WEB CRAWLING AND COMMUNITY REVIEW

TO PREVENT MISLEADING LINKS

Senior Project 2019-2020

Presented to

The Faculty of the Department of Computer Science

California State University, Bakersfield

In Partial Fulfillment

Of the Requirements for the Degree

Bachelor of Science

in

Computer Science

By

Adam Arreguin, Adrian Gutierrez, Joel Staggs, Kenny Taylor

May 2020

ABSTRACT

Web Crawling and Community Review to Prevent Misleading Links

By

Adam Arreguin, Adrian Gutierrez, Joel Staggs, Kenny Taylor

This project is aimed at creating an automated system to allow users to judge the degree of fidelity a web link will provide without actually visiting the site. Since a common method of generating revenue for online-only companies involves displaying advertisements on their website. In order to maximize this revenue; however, some companies may manipulate potential viewers. An increasingly standard tactic is to use titles or thumbnails that intentionally mislead and exaggerate the expectations of those that may potentially visit a webpage. This can lead to a disappointing or frustrating experience for the user, while also lowering standards for web-delivered content. By giving users a measure of quality of a page's content or summation of that page, this software seeks to provide a more pleasant experience to vulnerable web users.

ACKNOWLEDGEMENTS

The project team would like to thank Dr. Chengwei Lei for guidance and direction during the development process. The project team would also like to thank California State University, Bakersfield Department of Computer and Electrical Engineering and Computer Science for use of computer hardware and facilities.

TABLE OF CONTENTS

LIST OF TERMS

Application Programming Interface (API): An interface or communication protocol between different parts of a computer program intended to simplify the implementation and maintenance of software.

CHAPTER 1

Introduction

Description of the Problem

Our project seeks to improve users' web experience and to diminish the impact of companies that utilize misleading tactics to advertise their websites. A misleading tactic can be duplicated versions of one article that force users to load new pages in order to view the rest of the article; causing unwanted advertisement spam. Utilizing an automated and user review system, articles and websites will be given scores regulated by metrics to ensure the user can control what they explore.

Methods

Web pages from selected sites will be parsed regularly and given scores based on several metrics. These metrics may include: keywords, diction, proper citation, author, and possibly other metrics that are identified as valuable during the process. The generated scores will include article score, author score, and website score. This is intended to provide a more granular assessment of very large sites. For example, CNN may have Opinion or Blog sections that would likely bring down the site's score if only one total score were to be chosen, since a blogger might not be nearly as objective as a professional journalist.

While many large sites will be automatically reviewed, users will be able to simply hover over an unknown article and automatically generate scores. This functionality is intended to be as transparent to the user as possible and provide little additional interaction. While the automated review process is key to our system, a comment system is also planned. These

comments will be made on a separate website where users can search the data our program has gathered by themselves and add their own reviews.

A well-worded article might still omit important truths, or misrepresent data it displays, or lie about the content of its sources. Another website may lead a user to the site to simply tell them a very small bit of information that would be easily included in the title. Such information would allow the user to better know whether or not they'd find the article interesting. These two examples of manipulation might not be clear to an average user. The internet has a breadth of users and some less critical users may be taken advantage of to generate easy ad revenue. That issue is the crux of our project and what we seek to improve.

Description of the System

The project contains several interworking parts: an automated web crawler, a text parser/reviewer, a browser plugin, a large database, and user review website. These components will be distributed across two or more cloud servers on Amazon Web Services or similar platform.

Crawler

The foundation of the project is the web crawler running on a Linux server. This program, likely written in C++, PHP, or Python, will scrape several popular websites for titles, URLs, authors, and content. This will be written to honor the "robots.txt" policy of targeted websites and will parse data in non-intrusive way. The crawler will use delays and randomization to be as low impact on the target site as possible.

Parser

The text parser is the next program our project will run. The program will periodically parse obtained articles and extract crucial data. This data will be parsed using possibly machine learning algorithms or dictionary parsing. A performance intensive language, likely C or C++, may be necessary to ensure that new articles (new links, not included in our database, that the user hovers over) can be analyzed quickly, without significant waiting by the user.

User Review System

Registered users will be able to post comments on web links on our separate website. This platform will be a method of accessing a large amount of data at once. Topics, authors, and websites will be searchable by users. A user searching for a topic, such as global warming, will be able to see reputable articles across several websites.

Browser Plugin

The culmination of this will be a browser extension that will conveniently display the data the other branches of our project have gathered. When hovering over a link, our database will be queried and the results will be displayed in a window over the hovered link. This window will display the page score, author score, and site score, as well as a link to user reviews. If the webpage has been previously scraped, the data will be readily available. However, if the webpage has not been previously added to our database, it will trigger a on-demand analysis and display to the user.

Database

The backend database will be run on an instance of MySQL. The database will contain a collection of user scores and comments as well as page, site, and author scores generated by

the parser. The database will also include metadata, such as the date/time when the page was last analyzed, parent domain, and author identity. Data from the database will be served to other components using a custom web API running on the frontend web server.

Significance of the Project

The targeted application of this project is to prevent misrepresentative journalism and advertising from impeding on a user's browsing experience. There are several other possible uses that stem from the intended use, such as phishing detection, but the program will be written with simple 'clickbait' in mind. The program will also serve as an automated objectification of websites and websites' articles to ensure a user's browsing experience is in control of the user.

Chapter 2

Design Considerations

System Architecture

The project team decided on a modular architecture consisting of a central database and web API, user browser plugin, analyzer, crawler, and website. This provides a layer of abstraction and scalability as the project grows.

When visiting a website, the user's browser plugin enumerates the hyperlinks on the web page and sends them to the web API. The API first checks whether a recent score has been stored for each URL. If a recent score is available, the cached score is returned to the browser plugin. If no recent score is available, the API calls the analyzer Python script to retrieve and score the page. The analyzer script returns a score and the page title in stdout, which is captured by the API, stored in the database, and returned to the user's browser plugin. When the user's mouse moves over a hyperlink, the page and site scores are displayed to the user, along with a link to user reviews.

The standardized interface between the API and analyzer allows the team to test different analyzer code without modifying other parts of the project. The team expected many incremental enhancements to the analyzer code during the development cycle. A script to update all cached scores was created and can be ran whenever the analyzer code is updated.

The web crawler was designed to allow users to request the indexing of an entire website. It defines a root URL and follows hyperlinks N levels deep, adding each URL to a list. The analyzer code is then ran on each URL and the score cached in the database. The team's intent is to display live updates as each URL is scored.

The website is intended to provide a user-friendly front end to the system. It will provide a high-level view of the data, such as the top 10 highest and lowest-ranked websites. It will also provide an interface for user reviews and social features, as detailed below. The website will run PHP code and interact with the database directly, pulling the cached URL and site scores stored by the API and analyzer scripts.

Algorithm Selection

The project team originally intended to select a single algorithm for all scoring. Individual pages would be scored and stored in the database, then the average score per domain would be used as the site score. The Naive Bayesian algorithm was selected for scoring individual pages. The team later encountered the PageRank algorithm, which generates a score for each domain based on the number and quality of links to it. PageRank appears to provide a better site reputation score than averaging the Bayesian results, and the team opted to use it to generate the site score.

Bayesian Algorithm

For the task of determining whether a page is clickbait, the team selected the Multinomial Naive Bayes classification algorithm. This algorithm processes the contents of the page's title and attempts to differentiate articles that may use manipulative language in their title. This algorithm will be elaborated further in the Procedure section.

Pagerank Algorithm

In order to determine the page ranking of urls to help better validate whether web pages are clickbait, the team decided to implement a page rank algorithm based off of the Google

page rank algorithm. The algorithm ranks web pages based on the popularity or outbound links a web page may have linked to it. This will be elaborated further in the Procedure section.

User Engagement

User engagement will be key in providing life to the website. Communication amongst one another will be the main driving force behind the website's main focal point. This communication will evolve into various social features provided to ensure user engagement is at its highest. Consisting of a user review system, user friend or follower system, users having the ability to create and customize their own personal profile page and view others' profile pages, private messaging system amongst friends and followers, and more ideas to be developed. Other social media platforms contain all of these features and more in order to provide users with as much ability to engage with one another as possible.

Websites such as Facebook, Twitter, Instagram, and other social media applications are driven by user engagement. Whether it be by a friending system, private message system, group chat system, or a main feed in which all friended users can contribute to. The goal of our projects website is to replicate the basis of these ideas with our own spin on each social feature. Of course, there is only so much that can change from each basis therefore deviation may not be necessary. The first feature to be overviewed will be the user review system we plan to implement in the coming months.

User Review System

One of the biggest focuses behind our project will be the automated ranking system charged with the task of judging a page by the information and data fed through our analyzer and placed into our database. Aside from this feature will be the maunal ranking or review of a

page created by a user with an active account. This user review system will allow for users to comment on along with the ability to rate domains and articles. These forms of rating can be viewed while having the browser extension active or viewing an article's rank through our website.

The user review system we plan to implement and develop will have no connection to our automated ranking system. This is to ensure subjective rankings, comments, and other user provided reviewing features do not affect the results from our analyzer. However, it is also important to allow for user's to express their opinions regardless of the results from our analyzer. Therefore the user review system will be put in place to open the conversation of like-minded or differing viewpoints on a page. It will encourage and engage users to communicate and socialize with each other.

Social Features

The social features planned to be incorporated will give the user the ability to communicate and follow other users with active accounts. The process will begin with users specifically searching or finding another user to friend or follow. An "add friend" button will be displayed on a user's profile page if they are currently not added. Once, added users can private message one another, tag, and comment under posts. Large social media platforms all allow for these features to exist because they drive user engagement.

Users can create accounts with unique usernames which are used to identify themselves with when interacting with others. One of the key social features to be implemented will be the ability to friend or follow other users. We are still deciding on whether to design a friending system or a following system because they both imply two different connotations. They do, however, result in two users mutually agreeing to connect together through our website. Once

two users have participated in adding one another to their social feed, private messaging would then be available.

A private messaging system is planned to be developed with the hopes of driving user engagement. This is a key feature that comes with every website that relies on user engagement. Users will be able to open up a new section to their account which shows all previously read messages along with any unread new messages. When opening a conversation, each message will contain the time sent along with whether or not the receiving user has read the message. Currently, the basis of this private messaging system will be simplified to justify time spent working on it. We are looking into other key features for it which leaves future development open to different possibilities.

Users will have the ability to tag those they have added as a friend or follower to posts or other reviews created by other users. They will also be able to comment on user reviews. This idea is still in development as we decide the direction to take the social media experience with our website.

Chapter 3

Procedure

Platform Selection

During the initial planning, it was unclear what computational, network, and disk storage resources would be necessary to support the project. The project team decided on a modular approach, where the browser plugin, analyzer, and website communicate through a web API. The modular design allows the system to be scaled horizontally, where each component could be hosted on one or more servers, if needed. The team decided to build the project on a single server and scale out as needed.

Amazon Web Services EC2/Lightsail was selected as the hosting provider. Utilizing AWS avoids some of the potential security and network use policy issues that might arise from building the project on the Computer Science department server. Working in the AWS virtual environment also offers more flexibility in scaling. The initial single server can be stopped and instantly started back up as a larger instance with more CPU, disk, and memory resources. It also provides the flexibility to instantly start additional hosts and temporary test hosts with no upfront cost or long-term commitment.

Web API Design

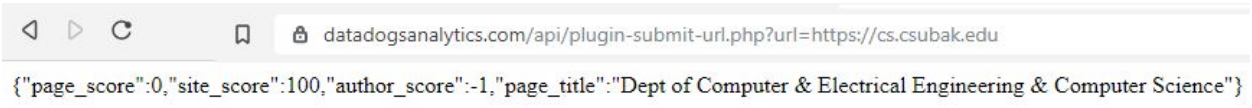
A web API was designed to provide a specific, limited interface for the browser plugin and other components. The API currently consists of two pages ‘plugin-submit-url.php’ and ‘plugin-submit-multiple.php’. The former allows the browser plugin to submit a single URL for scoring, and the latter allows the browser plugin to submit a JSON-formatted list of URLs for

scoring. JSON was selected due to its native support in Python and decent level of support in PHP and Javascript.

When a URL or URLs are submitted to the API, the API code first queries the database to determine whether each page is already scored. If the pages are already scored, the API will return the scores to the browser plugin. Using these cached results reduces the latency of the overall system and reduces the amount of requests imposed on the target websites. Future updates to the system will consider the age of cached scores and either remove or invalidate cached entries that are deemed to be too old.

If the URL score is not available in the database, the analyzer Python script is called for each unscored URL. The Python script fetches the page, runs its algorithm, and returns the page score and page title on two separate lines. This adds a level of modularity to the system, since the analyzer scripts can be interchanged for testing as long they conform to the same input and output parameters. When the score and page title are received from the analyzer script, the API stores them in the database cache and returns them to the browser plugin. In both cases, the scores are returned to the browser plugin as a JSON-formatted list.

Figure 2.1 - Example JSON-formatted API Output



Bayesian Algorithm Design

The Bayesian classifier relies on principles of Bayes' probability theory. Each word in the webpage's title is considered an event. The program is given labelled data, which consists of many articles (documents) that have been given a classification (clickbait or not). The words (or

events) within that document are then assigned two probabilities, which are determined by the quantity of that event contained within the two document classes of the labelled data. These probabilities are combined to determine the final likelihood a document is a part of either class.

Pagerank Algorithm Design

The page rank base derives from the Google's Page Rank Algorithm. The algorithm outputs a rank or popularity score that represents the likelihood a person will click on the web page which helps provide input whether a web page is clickbait. The page rank initializes the same value for all pages with an initial value of 1. Suppose a web page had only three outbound links, one third of the web page's value will be distributed to all the outbound links. Hence, each page rank score will be dependent on the page rank values of other pages.

Website Design

Our website begins with the sign in page or if the user does not have an active current with Data Dogs, under the sign in form, there is a link which takes them to our sign up page. Currently, users can sign up with a unique email and username not already in the database along with a strong password. Once signed in, the user proceeds to the main feed page where articles are able to be sifted through to be viewed with the article's rank attached. Another option for the user is to click their profile picture which takes them to their personal profile. Their profile can be customized to have a profile picture, bio, occupation, and birth date.

The sign in page is the basic query call to the database requesting to see if the user entered username and password find a match in the database. If user wishes to created account, there is a hyperlink under the sign in form which leads to our sign up page. Here, users can create an account using a unique email, unique password, and strong password. For

unique email requirements, the first check is to see if there is the '@' symbol in the user entered email. If there is one, then it splits the string to check if there is a '.' symbol after the '@' symbol. If there is, then it is a valid email which proceeds to query a request to the database to see if there is a matching email. The only unique username requirement is that there is no matching username already in use with an active account. As for the strong password requirements, the entered user password must be eight characters long with at least one number.

The main feed page after a successful sign in is currently being developed. As of right now, it contains a search bar which allows users to search our database of ranked articles. The main page also displays the user's profile picture which if clicked, takes the user to their personal profile page. Users can also press the sign out button under their profile picture which signs their account out and returns the user to the sign in page.

After a user proceeds to their personal profile, here they can view their profile picture, bio, occupation, and birthdate. They also have the option to edit any of these four currently provided features. The hyperlink to edit their profile opens a new dialog window which allows the user to upload a new profile picture and update their profile information. Once the user has pressed the save button after making any changes or updates, the edit profile window is closed and their account information is updated.

Performance Analysis

With regards to performance, our system was able to complete its tasks with minimal delay. Given the extremely small amount of data being sent or received at any given moment, we were able to transmit results at whatever speed the user's internet connection could manage. Originally, there were plans in place to accomodate for network latency in order to provide a smoother user experience; however, this did not provide any noticeable improvement as latency would soon be revealed to be a minor issue. An early version of the browser extension would

gather links as they appear on the page and gradually submit them for assessment; however, this proved to be detrimental to the performance of our extension as it would cause significant delay to the appearance of the assessment due to additional complexity to the user's browser.

Correctness Analysis

Due to the subjective nature of "clickbait" labelling, the team had to agree on some common features for a clickbait article. These features included flattering the user, suggesting user outrage, provoking fear in the user, implying need, and some others. Our static corpus was geared toward news websites, as a large amount of misleading news and political content could do the most damage to users and social wellness. We created a testing dataset that consisted of 30 articles, some clickbait and some news. We did not want to overfit that data, but we did include new news and clickbait articles within our corpus in an effort to improve our predictions. After some minor changes we were able to achieve 100% accuracy in that testing data. While this would be considered a success normally, it's important to test our algorithm outside of a controlled environment. We tested two reputable news domains: Fox News and CNN, and two clickbait domains: Cosmopolitan and Buzzfeed. When using our browser extension, the reliability for the two news sites, Fox News and CNN, both converged on approximately 95% and 88%, respectively. The two clickbait sites, Buzzfeed and Cosmopolitan, reached 44% and 26%, respectively. While it would be impossible to fully quantify the accuracy of our corpus given the immeasurable scale of the internet, and the subjectiveness of clickbait categorization, we felt that these results were consistent with what we had hoped to achieve with our algorithm..

Chapter 4

Evaluation/Conclusion

Effectiveness at Identifying Clickbait

The implementation of both page rank and naive bayes algorithm proved to be effective in the attempt of identifying clickbait when compared against our training data. Our team's analyzer implements a multinomial naïve Bayes classification model. This model weighs each word, assigning them a probability based on their frequency within labelled documents, in this case "clickbait" or "not clickbait". We were primarily concerned with clickbait in regards to news and political topics, as such the corpus used to build our model consisted largely of words contained within news articles from reputable sources along with clickbait. Our model currently only uses a static corpus, so it has lesser success when confronted with topics not included within the initial corpus. Furthermore, with the integration of the page rank algorithm ranks websites based on the quality of links that are associated with the website. Checking the quality of a domain and analyzing the data of the domain or document validates that these two algorithms were effective enough to achieve the results that were sought.

Usability, Value of User Engagement

Our community form web application is designed for users to create a personalized account to interact with other users and domains stored in our database via a following system. A user's main feed is only populated once they are following another user that has made any comments on an article stored in our database, the user is following a domain which contains an article that has any comments posted on it, or of comments made by the user who is signed

in. The main feed consists of the 20 most recent comments made by a user they are following or from a domain being followed. Each comment card is made up of three sections: the heading, comment, and article title. The heading section has the username of who posted the comment and the time and date the comment was posted. In the middle field below, is the section where the comment is. The bottom section has the domain icon of the domain of the article commented on and a hyperlink with the title of the article and the link to the article. This main feed encourages users to keep up with current news, public opinion of others, or follow their favorite domains.

The first thing a user can do after setting up their account is personalize their profile page for the world to see. Here, a user can upload their own profile picture, add a personal bio, add their occupation, and add their birthdate. Any other user on the website can visit another user's profile page simply by searching their username in the user search bar and clicking a username. When visiting a user's profile page, there will be either a "follow" or "unfollow" button depending on the users' follow relationship. Viewing other user's profile pages gives a more detailed description of the type of person a user is before deciding to follow one another. This allows for a user-to-user engagement to develop and encourages users to follow multiple people to populate their main feed with diverse thoughts.

Through our browser extension, users have the ability to post comments on any given article of their choosing, regardless if it is in our database or not. However, on the website application, a user can only post a comment on an article if it is already in our database. This is how our browser extension and community forum web application co-exist. It gives purpose to using both services provided rather than only using one while disregarding the other.

Our project aims to provide limitless usability to users to keep them engaged with the online world around them. It allows for them to dictate what they really want to see and without

the bombardment of advertisements and misleading content. Our community forum is designed to give users the ability to facilitate and organize the thoughts of others and engage with articles on the internet.

Possible Future Improvements

Future improvements to the system include primarily scaling the system up to a large user base and increasing the page score accuracy.

The system's modular design lends itself to scalability. The project server resides on a small Amazon EC2 computer instance, with the web API, database, and analyzer code all residing on the same virtual server. The API and website are the only component of the system exposed to the outside world. In a large-scale production environment, the API and website could be run on a number of independent virtual servers placed behind a load balancer. These front-end servers could talk to a large, backend MySQL cluster. With some minor API modifications, the analyzer scripts could be offloaded to either a pool of dedicated analysis servers or a serverless system such as AWS Lambda. This would provide a high level of fault tolerance and scalability to the system.

Page score accuracy could also be improved. The existing data set of 190 clickbait and non-clickbait page titles provided a reasonably-good degree of accuracy. This could be extended further with more sample data. The system currently considers only the page title for scoring. Analyzing the full content of the page might yield better results, and might also require adjusting the scoring criteria (i.e. considering only nouns and verbs). In a production environment, it might also be necessary to add a

mechanism to manually whitelist pages or sites as non-clickbait. It would be far less time consuming to whitelist the algorithm's occasional errors than to try to make the algorithm conform to all possible scenarios.

REFERENCES

How Naive Bayes Algorithm Works. (2019, December 16). Retrieved from https://www.machinelearningplus.com/predictive-modeling/how-naive-bayes-algorithm-works-with-example-and-full-code/.

PageRank. (2019, December 13). Retrieved from https://en.wikipedia.org/wiki/PageRank.

APPENDIX A

Anticipated Division of Labor	
Adam Arreguin	Front-End web design and social media web development
Adrian Gutierrez	Machine learning algorithm implementation and automated web crawler design
Joel Staggs	Browser extension programming and machine learning algorithm implementation
Kenny Taylor	Design and implementation of MySQL relational database, web API, and automated web crawler

APPENDIX B

Tentative Project Timeline	
10/1/2019	The primary responsibilities of each member are defined.
10/15/2019	Components of the system will be able to effectively send data to each other.
10/20/19	The browser extension will be able to dynamically append an html inlay over a hovered link.
11/15/2019	Each component should be able to fulfill its own role sufficiently.
11/20/19	The optimal machine learning algorithm will be fully implemented.
12/1/2019	The system will be functional but may require additional refinement.
1/1/2019	The browser extension will allow media previews.
2/1/2019	Extension may warn user of ad quantity on page.
3/1/2019	Back-end algorithms will be optimized for faster runtime and higher accuracy.
5/1/2019	The system will be complete and ready for presentation.