will map click samples into bait and non bait categories.

Similar to a click bait report, a non click bait report entry can be logged in two different ways:

1) an explicit report of a non click bait. Unfortunately, the cases when a satisfied user will offer such a feedback will be rare. We can think to a web visitor as to a content consumer, a satisfied consumer (i.e. client) will rarely offered feedback of his/her positive experience. Rather, the feedback will come from unsatisfied consumers (in fact this is what we will rely on to gather click baits reports). Another approach is to prompt user when a non click bait user behaviour is detected (see below). But, this could be considered an annoying intervention and we want our plug-in to be as less intrusive as possible. Additionally, a non click bait behaviour could only be detected after some time has passed since the click in question, the user performing meanwhile some other actions following that click. It will be useless and rather confusing to prompt the user for feedback related to a link he or she followed some time ago.

2) an implicit report of a non click bait. As similar to the implicit report of a click bait, such a feature could be used to automatically submit non click bait reports. While the automatically detection of a click bait involved the closing of a browser tab soon after opening, the detection of a non click bait relies upon an opposite heuristic: the user is "happy" about the reached content, spending more time on the newly open external link, consuming and scrolling the content, and eventually accessing more than one page of the external website.

Due to the same user privacy concerns, we will not rely on the implicit report of a non click bait either. Still, a future version of our plug-in might implicitly submit non click bait reports if explicitly enabled by the user.

In order to populate the data samples with non click bait entries as required by a supervised learning based algorithm, we will rely for now on a different approach: we will detect and collect such entries from an HTTP proxy server's logs. The proxy logs are obtained from the proxy server of our university which is running Squid, the most popular proxy server in the Internet [7]. The detection and extraction of non click bait entries from the Squid logs is currently a work in progress that is conducted as a graduation thesis research by one of our students.

VII. Conclusions

We have presented in this paper an academic research plug-in meant to be used in the click bait reporting process. The main scope of this paper was not to advance or evaluate any click bait detection methods or algorithms, we rather focus in providing to the research community a click bait samples database (and a way of building it). Previous sample databases provided by [2], [3], [5] were built exclusively with bait samples collected from a social network (Twitter in their case). By logging also the source of the baits and also considering sources other than social networks (3rd party web sites, search engines), our plug-in and the samples database build upon it, offers a serious advantage over the previous work. Another advantage is that the click bait samples database is community driven (and not under the influence of any of the big actors on the WWW scene). Moreover, the batch processing tool, delivered together with the plug-in, fills the bait database with additional data such as: the content language, destination URL's content, excerpt, content description and image used in the bait link. This additional data can play a fundamental role in any further click bait detection analysis, with an important focus on the content language which can be used to localize any detection methods to a specific language.

For future work, authors plan to be more involved in the click bait research community, with the intention of implementing and enhancing a supervised learning based algorithm for click bait detection on their samples database built upon the current research.

VIII. Acknowledgments

The authors of this paper would like to thank 2nd year students of the Faculty of Mathematics and Computer Science of Babeş-Bolyai University Cluj-Napoca for their involvement in testing the plug-in in its development phase and in the click bait reporting process.

References

[1] P. Biyani, K. Tsioutsiouliklis, and J. Blackmer, ""8 amazing secrets for getting more clicks": Detecting clickbaits in news streams using article informality." in *AAAI*, 2016, pp. 94–100.

[2] M. Potthast, S. Köpsel, B. Stein, and M. Hagen, "Clickbait detection," in *European Conference on Information Retrieval*. Springer, 2016, pp. 810–817.

[3] G. Lockwood, "Academic clickbait: Articles with positively-framed titles, interesting phrasing, and no wordplay get more attention online," *The Winnower*, vol. 3, 2016.

[4] "Clickbait challenge 2017," https://www.clickbait-challenge.org/, Last visited on 20.05.2018.

[5] 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," *to appear*, 2017.

[6] A. Chakraborty, B. Paranjape, S. Kakarla, and N. 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)*. IEEE, 2016, pp. 9–16.

[7] "Squid: Optimising web delivery," http://www.squid-cache.org/, Last visited on 09.05.2018.