BaitBuster: A Clickbait Identification Framework

Md Main Uddin Rony §, Naeemul Hassan §, Mohammad Yousuf *
§Department of Computer Science and Engineering, The University of Mississippi
*Gaylord College of Journalism and Mass Communication, The University of Oklahoma
mrony@go.olemiss.edu, nhassan@olemiss.edu, yousuf@ou.edu

Abstract
The use of tempting and often misleading headlines (clickbait) to allure readers has become a growing practice nowadays among the media outlets. The widespread use of clickbait risks the reader’s trust in media. In this paper, we present BaitBuster, a browser extension and social bot based framework, that detects clickbaits floating on the web, provides brief explanation behind its decision, and regularly makes users aware of potential clickbaits.

Introduction
Facebook engineers panic, pull plug on AI after bots develop their own language1—this headline and its similar versions, suggesting an apocalyptic situation, have misrepresented the true facts but, nonetheless, disrupted (shared more than 300K times2 on Facebook alone) the social media by exploiting a technique called as Clickbait. The term clickbait refers to a form of web content that employs writing formulas and linguistic techniques in headlines to trick readers into clicking links (Palau-Sampio 2016; Chakraborty et al. 2016), but does not deliver on promises3. Examples of clickbait are– You Won’t Believe What These Dogs Are Doing!, What OJ’s Daughter Looks Like Now is Incredible! and so on. According to a study performed by Facebook4, 80% users “preferred headlines that helped them decide if they wanted to read the full article before they had to click through”. (M. Scacco and Muddiman 2016) shows that clickbait headlines lead to negative reactions among media users. In this paper, we describe a novel approach to identify clickbaits, and present BaitBuster, a browser extension and social bot based solution framework.

Clickbait Detection
The key component of BaitBuster is the clickbait detection model. We define the clickbait detection task as a supervised binary classification problem. A traditional approach to text classification problems is to use bag-of-words (BOW) model to transform textual data into feature vectors and apply learning algorithms. However, motivated by the recent success of deep learning techniques in natural language processing, we use distributed subword embeddings as features instead of applying BOW model. Specifically, we use Skip-Gramsw, an extension of the continuous skip-gram model (Mikolov et al. 2013), which leverages subwords (substring of a word) to compute word embeddings (Bojanowski et al. 2016). Rather than treating each word as a unit like skip-gram, Skip-Gramsw breaks down words into subwords (length = 3) and wants to correctly predict the context subwords of a given subword. Then, the learned subword embeddings are aggregated to form the embedding of a word. This extension allows sharing of the subword embeddings across different words, thus allowing to learn reliable representations for rare words. We average the embeddings of words present in a sentence to form the hidden representation of it. These sentence representations are used to train a softmax classifier. Further technical details can be found in (Rony, Hassan, and Yousuf 2017).

Dataset: Using Facebook Graph API, we accumulated all the Facebook posts created within January 1st, 2014 and December 31st, 2016 by 153 U.S. based media outlets. After filtering picture and other unrelated type posts, we had about 1.67 million Facebook posts containing news headlines (details in (Rony, Hassan, and Yousuf 2017)). We use this dataset to learn the subword embeddings. A collection of 32,000 manually labeled news headlines (15,999 clickbait and 16,001 non-clickbait), curated by (Chakraborty et al. 2016), was used to train a softmax classifier.

Evaluation: We performed 10-fold cross-validation and repeated each experiment 5 times to avoid any random be-

---

2http://bit.ly/FacebookACount
3https://www.wired.com/2015/12/psychology-of-clickbait/
4http://bit.ly/FacebookClickbaitStudy

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
havior. We proposed 5 different models and the best model achieved an accuracy of 98.3% which is significantly better than the accuracy (93%) reported in (Chakraborty et al. 2016). We also compared our models with other works in terms of precision, recall, and other measures. More detailed evaluation can be found in (Rony, Hassan, and Yousuf 2017).

**BaitBuster**

BaitBuster consists of a browser extension that identifies clickbaits present in a Facebook timeline and a Facebook page which is administered by a bot through the Facebook API. The reason behind choosing Facebook is, according to a study performed by Pew Research, 44% of American adults get their news from Facebook.

**System Overview:** Figure 1 shows the architecture of the BaitBuster framework. The browser extension monitors a user’s Facebook news feed and alerts her if a post (link) contains a clickbait headline. In addition, it provides a brief explanation behind the decision which includes what language features present in a headline makes it a clickbait, whether the headline represents the corresponding body fairly, and a brief summary of the corresponding article so that the user can get the gist without leaving the current page.

**Implementation:** BaitBuster is implemented following the client-server system architecture model. The client side has a JavaScript based browser extension and a bot powered BaitBuster Facebook page. The extension scans the Document Object Model (DOM) of the current page, identifies the anchor elements, and sends the data to the server side using POST request. The server processes the request and sends response back to the client. Then the extension creates a new DOM object with the clickbait decision for each anchor and inserts the object along the corresponding anchor element. Figure 2 and 3 show the graphical user interface (GUI) of the extension. The server side has several components. The Article Body Scraper component uses Newspaper to extract the headline and the body of an article. We prepare a list of 1000 most frequent n-grams (n = 3) present in the 15,999 clickbait headlines. The Explanation and Summary Generator component detects if any of the n-grams is present in the requested post’s headline. It uses Gensim’s TextRank based summarizer to extract the summary of a body (collected through Newspaper) and cosine similarity to measure the similarity between the summary and the corresponding headline to give an idea of how fairly a headline represents its body. Activities of the browser extension are logged in a database. These data allows us to know, for any time interval, the most viewed clickbait posts and their source controversy. Using Facebook API, it publishes the report to the client side page on a daily basis. As numerous malicious bots are spreading disinformation across the web (Woolley and Howard), we believe this is a small step towards fighting misinformation using benevolent bots. We plan to release an API service so that third party programs can use BaitBuster.

**Related Work**

There have been several attempts to limit clickbaits using browser extensions. For instance, B.S. Detector and Check This by MetaCert maintain an aggregated list of sources and check web contents against the set of sources. One limitation of this approach is it doesn’t allow checking content which hasn’t already been checked by the aggregated list. Stop Clickbait (Chakraborty et al. 2016) uses supervised models to check clickbaits. However, they don’t provide any explanation behind the model’s decision. According to our knowledge, only BaitBuster provides deep learning powered classification and supplements it with explanation and summary by leveraging the headline-body relation.

**References**


---

3http://pewrsr.ch/2yzFiRr
4https://github.com/codelucas/newspaper
5https://radimrehurek.com/gensim/
6http://bsdetector.tech/