A Multi-strategy Recommendation Algorithm to Retrieve Broken Hyperlinks

Emiraldo Lozano, Eng. (c) [1], Roger Calderón, M.Sc. [1], and Javier Enciso, M.Sc. [1]
[1]Universidad de los Llanos, Colombia, {emiraldo.lozano, rcalderonmoreno, jenciso}@unillanos.edu.co

***Abstract*– Broken hyperlinks are a growing problem on the Internet that affects all kinds of websites; especially as websites age and contain a reference to content that is no longer available, or has changed its location. This is a big problem as information from the Internet is being used in all kinds of academic, business and legal contexts. In this work, we present a multi-strategy recommendation algorithm to retrieve broken hyperlinks. As result, we show 72% accuracy in the retrieved hyperlinks by combining multiple strategies.**

Keywords—Broken hyperlinks, archive.org, hyperlink life cycle, web information retrieval.

I. Introduction

Hyperlinks are fundamental to the web, although they present some problems: one of them is the *404 error* (not found), which indicates that the client was able to communicate with the server but it could not find what was requested. These errors are a frequent problem that affects user experience and ends up affecting much more than that [2].

Internet's fast expansion has allowed it to be used in different fields such as education, government, commerce, among others. Thus the importance of making it reliable, as nowadays it is unclear if in the coming year, forthcoming semester and even nearby month, the content exists. Due to this problem applications have arisen; browser add-ons, desktop applications, web applications, etc.

The objective of this project was to design and implement an algorithm that recommends an "equivalent" webpage for a given broken hyperlink. The algorithm was developed during an internship at Mobile Corp S.A.S. The implementation was further integrated into a web application that detects broken links, and suggests how to fix them based on different advanced techniques.

The remainder of this paper proceeds as follows: section II presents an overview of the related work and context. Section III exposes the development process by using flow charts and supporting open source technologies. Finally, results and conclusions are discussed in sections IV and V, respectively.

II. Methodology

A. Related work

Some previous works related to this project are "PageChaser: A Tool for the Automatic Correction of Broken Web Links" [3] and "Updating broken web links: An automatic recommendation system" [4]. These solutions were implemented for research proposes. In contrast, the present work was integrated into a commercial solution available on the web.

B. Context

According to a 2013 study cited by *The New York Times* [5], 49 percent of hyperlinks used to support decisions of the U.S. Supreme Court, no longer work, and what is worst, links in the Court website, which pointed to pages inside it, do not work either.

In a previous experiment from 2003, Fetterly et al. found that every week 1 out of 200 links disappeared from the Internet [1]. Two years later McCown et al. would discover that half of the documents cited in D-Lib were not available 10 years after published. Other studies have found that this situation is even worse in academic literature [6] and is definitely not better in the medical and business fields [9].

As it can be seen, keeping all the hyperlinks on a website running is a difficult and increasingly important task, as information from the Internet is being used as support in all kinds of academic, commercial and legislative processes, replacing the traditional, static and permanent sources that were used before: books.

Different institutions have started to take steps to try to repair the damage produced by broken hyperlinks. In the Court's case, the solution adopted consists of having a record of the exact date when the website was visited and saving a physical copy of the information consulted. Other government entities have opted for saving a PDF file of what they call "webcites"[1].

In the field of laws, people like Jonathan Zittrain, professor of Law and Computer Science at Harvard, work in more ambitious solutions. Perma.cc, Zittrain's solution, is a platform built and maintained by a consortium of law libraries, that will allow writers and editors to capture and save information, normally transitory in the web, with a new link, this time, permanent [5].

Digital Object Identifier (DOI): http://dx.doi.org/10.18687/LACCEI2018.1.1.140
ISBN: 978-0-9993443-1-6
ISSN: 2414-6390

[1] https://www.webcitation.org/

III. DEVELOPMENT

A. Flowcharts

Four solution strategies are proposed: '*Database search', 'Broken link search on the Internet', 'Text search on the Internet' and finally, 'Search on archive.org and the Internet'*. In figures 1 to 4, flowcharts of the proposed solution strategies are presented. As the number strategy increases, the same is true for the amount of computing power required to find a recommendation.

Fig. 5 shows the execution order of the proposed strategies.

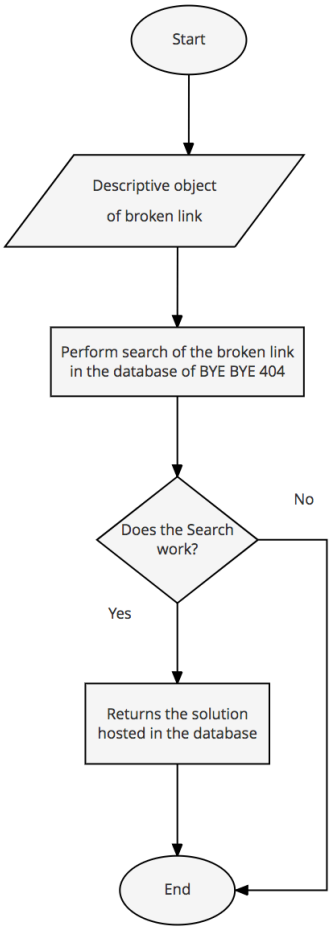

Fig. 1 Strategy #1: Database search.

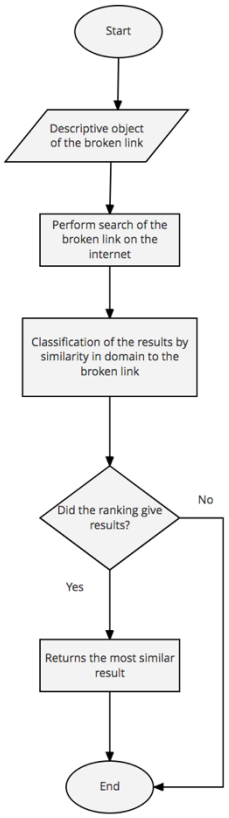

Fig. 2 Strategy #2: Broken link search on the Internet.

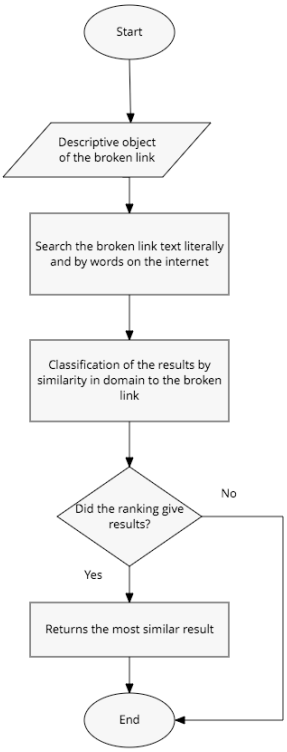

Fig. 3 Strategy #3: Text search on the Internet.

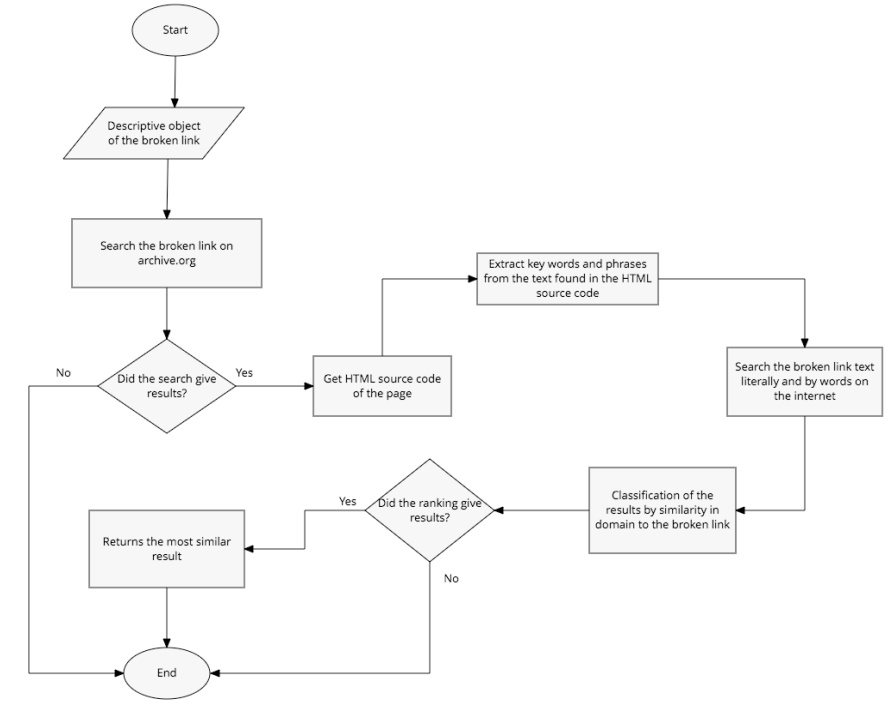

Fig. 4 Strategy #4: Search on archive.org and the Internet.

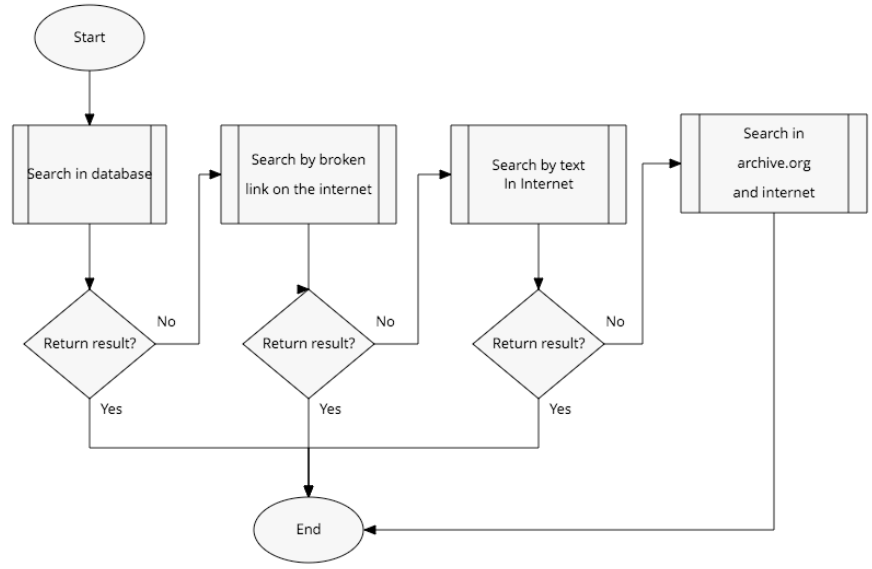

Fig. 5 Strategy #5: Algorithm execution order.

B. Technologies

Python[2] was the programming language used in the implementation of the strategies, with *Django* as the web development framework.

Python is an open source multi-paradigm scripting language released in 1991 by Guido Van Rossum and widely used for web applications [12]. *Django* is an open source web framework released in 2005 which follows the model-view-template (MVT) architectural pattern [11]. Web frameworks like *Django* support developers in the design and maintenance of complex applications; both *Python* and *Django* were established as requirements by the customer.

BeautifulSoup[3] is the python library that was used to perform the web scrapping, which is a technique for extracting attractive or relevant information from websites [13]. *BeautifulSoup* allows for the syntactic handling of HTML and XML files extracted from the website using *Requests* [14] in order to provide various forms of navigation, search and modification of the parsing tree.

Requests is a Python library written to perform HTTP requests in a simple way with functions such as: thread safety, connection pooling, SSL/TLS verification on the client side among others [15].

Git[4] was used as the version control system and Unit testing was implemented via *Django's Unit Test* [5] infrastructure.

At the costumer's request the complete web application had to be deployed on an Infrastructure as a Service (IaaS) platform. IaaS is a type of informatics infrastructure which is provisioned and managed through the Internet and allows for easily reducing or scaling vertically the resources to make them fit the website demands [10].

Finally, 'Google', one of the most famous search engines, and 'archive.org' were used. Also known as Internet Archive, 'archive.org' is a digital library whose goal is to preserve files, websites and other media resources; it was created in 1996 and currently stores approximately 324 billion 'screenshots' of different websites [8].

IV. RESULTS

A. Tests

In statistics, the sample size is the number of subjects that make up the sample drawn from a population, necessary for the data obtained to be representative of it [7]. Equation (1) is used to obtain that value.

$$n = \frac{N\sigma^2 Z_\alpha^2}{e^2(N-1)+\sigma^2 Z_\alpha^2} \quad (1)$$

Where 'n' is the sample size, 'N' is population size, 'σ' is the standard deviation, 'Z' is a value obtained through confidence levels and 'e' is an acceptable sampling error limit. 'Z' and 'e' are both constants whose value is at the discretion of the researcher.

For this work, population size was 497 records, so Equation (1) gives 175 as a result, this being the amount of records taken into account to measure the accuracy of the proposed algorithm.

Table 1 shows the number of times each strategy was used to find a solution hyperlink recommendation. In the analysis, 175 random records out of the 497 were used, based on the sample size obtained using Equation (1).

TABLE I
STRATEGY COMPARISON

Strategy	Amount of times used	Records found
Search text	24	14%
Database search	0	0%
Broken link search	57	33%
Search in archive.org and google.com	54	31%
No results	40	23%

Table 2 shows the results of a human verification of the obtained results that classified each one as correct or incorrect.

TABLE II
SEARCH ACCURACY

Strategy	Correct results	Accuracy
Text search	6	25%
Database search	0	0
Broken link search	32	56%
Search in archive.org and Google.com	39	72%

[2] https://www.python.org/
[3] https://www.crummy.com/software/BeautifulSoup/
[4] https://git-scm.com/
[5] https://docs.djangoproject.com/en/dev/topics/testing/tools/

B. Lecture at Villadevs

Villadevs is an open forum for the discussion of the latest technology trends and developments which takes place in Villavicencio, Colombia, and is sponsored by several public and private institutions. Fig. 6 shows the authors presenting the work in progress. As result, authors profited from the comments and discussion during this event that led to the refinement of the strategies.

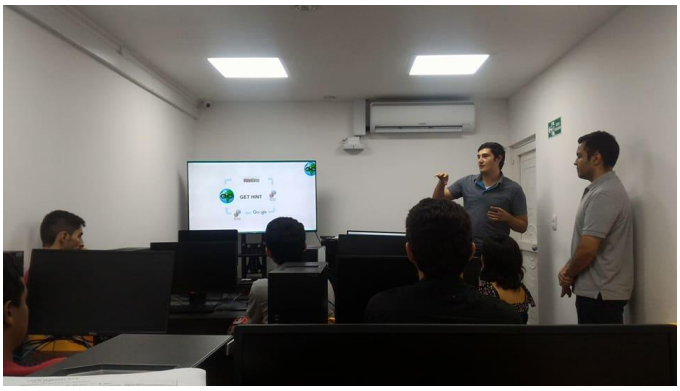

Fig. 6 Authors presenting and discussing algorithms at Villadevs. Source: Villadevs.

V. CONCLUSIONS

Web Scraping is a powerful technique for retrieving information from web pages or hypertext files. Nevertheless, it presents some limitations when dealing with websites developed using frameworks such as *AngularJS*, *React, Vue, Laravel, Meteor* and *Polymer*, among other modern frameworks, as these load their content through *Ajax* requests.

Among the suggested solutions produced by the different strategies there were some special cases where the result was the same as the broken link provided. These results were classified as false-positives and obviously were not taken into account as correct solutions.

When similar results appear it may be useful to carry out a semantic search using the content from the set of possible hyperlink solutions and compare the findings with the results from the search of the broken hyperlink on 'archive.org'.

Omitting descriptive text in <a> HTML tags is a bad practice of web editors that prevents the *'Text search on the Internet'* strategy from producing satisfactory results.

Hyperlinks life cycle varies depending on the type of websites they are located in. According to the website content some trends showed up:

- Pages about events tend to disappear after the event date passes.
- Institutional, governmental, educational and informative sites tend to be relocated year by year, most of the time in randomly unexpected ways.
- Personal web pages change domains more frequently, although their content does not change a lot.

For this project, the search engines used were only 'Google' and 'archive.org'; for future versions of the application different internet search engines such as 'Bing', 'DuckDuckGo', 'Yahoo' and others should be included in the system to obtain a more diverse set of results and avoid inconveniences like recurrent search blocking.

Finally, the authors think that storing in a database keywords related to the active hyperlinks in a website could be of great help where one of those hyperlinks is found broken as there would be relevant information available to carry out the search for the solution or replacement of it.

ACKNOWLEDGMENT

The authors want to thank all the people that made this project possible, especially their families and colleagues.

Special thanks go to Jerson Pabón and the entire Mobile Corp S.A.S team for logistic support and Juan Vargas for his collaboration in the translation of the paper.

DISCLAIMER

Some of the algorithms implemented in this work are under Provisional Application by the United States Patent and Trade Mark Office (USPTO), Application Number 62582968.

REFERENCES

[1] D. Fetterly, M. Manasse, M. Najork, and J. Wiener, "A large-scale study of the evolution of web pages," in Proceedings of the twelfth international conference on World Wide Web - WWW '03, 2003, p. 669.

[2] D. W. Brooks and J. Markwell, "Broken Links: The Ephemeral Nature of Educational WWW Hyperlinks," J. Sci. Educ. Technol., vol. 11, no. 2, pp. 105–108, 2002.

[3] A. Morishima, A. Nakamizo, T. Iida, S. Sugimoto, and H. Kitagawa, "PageChaser: A Tool for the Automatic Correction of Broken Web Links," in 2008 IEEE 24th International Conference on Data Engineering, 2008, pp. 1486–1488.

[4] J. Martinez-Romo and L. Araujo, "Updating broken web links: An automatic recommendation system," Inf. Process. Manag., vol. 48, no. 2, pp. 183–203, Mar. 2012.

[5] A. Liptak, "In Supreme Court Opinions, Web Links to Nowhere," 2013.

[6] F. McCown, C. Sheffan, M. L. Nelson, and J. Bollen, "The Availability and Persistence of Web References in D-Lib Magazine," in 5th International Web Archiving Workshop (IWAW05), 2005.

[7] P. I. M. Torres, I. K. Paz, I. Federico, and G. Salazar, "TAMAÑO DE UNA MUESTRA PARA UNA INVESTIGACIÓN DE MERCADO. "Manuscript Templates for Conference Proceedings, IEEE. http://www.ieee.org/conferences_events/conferences/publishing/templates.html

[8] archive.org, "Internet Archive: About IA," *archive.org*, 2017. [Online]. Available: https://archive.org/about/. [Accessed: 22-Nov-2017].

[9] J. Lepore, "The cobweb - can the internet be archived?" *The New Yorker*, Jan. 2015. [Online]. Available: https://www.newyorker.com/magazine/2015/01/26/cobweb [Accessed: 30-Nov-2017]

[10] Mateo Santos, "SaaS, IaaS y PaaS: ¿qué son, cómo usarlos y para qué? • ENTER.CO," *enter.co*, 2015. [Online]. Available: http://www.enter.co/guias/tecnoguias-para-empresas/saas-iaas-y-paas-que-son-como-usarlos-y-para-que/. [Accessed: 24-Nov-2017].

[11] N. George, *Mastering Django core*, 1st ed. GNW Independent Publishing, 2016.

[12] M. Lutz, *Learning Python*, 5th ed. Sebastopol (California, USA): O'Reilly, 2013.

[13] R. Mitchell, *Web Scraping with Python*, 1st ed. O'Reilly Media, 2015.

[14] A. T. Hernández, E. Gómez Vázquez, A. B. Rincón, J. Montero García, A. C. Maldonado, and R. Ibarra Orozco, "Metodologías para análisis político utilizando Web Scraping," Res. Comput. Sci., vol. 95, pp. 113–121, 2015.

[15] K. Reitz, "Requests: HTTP for Humans — Requests 2.18.4 documentation", Docs.python-requests.org, 2016. [Online]. Available: http://docs.python-requests.org/en/master/. [Accessed: 30- May- 2018].