Computer Science Theses & Dissertations   Computer Science

Spring 2019

Expanding the Usage of Web Archives by Recommending Archived Webpages Using Only the URI

Lulwah M. Alkwai
Old Dominion University

EXPANDING THE USAGE OF WEB ARCHIVES BY RECOMMENDING ARCHIVED WEBPAGES USING ONLY THE URI

by

Lulwah M. Alkwai
B.S. July 2007, King Saud University, Saudi Arabia
M.S. August 2013, Old Dominion University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
May 2019

Approved by:

Michele C. Weigle (Director)

Michael L. Nelson (Member)

Jian Wu (Member)

Sampath Jayarathna (Member)

Christian Zemlin (Member)

ABSTRACT

EXPANDING THE USAGE OF WEB ARCHIVES BY RECOMMENDING ARCHIVED WEBPAGES USING ONLY THE URI

Lulwah M. Alkwai
Old Dominion University, 2019
Director: Dr. Michele C. Weigle

Web archives are a window to view past versions of webpages. When a user requests a webpage on the live Web, such as `http://tripadvisor.com/where_to_travel/`, the webpage may not be found, which results in an HyperText Transfer Protocol (HTTP) 404 response. The user then may search for the webpage in a Web archive, such as the Internet Archive. Unfortunately, if this page had never been archived, the user will not be able to view the page, nor will the user gain any information on other webpages that have similar content in the archive, such as the archived webpage `http://classy-travel.net`. Similarly, if the user requests the webpage `http://hokiesports.com/football/` from the Internet Archive, the user will only find the requested webpage, and the user will not gain any information on other webpages that have similar content in the archive, such as the archived webpage `http://techsideline.com`. In this research, we will build a model for selecting and ranking possible recommended webpages at a Web archive. This is to enhance both HTTP 404 responses and HTTP 200 responses by surfacing webpages in the archive that the user may not know existed. First, we detect semantics in the requested Uniform Resource Identifier (URI). Next, we classify the URI using an ontology, such as DMOZ or any website directory. Finally, we filter and rank candidates based on several features, such as archival quality, webpage popularity, temporal similarity, and content similarity. We measure the performance of each step using different techniques, including calculating the F_1 to measure of different tokenization methods and the classification. We tested the model using human evaluation to determine if we could classify and find recommendations for a sample of requests from the Internet Archive's Wayback Machine access log. Overall, when selecting the full categorization, reviewers agreed with 80.3% of the recommendations, which is much higher than "do not agree" and "I do not know". This indicates the reviewer is more likely to agree on the recommendations when selecting the full categorization.

But when selecting the first level only, reviewers only agreed with 25.5% of the recommendations. This indicates that having deep level categorization improves the performance of finding relevant recommendations.

I dedicate this dissertation to my father who had always supported and encouraged me to move forward, and to my mother for her everlasting love and support. I also dedicate my work to my beloved husband and my amazing daughters who have been my motivation and inspiration. I am grateful to have them in my life.

ACKNOWLEDGEMENTS

First, I am thankful to Almighty Allah for all the blessings he bestowed on me, Alhamdulillah. I am thankful for the completion of this academic journey. I am thankful for the support and guidance from the people around me.

I am deeply thankful to my great advisor Dr. Weigle for her guidance, support, valuable feedback, and encouragement. I cannot thank her enough for her patience and role in my PhD journey. I would also like to thank my great co-advisor Dr. Nelson for his guidance and support. I am grateful for both Dr. Weigle and Dr. Nelson, they provided me with academic mentoring and research guidance through the years.

I would also want to express my appreciation for the members of the committee Dr. Wu, Dr. Jayarathna, and Dr. Zemlin for their support and input to enhance the dissertation. I am grateful for Dr. Hussein Abdel-Wahab who passed away in 2016, God bless his soul. For his help and support from the first day I joined the department. His words and kindness will never be forgotten.

I would like thank my colleagues at the web science and digital library group, for their support and insightful discussions, to Sawood, Mat, Alex, Mohammed, Shawn, and the reset of the wsdl group. To all former wsdl PhD graduates and friends, especially to my best friend Yasmin AlNoamany who always believed in me and encouraged me to move forward. I would also want to thank Lamia Alkwai, Hend Aldabagh, Abrar Alali, Alexander Nwala, Sawood Alam, and Mat Kelly, for taking their time by helping me in evaluating my work and providing feedback.

I would like to thank the ODU Computer Science department for providing me with opportunities and support during my masters and PhD. In addition, I am thankful to University of Hail for their scholarship, and to my country for funding my education.

I am thankful to my family for their love and support. My parents, Mohammed, and Zahwah, for their endless support, encouragement, their prayers, and words of wisdom. They were there for me in every step of my journey, they felt my stress and emotions, without me saying anything. To my sisters Dr. Hend and Dr. Hala, who were alway there for me and who helped me throughout the years when I needed some medical advise, especially when my kids were sick. To my computer science sister Lamia, who is always ready to share thoughts and talk computer. To my sister

Sarah and my brother Khaled, for their kindness, support, and encouragement. To my uncle Fahad who passed away in 2014, God bless his soul. He always put a smile on my face, he called me daily during the start of my academic journey to help me feel at home. He was a second father to me. I will never forget the stories he told me and his words of wisdom.

Most importantly, to my dear husband, Khaled Aljannakh, for his love, selfless support, encouraging me to reach my dreams, and always being there for me. I am blessed to have him in my life. To my daughters, Judy (9) and Jenna (5), who I am very proud of, they made this journey very special.

To everyone, thank you again!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

Web archives provide valuable services in allowing users to view past versions of webpages, even those that are no longer on the live Web. But not all webpages have been archived, so requests to an archive may return an HyperText Transfer Protocol (HTTP) 404 Page Not Found status. Because the majority of Web archives offer only query by a Uniform Resource Identifier (URI) [9], users cannot search archives by page content to look for alternate archived pages.

Consider a user who had previously bookmarked the travel webpage `http://www.tripadvisor.com/where_to_travel/`. When the user tries to view the page on the live Web, they find it is no longer available (Figure 1), the webpage results in a customized error page (Section 2.2.4). Then, because they are aware of Web archiving, the user queries a Web archive, such as the Internet Archive, for the URI. Unfortunately, this page had never been archived, so the archive will return a "not found" page (Figure 2).

What if, instead, the archive could provide recommendations for other archived pages that might be related to the requested `tripadvisor.com` page? Pages such as `http://www.classy-travel.net` (Figure 3) or `http://www.travelassist.com/mag/mag_home.html` (Figure 4) might be acceptable substitutes.

In addition to providing recommendations for pages that have not been archived, the archive could also make recommendations for related pages even when the requested page is found. These related pages could potentially be hidden gems, pages that may not have been popular but are still quality pages. This is one way to reveal "cold spots" in the archive and introduce previously undiscovered, or under-utilized, archived pages to new users.
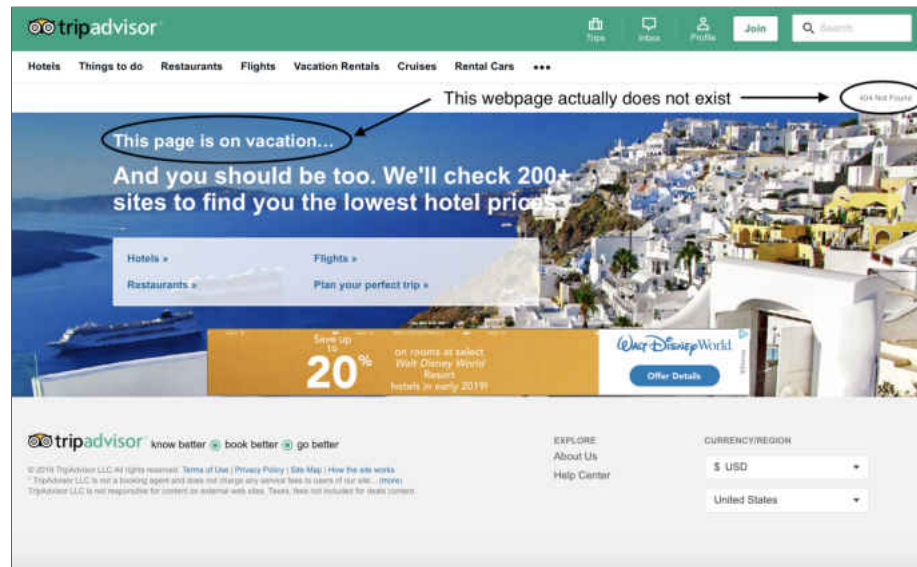
Fig. 1 Webpage not found in the live Web
`http://www.tripadvisor.com/where_to_travel/`
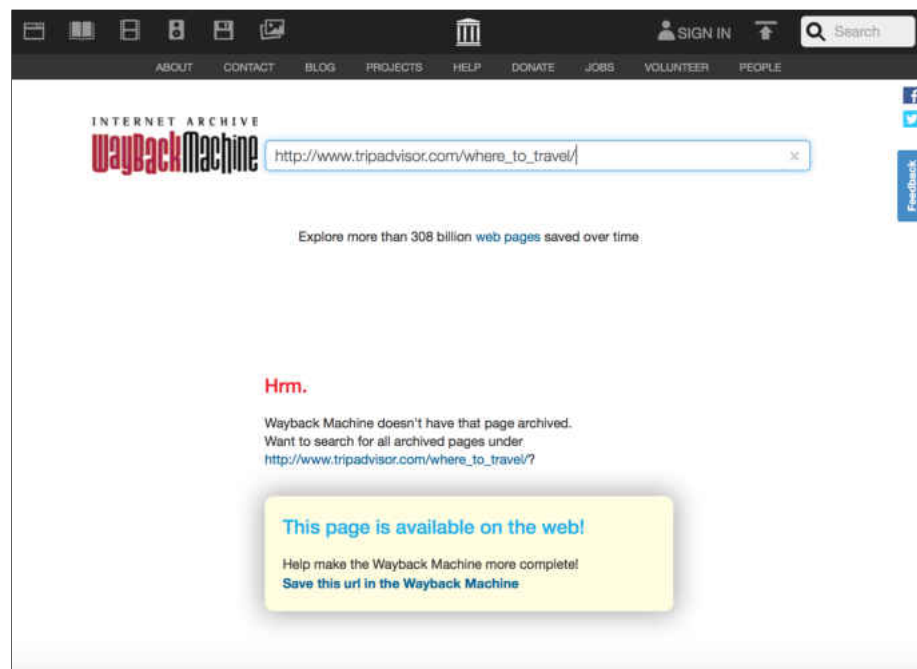
Fig. 2 Webpage not found in the archive
`http://www.tripadvisor.com/where_to_travel/`

Fig. 3 Potential recommendation for the unarchived page from Figure 2, `https://web.archive.org/web/20070409024219/http://classy-travel.net`

Fig. 4 Another potential recommendation for the unarchived page from Figure 2, `https://web.archive.org/web/19961130090356/http://travelassist.com/mag/mag_home.html`
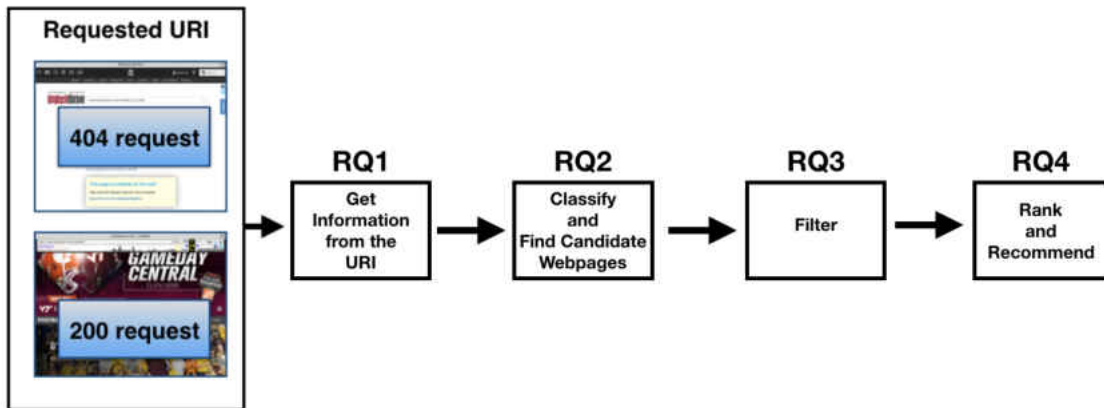
Fig. 5 General approach for recommending archived webpages

1.2 RESEARCH GOALS

The goal of this research is to develop a method to allow Web archives to provide recommendations for user queries, potentially surfacing high-quality, but underutilized webpages. Ideally, we want to find webpages that are not on the live Web and may not have been requested frequently from the archive. We focus on techniques that use only the text of the provided URI and hints from ontologies of webpages (e.g., webpage directories).

Our main research question is: **Given a URI, how can we build a model for selecting and ranking possible recommendation webpages at a Web archive?**

Our general approach is shown in Figure 5, where each box corresponds to one research question proposed here.

RQ1: Given a URI, how much information can we learn about the referenced webpage without examining the content of the page?

Learning about the referenced webpage without examining the content of the page can be challenging due to little context and content available. We use the URI, which is a string of characters used to identify a resource. There are several advantages to using the URI over using the content of the webpage. First, in some cases the content of the URI is not available on the live Web or in the archive. Second, the URI may contain hints about the resource it identifies. Third, it is more efficient both in time and space to use only the URI text than to extract its content. Fourth,

some webpages have little or no textual content, such as containing only an image or a video, so extracting the content will be not useful. Finally, some webpages have privacy settings that do not permit them to be archived.

We investigated different URI tokenization methods that aim to collect meaningful terms, so that we have enough information to know what the webpage is about. This led to the detection of common URI stop words that should be removed from a URI keyword list, such as "pages" and "index".

RQ2: Given a set of terms that describes a webpage, how do we map the terms to an ontology of webpages?

An ontology is a set of categories that contain a group of related instances (documents). In our work, we will focus on categorized webpages. Common webpage ontologies are DMOZ and Wikipedia. Mapping terms to an ontology can result in challenges that we have to address. One challenge is selecting the proper ontology for our work. We use existing ontologies and build on them as we move forward. Keywords extracted from the URI may have matching terms in the ontology. In this case, we used URIs that have matching tokens as candidates. An example is extracting tokens from the URI `http://www.tripadvisor.com/where_to_travel/` to {trip, advisor, where to travel}, and matching them to DMOZ tokens, which will lead to classifying the URI to the category `Recreation/Travel/Guides_and_Directories/`. This will result in finding related webpages in that directory such as `http://classy-travel.net` and `http://travelassist.com/mag/mag_home.html`. We can also use machine learning techniques to predict the category of the requested URI, then use the URIs that are in the same category as the requested URI as candidates for recommendation. This requires an evaluation of different machine learning techniques and choosing the proper one for our work. Also, some ontologies could contain different parts, such as a categorization, URI, title, and description, and we need to decide if we want to match tokens from one part, all parts, or favor one part over the other.

RQ3: Given a set of webpages discovered through the ontology, what metrics do we use to filter the best candidates for recommendation?

The goal of filtering is to keep the desired dataset based on specific feature selections. As an initial step to filter candidates, we removed any candidates that are not archived, since we want to recommend archived webpages only.

RQ4: Given a set of candidate webpages for recommendation, how do we rank the candidates?

The goal of ranking is to sort related candidates in ascending or descending order based on some preset features. There are many features we could consider when selecting the best candidates. The weights of these features must be adjusted to rank them. Some features we consider include archival quality, webpage popularity, temporal similarity, and URI similarity.

Archival quality refers to measuring memento damage [10] by evaluating the number of missing resources in a webpage, which will be discussed further in Section 3.5.1.

Webpage popularity considers how often the webpage has been archived, its popularity on the live Web. A special case of popularity are webpages in "cold spots", which are pages that are not on the live Web, are not currently popular, but are archived. Cold spots will be discussed further in Section 2.1.4.

Temporal similarity can refer to how close the candidate webpage's Memento-Datetime is to the requested URI.

URI similarity assesses the similarity of candidate URI tokens to the requested URI tokens.

1.3 EXAMPLE RECOMMENDATIONS

Here we show four scenarios based on the amount of information that can be obtained solely from the URI itself and the existence of the webpage in both the live Web and the archive. An overview of the examples mentioned is shown in Table 1.

Table 1 An overview of the examples

	On the Live Web	In the Archive	URI Match in an Ontology	Domain Match in an Ontology
Example 1- Virginia Tech Sports Webpage (Section 1.3.1)	✓	✓	✓	✓
Example 2 - Local News Webpage (Section 1.3.2)	✓	✗	✗	✓
Example 3 - History of Heart Transplantation Webpage (Section 1.3.3)	✗	✓	✗	✗
Example 4 - Travel Webpage (Section 1.3.4)	✗	✗	✗	✓

1.3.1 EXAMPLE OF REQUESTING A VIRGINIA TECH SPORTS WEB-PAGE

Consider that a user requests the following URI-M from the Internet Archive (Section 2.1.1): `https://web.archive.org/web/20090105102551/www.hokiesports.com/football/`

The request is for the archived version of `https://www.hokiesports.com/football/`, a webpage about football at Virginia Tech.

The format of the request means the following:

- The archive choice is the Internet Archive, `https://web.archive.org/`

- The requested URI-R is `www.hokiesports.com/football/`

- The datetime is `20090105102551`. It is in the format `YYYYMMDDHHMMSS`, so it is for January 5, 2009, at the time 10:25:51.

We used the curl command to discover if the requested webpage is on the live Web. In this example, the response was a `200 OK`, meaning that the webpage is currently live (Listing 1 and Figure 7). We also used curl to check if the webpage is in the archive, and we got a `200 OK` response code meaning that the Internet Archive has the webpage (Listing 2 and Figure 6). In this example, although the webpage is live and in the archive, we want to find other archived webpages that have similar content.

Listing 1 Command line result of checking if the webpage is on the live Web

```
$ curl -I -L "http://www.hokiesports.com/football/"
HTTP/1.1 200 OK
Date: Thu, 21 Sep 2017 18:46:23 GMT
Server: Apache
X-Frame-Options: SAMEORIGIN
Accept-Ranges: bytes
Vary: Accept-Encoding
X-XSS-Protection: 1; mode=block
Content-Type: text/html
```

Listing 2 Command line result of checking if the webpage is in the archive

```
$ curl -I -L "https://web.archive.org/web
    /20090105102551/www.hokiesports.com/football/"
HTTP/1.1 200 OK
Server: Tengine/2.1.0
Date: Thu, 21 Sep 2017 18:34:29 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 47309
Connection: keep-alive
X-Archive-Orig-date: Mon, 05 Jan 2009 10:25:51 GMT
X-Archive-Orig-accept-ranges: bytes
X-Archive-Orig-connection: close
X-Archive-Orig-server: Apache/2.2.3 (CentOS)
X-Archive-Guessed-Charset: utf-8
Memento-Datetime: Mon, 05 Jan 2009 10:25:51 GMT
```
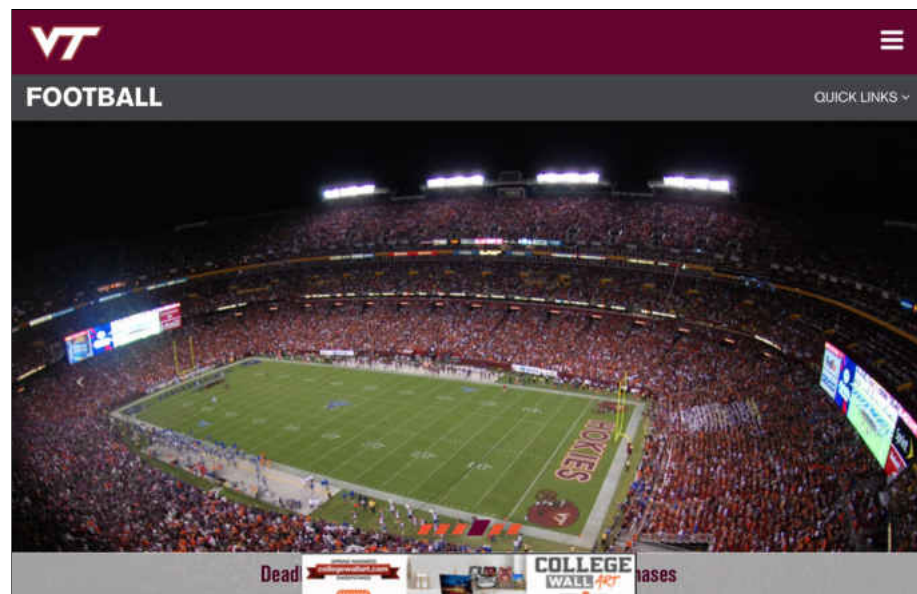

Fig. 6 The Webpage `http://www.hokiesports.com/football/` found in the live Web, accessed on Thursday, 21 Sep 2017 18:34:29 GMT
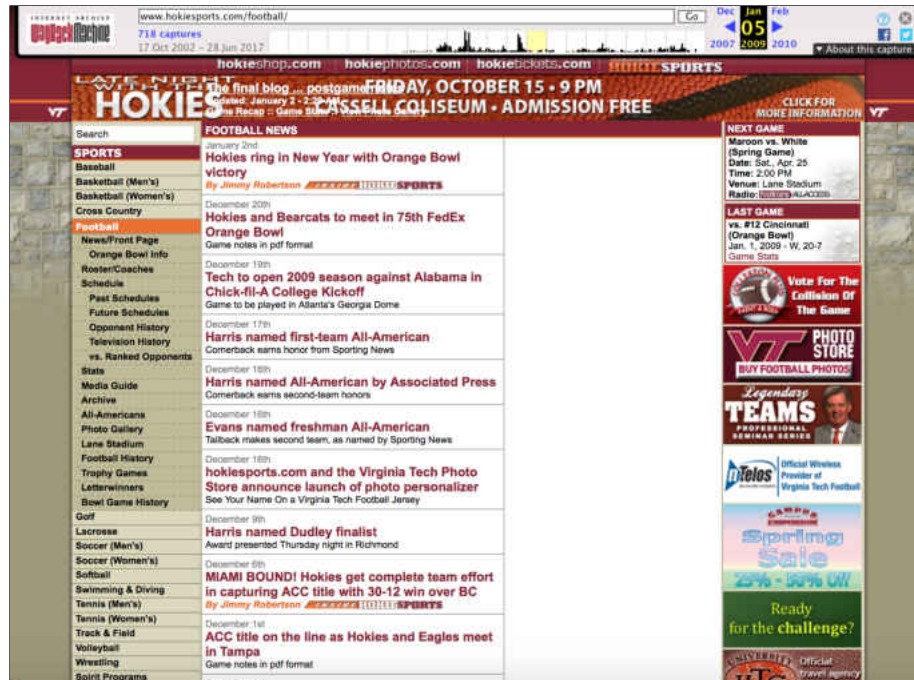
Fig. 7 Internet Archive response to the archived webpage `https://web.archive.org/web/20090105102551/www.hokiesports.com/football/`

From the requested URI, we can generate only the tokens {hokie, sports, football}, which may not be enough to describe the requested webpage. To expand our idea of what the webpage is about, we search the DMOZ directory for the requested URI. The URI is found in the category `Sports/Football/American/College_and_University/NCAA_Division_I-A/ACC/Virginia_Tech`.

Next, we can gather as candidates the list of other webpages in this same category. For each of these candidates, because we are only interested in recommending pages from the archive, we will discard any that have not been archived. The resulting list of candidates is shown in Table 2. From this set of archived candidate recommendations,

Table 2 URIs in the same classification as `http://www.hokiesports.com/football/` found in DMOZ

`http://www.hokiesports.com/football/`
`http://www.TailgateFever.com`
`http://www.sportsline.com/collegefootball/teams/page/VATECH`
`http://www.roanoke.com/sports/vtfootball/`
`http://www.usatoday.com/sports/college/football/acc/vatech.htm`
`http://www.topix.com/ncaa/virginia-tech-football`
`http://www.sportingnews.com`
`http://www.techsideline.com`

we then would apply our proposed ranking algorithm by adjusting the weights of the features we will measure. The ranking algorithm will be discussed further in Section 5.6. In this example, we favor the top three webpages that are high quality, in a cold spot, have similar content, and are in the same time frame as the requested URI. The resulting list of recommendations is shown in Table 3, and the archived webpages are shown in Figures 8(a), 8(b), and 8(c).

A summary of analyzing the requested URI is shown in Table 4.

Table 3 Ranked recommended URIs

1	`https://web.archive.org/web/20081216011510/http://www.techsideline.com:80/home`
2	`https://web.archive.org/web/20080705203833/http://tailgatefever.com/`
3	`https://web.archive.org/web/20031004235141/http://sportingnews.com:80/`

Table 4 Summary of analyzing the Virginia Tech sports webpage

Requested URI	https://web.archive.org/web/20090105102551/www.hokiesports.com/football/
Status	**In the Live Web**: Yes **In the Archive**: Yes
URI Tokens	**Domain**: hokie, sports **TLD**: com **Path**: football
Classification	**DMOZ**: Sports/Football/American/College_and_University/NCAA_I-A/ACC/Virginia_Tech
Candidate Recommendations	- https://web.archive.org/web/20090105102551/www.hokiesports.com/football/ - https://web.archive.org/web/20080705203833/http://tailgatefever.com - https://web.archive.org/web/20090105225712/http://cbssports.com/collegefootball/teams/page/VATECH - https://web.archive.org/web/20150226105515/http://ww2.roanoke.com/sports/vtfootball/ - https://web.archive.org/web/20090106055721/http://www.usatoday.com/sports/college/football/acc/vatech.htm - https://web.archive.org/web/20081118131227/http://www.topix.com/ncaa/virginia-tech-football - https://web.archive.org/web/20031004235141/http://sportingnews.com:80 - https://web.archive.org/web/20081216011510/http://www.techsideline.com:80/home
Ranked Recommendations	**1.** https://web.archive.org/web/20081216011510/http://www.techsideline.com:80/home **2.** https://web.archive.org/web/20080705203833/http://tailgatefever.com **3.** https://web.archive.org/web/20031004235141/http://sportingnews.com:80

(a) A candidate webpage in the Internet Archive that has been detected due to being in the same classification in DMOZ as the requested URI, `https://web.archive.org/web/2008121601 1510/http://www.techsideline.com:80/home`

(b) A candidate webpage in the Internet Archive that has been detected due to being in the same classification in DMOZ as the requested URI, `https://web.archive.org/web/2008070520 3833/http://tailgatefever.com/`

(c) A candidate webpage in the Internet Archive that has been detected by matching tokens from the URI and from an archived list of URIs, `https://web.archive.org/web/20031004 235141/http://sportingnews.com:80/`

Fig. 8 Recommendations for the requested URI `https://web.archive.org/web/20090105102551/www.hokiesports.com/football/`

1.3.2 EXAMPLE OF REQUESTING A LOCAL NEWS WEBPAGE

Consider that a user requests the following webpage from the Internet Archive: `ht tps://web.archive.org/web/20170405104555/https://pilotonline.com/news/ local/transportation/more-people-are-using-the-portsmouth-norfolk-ferr y-and-now/article_8f48e841-0c60-5ad6-b352-576b0eb80b78.html`

The request is for the archived version of a local news article about using the ferry between Portsmouth and Norfolk, Virginia. The requested URI contained meaningful tokens, {transportation, ferry, Portsmouth, Norfolk}. However, it also contained meaningless tokens such as the article number. In this case, the requested URI is not archived but exists on the live Web. The exact URI was not found in DMOZ, but the top-level webpage `https://pilotonline.com/` was. We can use the title and classification of the classified top-level webpage as additional tokens. For finding candidate recommendations, we can match the requested URI tokens. In this example, we favor the top three webpages that are high quality, in a cold spot, have similar content, and are in the same time frame as the requested URI. The summary of analyzing the requested URI including the resulting list of recommendations is shown in Table 5.

Table 5 Summary of analyzing the local news webpage

Requested URI	https://web.archive.org/web/20170405104555/https://pilotonline.com/news/local/ transportation/more-people-are-using-the-portsmouth-norfolk-ferry-and-now/article_ 8f48e8410c605ad6b352576b0eb80b78.html
Status	**In the Live Web**: Yes **In the Archive**: No
URI Tokens	**Domain**: pilot, online **TLD**: com **Path**: news, local, transportation, more people are using the portsmouth norfolk ferry and now,article 8f48e841-0c60-5ad6-b352-576b0eb80b78
Classification	**DMOZ**: top-level webpage: http://www.pilotonline.com/ Regional/North_America/United_States/Virginia/Metro_Areas/Hampton_Roads/ News_and_Media/Newspapers
Candidate Recommendations	- https://web.archive.org/web/20170709083750/https://gohrt.com/services/ferry/ - https://web.archive.org/web/20160818230621/https://www.virginia.org/listings/ outdoorsandsports/elizabethriverferry - https://web.archive.org/web/20150913155913/http://www.norfolk-va.worldweb.com/ Transportation/Ferries/ - https://web.archive.org/web/20170613205542/https://www.dfdsseaways.co.uk/ - https://web.archive.org/web/20170706102019/www.virginiaplaces.org/transportation/ ferries.html
Ranked Recommendations	**1.** https://web.archive.org/web/20150913155913/http://www.norfolk-va.worldweb.com/ Transportation/Ferries/ **2.** https://web.archive.org/web/20160818230621/https://www.virginia.org/listings/ outdoorsandsports/elizabethriverferry **3.** https://web.archive.org/web/20170706102019/www.virginiaplaces.org/transportation/ ferries.html

1.3.3 EXAMPLE OF REQUESTING A HISTORY OF HEART TRANS-PLANTATION WEBPAGE

Consider that a user requests the following webpage from the Internet Archive: `ht tps://web.archive.org/web/20170405104555/http://www.heart-transplant.o rg/history/`

The request is for the archived version of a webpage about the history of heart transplantation. In this case, the requested URI is not live but is archived. Neither the domain nor the URI exists in DMOZ. For finding candidate recommendations, we can match the requested URI tokens to tokens extracted from DMOZ. Assume we only found four URIs that contain matching tokens {heart, transplant}. In this example, we favor the top three webpages that are high quality, in a cold spot, have similar content, and are in the same time frame as requested URI. See Table 6 for the summarized analysis of the requested webpage.

Table 6 Summary of analyzing the history of heart transplantation Webpage

Requested URI	https://web.archive.org/web/20170405104555/http://www.heart-transplant.org/history/
Status	**In the Live Web**: No **In the Archive**: Yes
URI Tokens	**Domain**: heart, transplant **TLD**: org **Path**: history
Classification	**DMOZ**: None
Candidate Recommendations	- https://web.archive.org/web/20151011034215/https://www.newscientist.com/article/ mg20227114.600-hybrid-hearts-could-solve-transplant-shortage - https://web.archive.org/web/20170126172909/http://www.heart-transplant.co.uk/ - http://web.archive.org/web/20120615191355/http://mayoclinic.org/heart-transplant/ index.html - http://web.archive.org/web/20111006094921/http://newshawksreview.com/ arizona-heart-transplant-patient-helped-by-dental-angels/42455/
Ranked Recommendations	**1.** http://web.archive.org/web/20120615191355/http://mayoclinic.org/heart-transplant/index.html **2.** http://web.archive.org/web/20111006094921/http://newshawksreview.com/ arizona-heart-transplant-patient-helped-by-dental-angels/42455/ **3.** https://web.archive.org/web/20170126172909/http://www.heart-transplant.co.uk/

1.3.4 EXAMPLE OF REQUESTING A TRAVEL WEBPAGE

For the final example, consider that a user requests the following webpage from the Internet Archive: `https://web.archive.org/web/19971010080303/http://http s://www.tripadvisor.com/where_to_travel`

The request is for the archived version of a webpage that provides travel suggestions. In this case, the requested URI is not live and not archived. The exact

URI was not found in DMOZ, but the top-level webpage `www.tripadvisor.com` was. We use the title and classification of the classified domain as additional tokens. For finding candidate recommendations, we can match the requested URI tokens. In this example, we favor the top three webpages that are high quality, in a cold spot, have similar content, and are in the same time frame as requested URI. See Table 7 for the summarized analysis of the requested webpage.

Table 7 Summary of analyzing the travel webpage

Requested URI	https://web.archive.org/web/19971010080303/http://https://www.tripadvisor.com/where_to_travel
Status	**In the Live Web**: No **In the Archive**: No
URI Tokens	**Domain**: trip, advisor **TLD**: com **Path**: where to travel
Classification	**DMOZ**: top-level webpage: http://www.tripadvisor.com/ Recreation/Travel/Guides_and_Directories
Candidate Recommendations	- https://web.archive.org/web/19961130090356/http://travelassist.com/mag/mag_home.html - https://web.archive.org/web/19990128193256/http://cnn.com:80/TRAVEL/DESTINATIONS/ - https://web.archive.org/web/20070409024219/http://classy-travel.net - https://web.archive.org/web/20080726022035/http://travel-trips.org/
Ranked Recommendations	**1.** https://web.archive.org/web/20070409024219/http://classy-travel.net **2.** https://web.archive.org/web/20080726022035/http://travel-trips.org/ **3.** https://web.archive.org/web/19961130090356/http://travelassist.com/mag/mag_home.html

1.4 RESEARCH QUESTIONS

After providing these motivating examples, we recap our research questions.

RQ1: Given a URI, how much information can we learn about the referenced webpage without examining the content of the page?

RQ2: Given a set of terms that describes a webpage, how do we map the terms to an ontology of webpages?

RQ3: Given a set of webpages discovered through the ontology, what metrics do we use to filter the best candidates for recommendation?

RQ4: Given a set of candidate webpages for recommendation, how do we rank the candidates?

1.5 DISSERTATION ROADMAP

Below is the organization for the remainder of the dissertation.

Chapter 2: Background

To answer the research questions in this chapter we review some basic concepts and terminology used throughout the dissertation. We present Web archiving and explain why it is important. We also explain the Memento framework and the Internet Archive Wayback Machine interface. This addresses RQ3. To gain information from the URI and address RQ1, we discuss the URI structure, semantics in URIs, and the HTTP status code. Next, to address RQ2 we define webpage classification and present its different types. In addition, we give an overview of ontologies and describe some common ontologies. Finally, to address RQ4 we describe recommendations and list its popular usage.

Chapter 3: Related Work

In this chapter, we present the contextualization of our work and prior work. To gain information from the URI and address RQ1, we discuss splitting compound words. Next, we present clustering vs. classification. This addresses RQ2. After that, we present Web archiving related work. We discuss related work regarding archiving API tools, expected lifetime of a webpage, user intention of searching the archive, measuring the archived Web resources, searching the archive without indexing it, recovering missing web resources, soft 404s. This addresses RQ3. Finally, to address RQ4 we describe different features that will help us rank the recommendations. We will discuss temporal similarity, webpage popularity, URI similarity, and archival quality.

Chapter 4: Datasets

This chapter present the datasets we are going to use as our primary steps toward prototyping our framework by exploring and understanding the datasets we use. The datasets described are DMOZ, Wikipedia, and the Wayback Machine access log.

Chapter 5: Implementation

This chapter contains the steps of the framework, where we address the research questions and evaluate the methods for each question. We will focus on four steps. First, we detect URI semantics. Next, we map tokens to an ontology by determining if the requested URI is already present and categorized in a known ontology, DMOZ or Wikipedia. After that, we classify the URI based on the first level category. In the case where a request did not appear in an ontology, we will classify it using only

the tokens from the URI. We test three different methods of tokenization: tokens, all-grams from tokens, and all-grams from URI. We determine the best method by comparing the result of the F_1 score using machine learning. Next, we perform deep-level classification by indexing the dataset, calculating the cosine similarity, collecting N candidates, creating and pruning the tree, and finally classifying it using machine learning. Finally, we filter, rank, and recommend candidates. We filter the candidates to ensure that all recommendations come from a web archive. Then we rank and recommend the remaining candidates based on temporal similarity, webpage popularity, URI similarity, and archival quality.

Chapter 6: Evaluation

This chapter discusses the evaluation of the deep level classification and the effect of different features in the URI such as depth, presence of dictionary words, long strings, and the category that it belongs to. We also evaluate the entire model and present human evaluators with three different datasets all from the Wayback access log. The first dataset are requests that are found in an ontology. The second dataset are requests that are not found in an ontology and have depth zero, i.e., top-level webpages. The third dataset are requests that are not found in an ontology and have a path. We measure the agreement between the evaluators and analyze the results. We present the scores and the results for this model.

Chapter 7: Contributions, conclusions, and future work

In this chapter, we revisit the research questions and summarize how we addressed each question. We also mention other future work opportunities. In addition, we conclude the research results with a summary of our findings.

CHAPTER 2

BACKGROUND

Here we introduce some basic concepts related to our work. First, we present Web archiving and explain why it is important. We also describe the Memento framework and the Internet Archive interfaces. Next, we define cold spots. After that, we discuss URIs, their structure, status codes, and define "cool URIs". Then, we introduce both classification and standard machine learning algorithms. We also give an overview of ontologies and describe DMOZ, a common, well-known ontology. Finally, we describe recommendation systems and list their widespread usage.

2.1 WEB ARCHIVES

It is likely that a live webpage will change or be removed [11, 12]. This will result in a missing resource from the live Web that cannot be recovered. Web archiving is the process of collecting portions of the World Wide Web, preserving the collections in an archival format, and then managing the access of the archival material. The most common Web archiving method is using Web crawlers to automate the process of collecting webpages. Some of the popular Web crawlers are Heritrix [13], Wget[1], WARCreate[2] [14], Webrecorder[3] and HTTrack [15]. There are a variety of organizations that archive websites, including non-profits, the U.S. Government, libraries, and archives. Some existing Web archives include the Internet Archive[4], the UK Web Archive[5], and many others[6] [16, 17, 18, 19, 20, 21]. The Internet Archive [22] is the largest and oldest Web archive in the world dating back to 1996. The archived content is made available and can be accessed and viewed. The archived webpages are considered as snapshots, also known as *mementos*, of the information at particular points in time. Mementos refer to an archived representation of a resource [23].

The Internet Archive uses the Web ARChive (WARC) file format [24] to store Web crawls. The WARC file contains the webpage content in addition to some other

---

[1]`https://gnu.org/software/wget/`
[2]`http://warcreate.com`
[3]`https://webrecorder.io`
[4]`https://archive.org/web/`
[5]`http://webarchive.org.uk/ukwa/`
[6]`https://en.wikipedia.org/wiki/List_of_Web_archiving_initiatives`

details, such as the WARC header information, WARC header metadata, and other information. Each resource has a header containing metadata about how the resource was requested followed by the HTTP header and the response.

A CDX file [25] is an index into a WARC file given a URI-M. Each line of the CDX file represents a single Web document. An example of a CDX entry is the following:

```
odu.edu/compsci  20170101203142 https://www.odu.edu/
    compsci text/html 200
    E6T72C2R6BRRKSI3IZPMRJDXTFJIRC7P - - 123000 ODU
    -000001.warc.gz
```

The fields in the above example are the following:

- Canonicalized URL: `odu.edu/compsci`

- Timestamp: the time when the URI was captured: 20170101203142

- Original URI: `https://www.odu.edu/compsci`

- MIME type of original document: text/html

- HTTP response code: 200

- Checksum: E6T72C2R6BRRKSI3IZPMRJDXTFJIRC7P

- Redirect: -

- Meta tags: -

- Compressed ARC file offset: mark the beginning of file, 123000

- WARC File name: ODU-000001.warc.gz

When searching a Web archive such as the Internet Archive's Wayback Machine, the process includes looking for the requested URI in a CDX file to determine the appropriate WARC files. Then the Wayback Machine accesses the designated WARC file with the specified offset to read the webpage content.

The Internet Archive has developed a metadata file called Web Archive Metadata (WAT)[7] [26], that contains metadata from a WARC file. Usually, a WAT file is used

---

[7]`https://webarchive.jira.com/wiki/display/Iresearch/Web+Archive+Metadata+File+Specification`

for data analysis. A WAT provides a list of links in the page and the type of the link, which might be useful for determining what the page is about.

To extend our work, one of the methods to collect archived candidate recommendations is to use a list of archived webpages that could be in the format of WARC, WAT, CDX, or any other format available. Depending on the information needed we can use the proper archival file format. In some cases, we can use the URI without the need to dereference the webpage. In this case, we can use a CDX file for collecting a list of archived URIs. We can also use some information available such as the timestamp to determine the time of capture, MIME [27] to determine the document type, and HTTP [28] response code to determine if the webpage existed. We can also use the redirect link if the information is available. In other cases, the URI is not informative and we need to use the content to determine what the webpage is about to classify it. In this case, we can access the WARC file because it contains the content of the webpage. Also, we can use the WARC file in testing and evaluating some methods when we need the dereferenced URI. However, in other cases we just need the title of the webpage or the list of links that the URI contains and its anchor text. In this case, we can access the WAT file and get the information we need without dereferencing the webpage.

To give an example of a CDX, WARC, and WAT file, we generated those files for `http://www.cs.odu.edu/~lalkwai/publication.html`. The webpage is my personal publications, shown in Figure 9. It contains the ODU logo, links to my personal publication list, two other links, to my homepage and to the WSDL group.

In the CDX file shown in the Appendix (Listing 11), the first line is the file specification. This is followed by the requests for the two style sheets, the image, and the webpage. This information can be used if we want minimal information such as the archived URI, MIME type, status code, timestamp, or other details. This archival format is useful if minimal computational complexity and storage is required.

In the WARC file shown in the Appendix (Listing 12), we have the header, the metadata, a list of all the out links, the HTML code, and other additional information. This information can be used if we want more information such as the out link list and the HTML code. This archival format requires more computational complexity and storage.

In the WAT file shown in the Appendix (Listing 13), we have the header, the

metadata, and the links along with their anchor text, and other additional information. This information can be used if we want more information on the out links, such as the anchor text and the title of the webpage but we do not need the HTML code. This archival format requires less computational complexity and storage than the WARC file, but it contains more information than a CDX file.
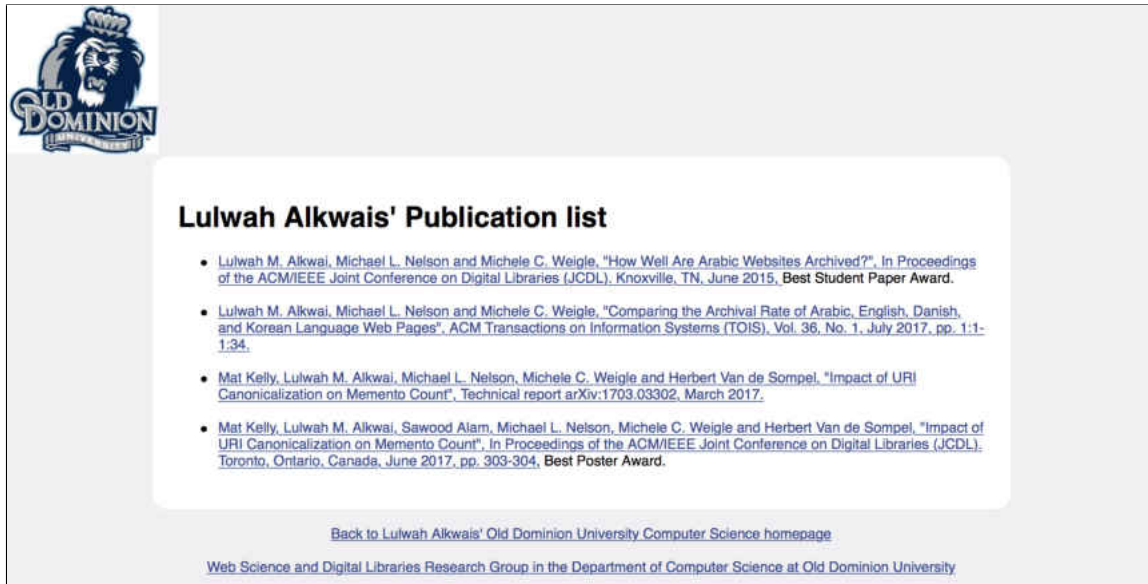
Fig. 9 Webpage that we generated CDX, WARC, and WAT files for (Appendix, Listings 11-13), Source: `http://www.cs.odu.edu/~lalkwai/publication.html`

2.1.1 MEMENTO FRAMEWORK

The Memento Framework [29] defines four types of resources: Original Resource, Memento, TimeGate, and TimeMap.

- An original Resource (URI-R) is a resource as it used to appear on the live Web. This resource may have 0 or more mementos. An example of a URI-R of an original resource is `http://www.cs.odu.edu/~lalkwai/publication.html`.

- A memento (URI-M) is an archived snapshot of the URI-R at a specific date-time, which is called the Memento-Datetime, e.g., $URI\text{-}M_i = URI\text{-}R@t_i$. An example of a URI-M is `https://web.archive.org/web/20190405151649/https://www.cs.odu.edu/~lalkwai/publication.html`.

- A TimeGate (URI-G) allows HTTP content negotiation in the time dimension. An example of a URI-G is `https://memgator.cs.odu.edu/timegate/`

http://www.cs.odu.edu/~lalkwai/publication.html.

- A TimeMap (URI-T) is a resource that is a machine readable list of mementos (URI-Ms) for a URI-R with their respective Memento-Datetimes, An example of a URI-T is:

```
<http://www.cs.odu.edu/~lalkwai/publication.html>; rel=
    "original",
<https://memgator.cs.odu.edu/timemap/link/http://www.cs
    .odu.edu/~lalkwai/publication.html>; rel="self";
    type="application/link-format",
<http://web.archive.org/web/20190405151649/https://www.
    cs.odu.edu/~lalkwai/publication.html>; rel="first
    memento"; datetime="Fri, 05 Apr 2019 15:16:49 GMT",
<http://web.archive.org/web/20190405151649/http://www.
    cs.odu.edu/~lalkwai/publication.html>; rel="last
    memento"; datetime="Fri, 05 Apr 2019 15:16:49 GMT",
<https://memgator.cs.odu.edu/timemap/link/http://www.cs
    .odu.edu/~lalkwai/publication.html>; rel="timemap";
    type="application/link-format",
<https://memgator.cs.odu.edu/timemap/json/http://www.cs
    .odu.edu/~lalkwai/publication.html>; rel="timemap";
    type="application/json",
<https://memgator.cs.odu.edu/timemap/cdxj/http://www.cs
    .odu.edu/~lalkwai/publication.html>; rel="timemap";
    type="application/cdxj+ors",
<https://memgator.cs.odu.edu/timegate/http://www.cs.odu
    .edu/~lalkwai/publication.html>; rel="timegate"
```

Figure 10 shows the relationship between the resources. When a request is made to get an archived version of the URI-R, the URI provides an HTTP link of type TimeGate pointing to the URI-G, which could be on the same or different server as the URI-R. The user requests a specific datetime of the URI-R and the URI-G responds with the proper URI-M, which also could be on the same or different server as the URI-R. The URI-G makes decisions and redirects based on the arguments. URI-Ms can point to each other to allow access of other mementos. Note that both the URI-M and the URI-G point back to the URI-R to allow retracing of steps.
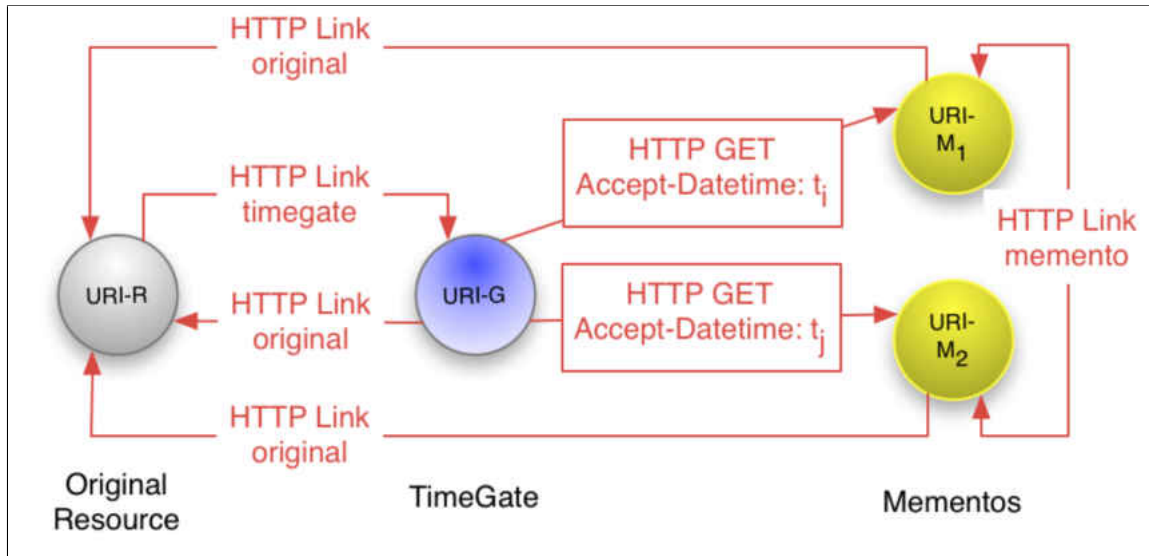
Fig. 10 An architectural overview of how the Memento framework allows accessing a prior version of a resource, Source: `http://mementoweb.org/guide/quick-intro/`

2.1.2 DATETIME OF WEBPAGES

Any webpage created could have some notion of time. Nelson [30] mentioned that each webpage could have:

- Creation-Datetime (CD): The datetime the resource was created

- Last-Modified (LM): The datetime the resource last changed

- Memento-Datetime (MD): The datetime the resource was crawled

- Aboutness Time (AT): The datetime of an event that the page contains.

2.1.3 ARCHIVE ACCESS INTERFACES

There are three common user access interfaces for Web archives: URI-based, collection-based, and full-text search. Currently, there is no interface where we can request a webpage and obtain a recommended list of webpages with similar content.

URI-based

The Internet Archive is implemented with URI-based access through the Wayback Machine. The user enters a URI in the text box and then gets a list of mementos

for that URI (Figure 11(a)). For example, if the user requests an archived version of the Department of Computer Science at ODU `http://cs.odu.edu` (Figure 11(b)), since the webpage is archived, the user will view the following: the number of times it is archived, when it was first and last archived, a histogram showing the number of times the webpage was archived each month, and a calendar that highlights the days the webpage was archived for the selected year. The user can select the memento to view from the calendar.

For example, the user can select the first memento available for the webpage (Figure 12(a)) or the last memento available (Figure 12(b)). Note that in this example the URI-Rs are not the same `http://www.cs.odu.edu` vs. `http://www.odu.edu:80/compsci`, but they have been combined into a single TimeMap.

Collection-based

Web archives can also be queried based on specific events or topics. Archive-It can be queried by URI or by full-text (Figure 13(a)). For example, if the user requested to explore collections that are about "museum", the user enters the query in the search bar (Figure 13(b)). Then, the result would show the number of collecting organizations, the number of collections, and the number of sites. Also, the list of collections with its details would appear below. When the user selects a collection such as the "Arab American National Museum", its detailed information would appear, in this case, a list of five different websites (Figure 13(c)).

Full-text search

With full-text search, archived webpages are queried using a text query. The UK Web Archive provides a URI search, and a word or phrase search. After entering a query, a side bar with filtering options appear. The filtering options are accessing content, domain, document type, suffix, data collection, and topics and themes.

For example, we can search for the terms "Arab Spring" and filter the search by the following: restrict the date to between January 1, 2015 and January 1, 2018, the topics and themes to "News Sites", and select the domain "gov.uk". This search produced 2,730,315 results, each showing the title, the URI, the part of the text where the searched text appeared and the date collected (Figure 14(b)).

In 2016, the Internet Archive added the Beta Wayback Machine [31], where the user can search by entering a URI or words related to a site's top-level webpages
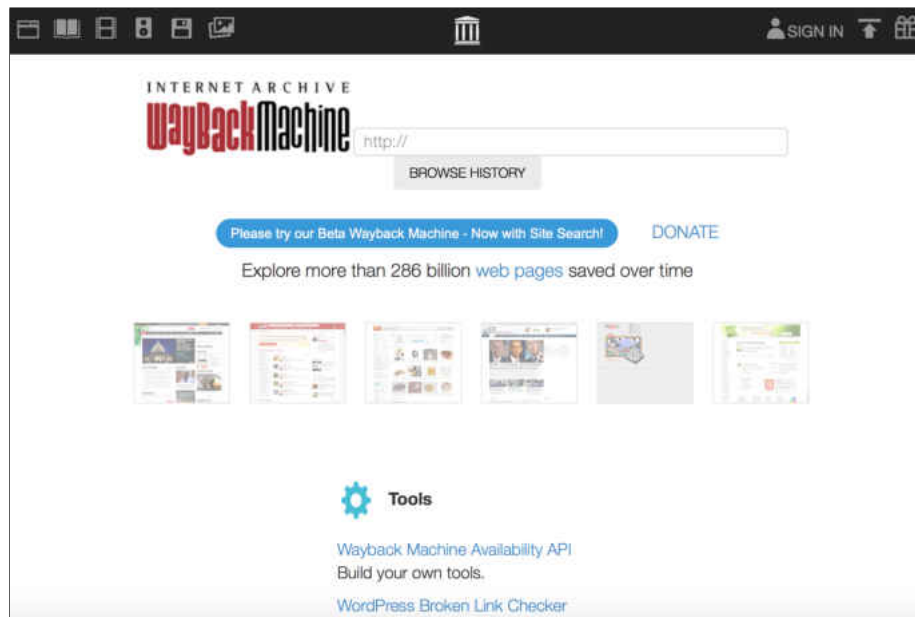
(a) Internet Archive interface
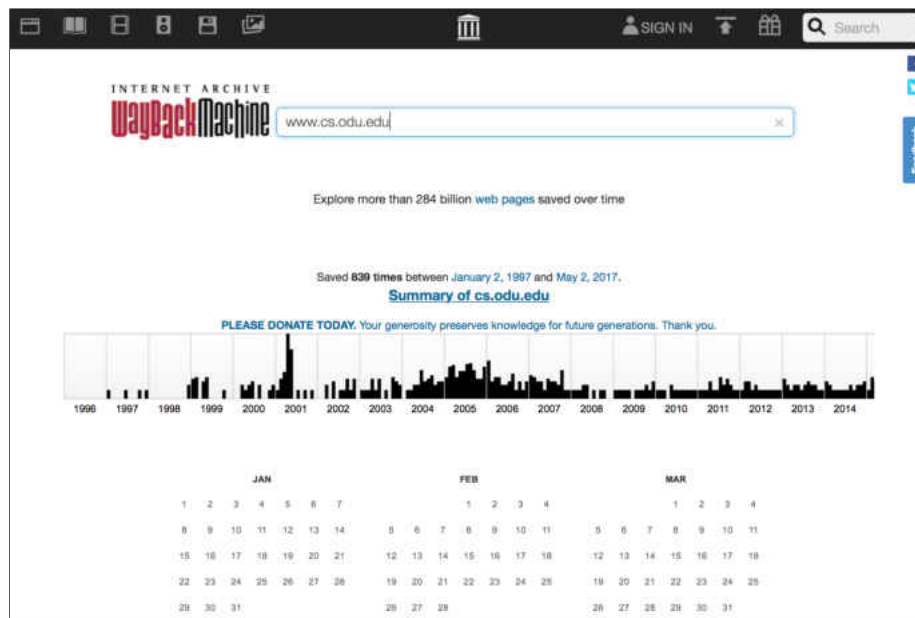
(b) Internet Archive Wayback Machine search for ODU CS webpage, `http://cs.odu.edu`

Fig. 11 Internet Archive URI-based interface

(a) Selecting the first memento for ODU CS webpage, `https://web.archive.org/web/19970102130137/cs.odu.edu`
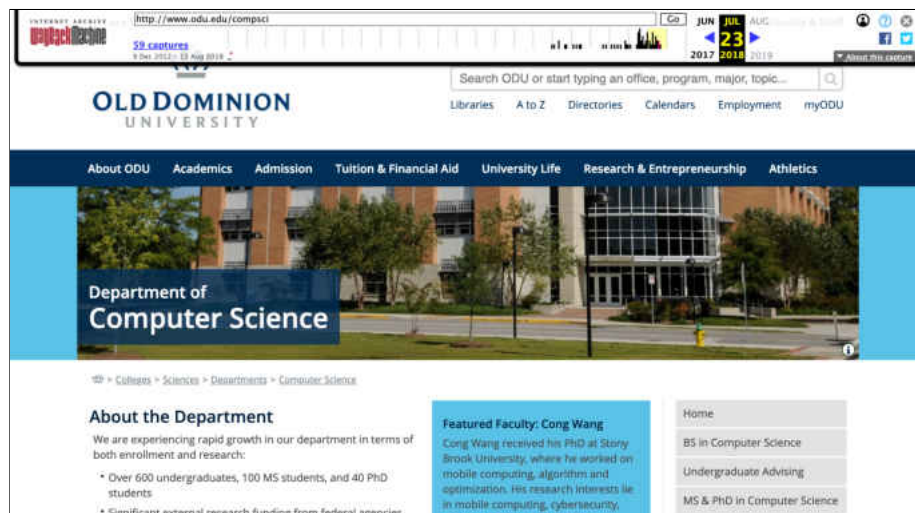
(b) Selecting the last memento for ODU CS webpage, `https://web.archive.org/web/20180723163102/http://odu.edu:80/compsci`

Fig. 12 ODU CS webpage on the Internet Archive the past and the present
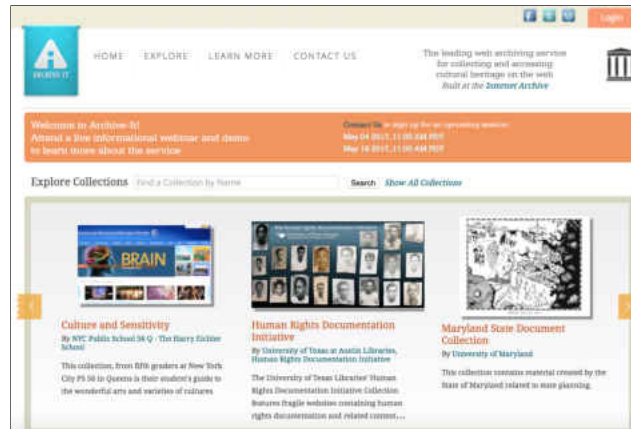
(a) Archive-It interface
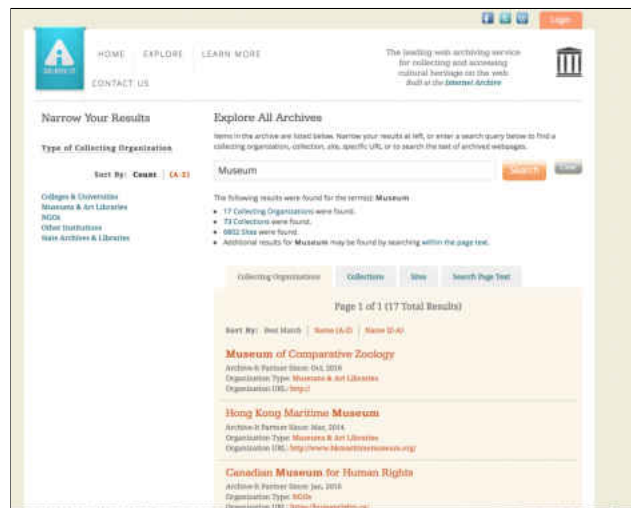
(b) Exploring Archive-It collections after searching for the text 'museum'
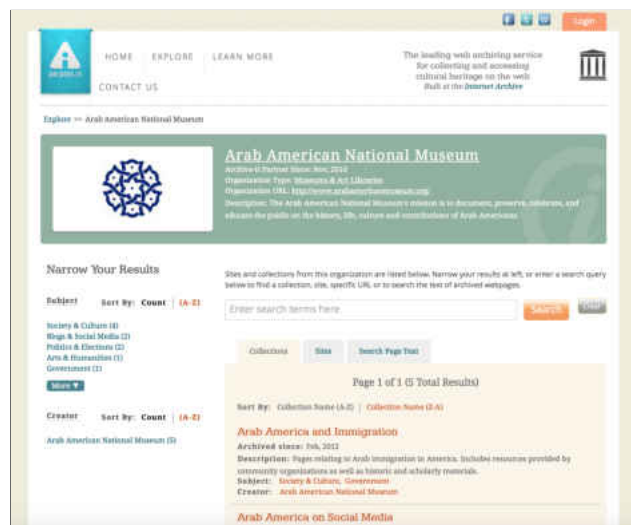
(c) Archive-It collection-listing the 'Arab American National Museum' collection information

Fig. 13 Archive-It collection-based interface

(Figure 15(a)). The search index was built by processing over 250 billion archived webpages [31]. The index contains more than a billion terms collected from over 400 billion hyperlinks to the top-level webpages. The results are ranked based on relevance. However, this is not a real full text index, because it links to the site's home page instead of linking to the exact page containing the searched keyword. In addition, the results of a query is based on matching the query to the words people have used to describe those sites. This requires a webpage description of each site in order to perform the recommendation.

For example, when the user enters a query such as "King Saud University", the user will view a list of top-level webpages where either its URI contains the phrase "King Saud University" or a webpage that had an anchor text that contains the phrase "King Saud University" (Figure 15(b)). The limitation to top-level webpages and not the full Web search is due to computation and storage requirements of the Internet Archive.

In our work, we will use the URI-based query method when checking if the candidate webpage is archived.

2.1.4 COLD SPOTS

In our work, we want to find and recommend webpages in the archives that are important but are not surfaced in an archival search result, usually this is because links to them on the live Web have been removed. AlNoamany et al. [32] found that 63.6% of users' successful requests to the archive are made when the request is not found on the live Web. In addition, they found that 75.6% of users' unsuccessful requests to the archive are made when the request is not found on the live Web. If the webpage that links to the missing resource is removed, the user would not know the existence of the archived webpage.

An example is when a user requests a Virginia Tech football webpage from the archive. The user knows about the popular Virginia Tech football webpage `http://hokiesports.com/football/` and will request that webpage. This webpage is currently on the live Web and archived. However, the user does not know that the webpage `http://techsideline.com` exists in the archive, shown in Figure 16. In 2013, when requesting the webpage from the live Web it redirects to `https://virginiatech.sportswar.com`. If the user did not have a link to that webpage on the live Web, the user will never know it existed. We consider the

(a) UK Web Archive interface
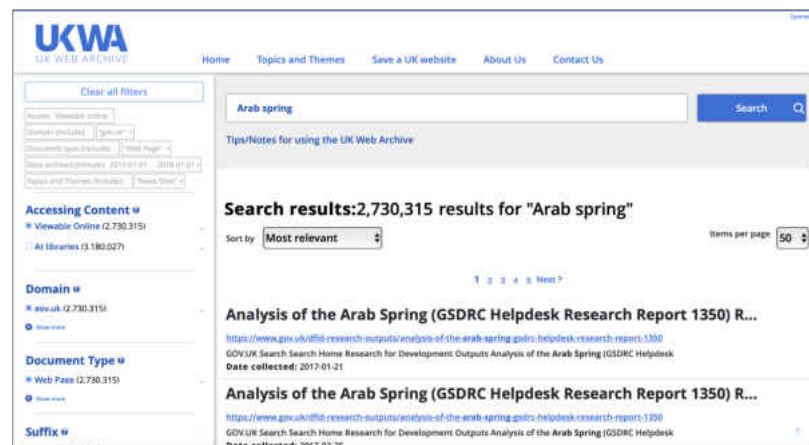
(b) UK Web Archive result of full-text search for the term "Arab Spring", and setting some filtering options such as restrict the date to between January 1, 2015 and January 1, 2018, the topics and themes to "News Sites", and select the domain "gov.uk".

Fig. 14 UK Web Archive full-text search based interface
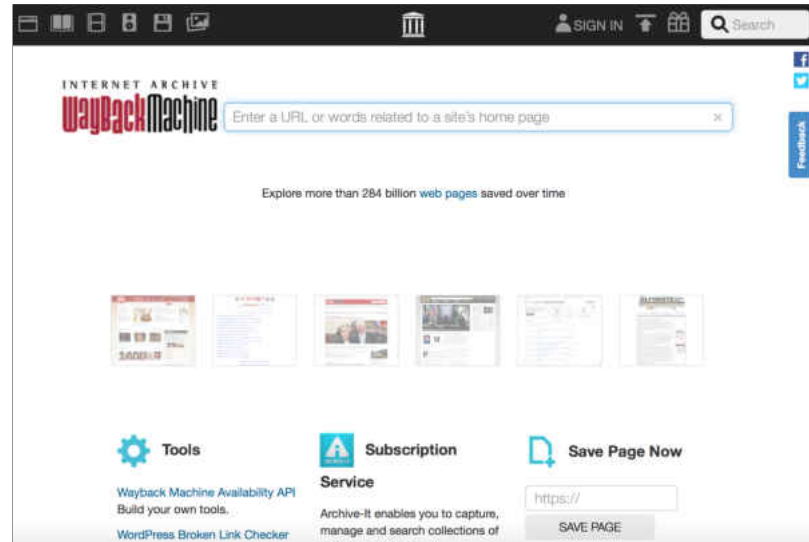
(a) Wayback Beta words related to a site's home page search based interface
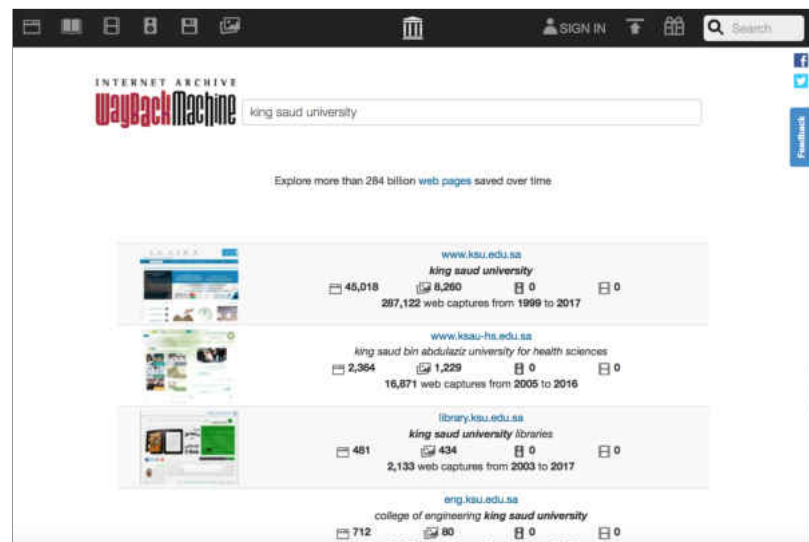
(b) Wayback Beta result after entering the query 'King Saud University'

Fig. 15 Wayback Beta full-text search interface

webpage `http://techsideline.com` to be a cold spot webpage.
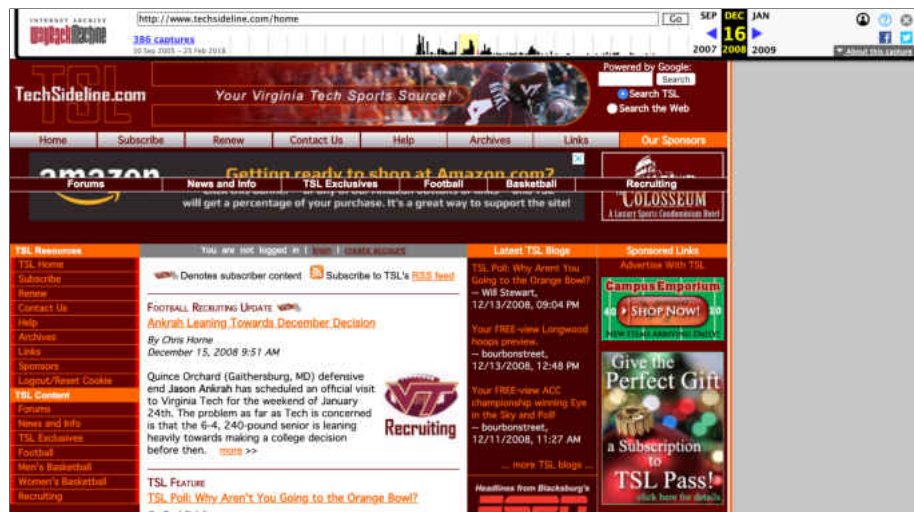
Fig. 16 A Virginia Tech football webpage found on the archive,
`https://web.archive.org/web/20081216011510/http://`
`www.techsideline.com:80/home`

Cold spot webpages are pages that are not live, are currently not popular, but are archived. We show how cold spots occur in Figure 17 where, at time t_1 URI1 links to both URI2 and URI3, URI2 links to URI4, and URI3 links to URI5. Also, all URIs are archived and are on the live Web. Then at time t_2 both URI2 and URI4 are removed from the live Web. However, URI1 still links to URI2. At time t_3 URI1 updates its links and removes the link from the webpages that are no longer live. This means that now it is hard to find both URI2 and URI4, and they are considered to now be in a cold spot of the archive.

An example of this case is assuming at time t_1, a DMOZ Directory (URI1) in 2005 lists different webpages and links to `www.techsideline.com` (URI2), which links to `www.techsideline.com/news` (URI4). At time t_2, URI2 and URI4 disappear from the live Web, although URI2 is still linked to URI1. At time t_3 in 2013, the DMOZ Directory (URI1) updated its links and eventually removed the offline links. As we see in later versions of the listings, URI2 `www.techsideline.com` was removed, so we would consider URI2 and URI4 to be cold spots in the archive.
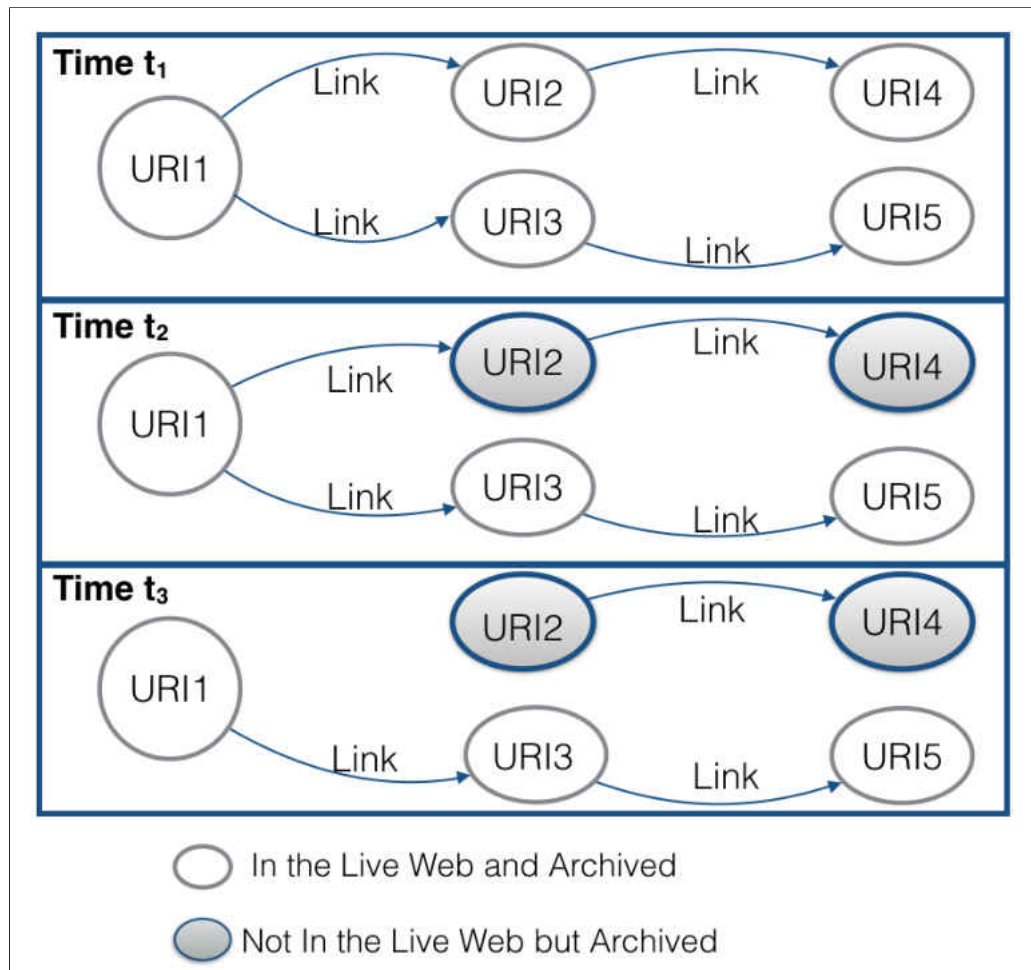
Fig. 17 How cold spots occur over time

2.2 URI AND THE HTTP STATUS CODE

To answer RQ1, "Given a URI, how much information can we learn about the referenced webpage without examining the content of the page?", we first need to discuss the structure, define cool URIs, and HTTP status codes.

2.2.1 URI STRUCTURE

A Uniform Resource Identifier (URI) is a string of characters, such as numbers, letters, and symbols, used to identify a resource [4, 9] (Figure 18). By entering a URI into the browser, the browser retrieves the resource representation and displays it to the user. For example, in Figure 18, the browser sends an HTTP GET request to the server at `weather.example.com` via TCP/IP port 80, and the server sends back a message containing a representation of the resource at the time of dereferencing.
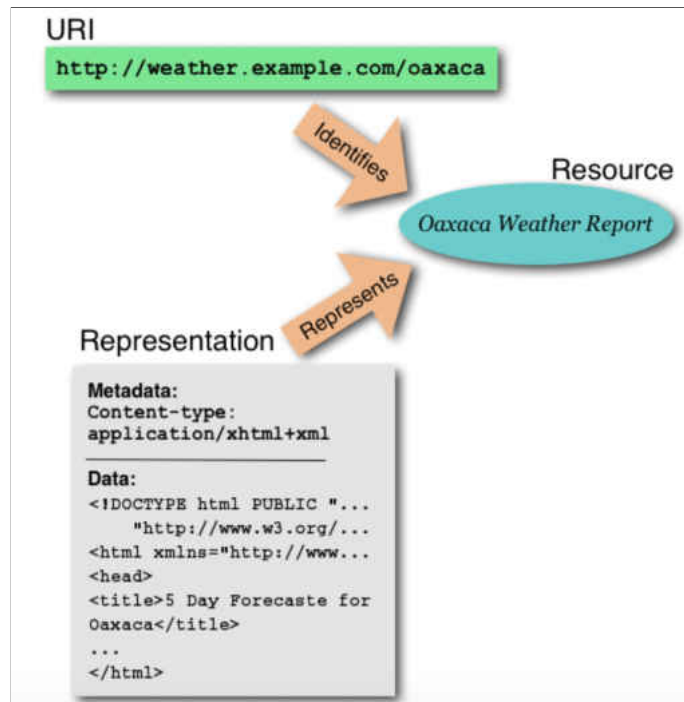
Fig. 18 The relationship between identifier, resource, and representation: URI identify resources and when dereferenced, the resources representation is returned [4]

A valid URI can identify a unique HTML page, a CSS document, an image, PDF document, or another format. In some cases, the URI can identify a resource that no longer exists or that has moved. A URI has two common components, the scheme and the hostname. For the URI `http://example.com/`, the scheme is `http`, and the hostname is `example.com`. The hostname includes the top level domain (TLD) or an effective TLD, which is located at the last part of the domain, such as `com` in the hostname `example.com` and `ac.uk` in the hostname `ox.ac.uk`. Other parts of the URI that may exist in the URI are the path, such as `page1/page2` from the URI `http://example.com/page1/page2`, the fragment such as `bar` from the URI `http://example.org/foo.html#bar`, and the query such as `key1=value1&key2=value2` from the URI `http://example.com/word?key1=value1&key2=value2`.

In general, a URI is an identifier consisting of a sequence of characters with the following syntax `scheme:hostname/path?query#fragment`. A URI can be a locator, a name, or both. A Uniform Resource Locator (URL) is a subset of a URI [33].

2.2.2 SORT-FRIENDLY URI REORDERING TRANSFORM

For sorting a list of URIs and detecting duplicates Sort-friendly URI Reordering

Transform (SURT)[8] is used. SURT is the process of transforming the URI to a natural hierarchy of domain names. It also includes changing the "https" scheme to "http" and making all characters lowercase. For example, the URI `<scheme://do main.tld/path?query>` is transformed using SURT to `<scheme://(tld,domain,)/ path?query>`.

2.2.3 SEMANTICS IN URIS

In 1998, Berners-Lee [34] introduced the term "Cool URIs" to describe URIs that could last for a long time. A URI should be designed to be short and easy to remember, such as `https://www.odu.edu/compsci`. Also, the URI should be designed so it will remain constant as long as possible by not including terminology that may change later, such as `.php`. An example of a cool URI is `https://en.w ikipedia.org/wiki/List_of_Presidents_of_the_United_States`. In addition, it may be helpful to add the creation date of the article or news page. The date the URI is issued is one thing which will not change. An example of this is `http:// www.cnn.com/2009/US/10/06/helen.keller.statue/`. In some cases, it is helpful to not to add some words that are considered to be stop-words, such as "latest" or "today" in `http://www.pathfinder.com/money/moneydaily/latest/`, because this does not help identify when it was published. Any information that might change over time should not be added to the URI, such as the author's name, status such as "old" or "new", file name extensions such as `cgi` or `html`, or software mechanisms, such as using `pl` when using Perl.

A URI may be composed of words that could indicate what the webpage is about. This improves its usability by immediately indicating what the webpage is about without containing details of the underlying Web application, such as `homepage.txt` or `example.asp`. This property is known as "Semantic URIs", as well as "Clean URLs" [35] or "User-friendly URLs" [36]. For example, with `https://en.wikip edia.org/wiki/United_States_presidential_election_2016` you can know that the webpage is about the 2016 US Presidential election. However, with `http:// www.firstmonday.org/article/view/7090/5653` you would not understand what the webpage contains without dereferencing it.

There are different types of semantics that might be encountered. We might have semantic information in the whole URI including both host and path (e.g.,

---

[8]`https://github.com/internetarchive/surt,https://pypi.org/project/surt/`

`https://www.bankofamerica.com/smallbusiness/business-financing/`), in the
hostname only (e.g., `http://americaslibrary.gov/jb/index.php`) or in the path
only (e.g., `https://en.wikipedia.org/wiki/Old_Dominion_University`).

Also, some hostnames, such as `https://www.thebestschools.org`, might have
more semantic information than others, such as `http://www.dhk.com`, by containing
text in the URI that give hints of its content.

On the other hand, some URIs may contain only stop-words, such as the URI
`http://www.them.com`, or contain tokens that have no semantics and do not match
an existing ontology, such as the URI `https://www.pltw.org`, where "pltw" has no
semantics for those who are unfamiliar with the acronym and may not be present in
certain ontologies.

2.2.4 HTTP STATUS CODES

We can get different responses when requesting a webpage. An HTTP response
status code [28] is part of the response a client receives from the server when making
a request. It consists of three digits that specify a certain response status. The
common responses are:

- `2xx` - Successful responses. This class indicates that the client's request was
  successfully accepted.

- `3xx` - Redirection messages. This class indicates that the client's request was
  redirected to another resource.

- `4xx` - Client error responses. This class indicates that the client is requesting
  something the server cannot or will not provide. When a response to a webpage
  request is a `200 Ok` (Section 2.2.4), but the resulting webpage is a customized
  error page, this is called a "soft 404".

2.3 CLASSIFICATION

To answer RQ2, "Given a set of terms that describes a webpage, how do we map
the terms to an ontology of webpages?", we need to discuss classification and the
ontology that we will use to classify the requested URI. We first discuss the different
webpage classification types and structure.

Webpage classification is the process of assigning a webpage to a predefined category. There are different types of webpage classification [37], such as subject classification, sentiment classification, genre classification, functional classification, and others:

- Subject classification: the process of assigning a topic to a webpage, such as sports or art.

- Sentiment classification: the process of determining the webpage author's opinion about a topic, such as a positive or a negative opinion.

- Genre classification: the process of determining the means of a webpage's form, style, or targeted audience, such as blog, commercial, or error.

- Functional classification: the process of determining the roles that the webpage plays, such as personal webpage or admissions webpage.

In this work, we will focus on *subject classification*, since we want to determine what the requested webpage is about in order to recommend other webpages in the same category.

In general, there are different types of classifications, such as binary classification or multi-class classification. Binary classification is to categorize instances to one of two categories, for example, to categorize a webpage to art or non-art. Multi-class classification is to categorize instances to one of many different categories, for example, to categorize a webpage to sports, art, news, or any other category.

The structure of the categories could be flat classification, where categories are parallel, as in Figure 19(a), or hierarchical, where categories are in a tree-like structure, as in Figure 19(b).

2.3.1 MACHINE LEARNING ALGORITHMS FOR CLASSIFICATION

In our work, we will use a machine learning (ML) algorithm to classify a requested webpage in order to find the most similar candidates. When choosing a ML algorithm, we will have to consider working on categorized ontologies that are large in size and contain a considerable number of categories and sub-categories. In addition, we will have to consider the structure of the data, which is an hierarchical classification. Also, when the user requests the webpage we need to provide recommendations with

(a) Flat Classification

(b) Hierarchical Classification
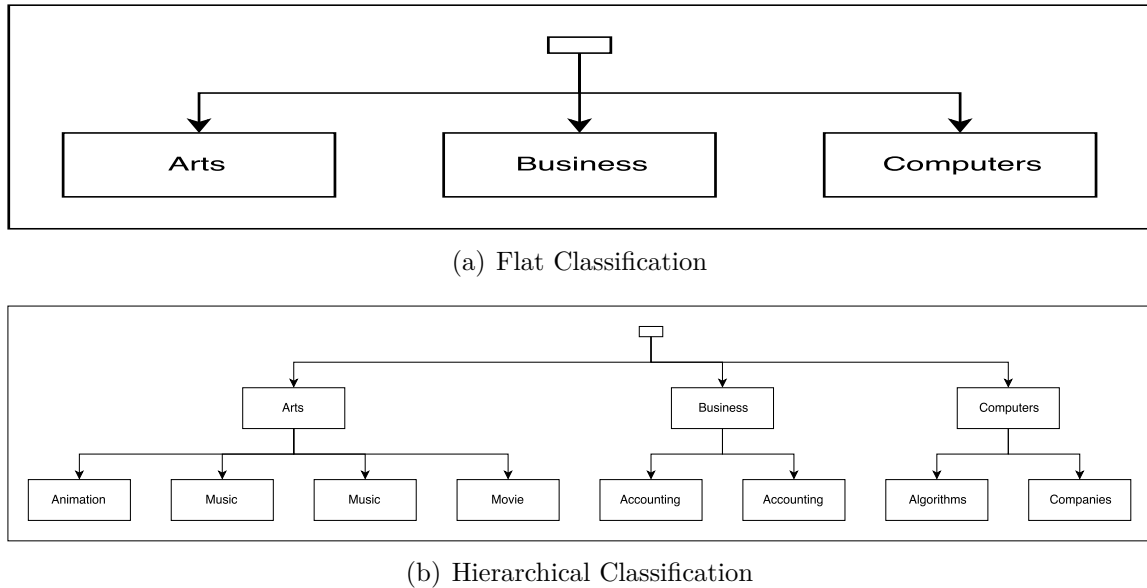
Fig. 19 Classification Structures

minimal time and storage used. Finally, we will consider algorithms that have been used in related work, which we will use to compare with our results.

Based on the ontology type and size we can use different type of algorithms. If we are working with a large dataset that contains independent features and we prefer simplicity, then for example, we can use the Naïve Bayes (NB) algorithm.

Depending on the ontology structure, different type of algorithms have been used. For flat classification, many machine learning algorithms have been used such as NB, Support Vector Machines (SVM), and other algorithms. For hierarchical structure, different approaches have been used, such as the big-bang approach, the top-down approach [38], and the deep classification algorithm [6]. These methods will be discussed in Section 3.2.2.

Both classification and clustering are techniques used in machine learning. In a categorized dataset, classification is the process of predicting which category an object belongs to, based on common attributes. In a non-categorized dataset, clustering is the process of placing the dataset into groups, based on determining if there is a common attribute. Both clustering and classification are two efficient methods of grouping a massive volume of data. In general, there are two common different learning styles in machine learning algorithms [39]:

- Supervised Learning: All instances are labeled, such as the classification of a

dataset. In this method, there is initially a categorized dataset and a function that maps a new input to one of the classifications.

- Unsupervised Learning: Instances are not labeled, such as the clustering of a dataset. In this method, the function will look for similarities to separate the instance into groups.

The most common machine learning algorithms are Naïve Bayes, Support Vector Machines, and Maximum Entropy. These algorithms were used in related work [1, 6] that we will use to evaluate our results.

Naïve Bayes (NB)

Naïve Bayes is considered the most straightforward machine learning algorithm [40]. It assumes that if a category is created given the existence of several features, the features themselves are independent from each other, this is why it is called "Naive". NB is commonly used when speed is required and when the dataset is relatively large. The Naïve Bayes equation is as follows:

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x} \mid C_k)}{p(\mathbf{x})} \tag{1}$$

where

- $p(C_k \mid \mathbf{x})$ is the posterior, which is the probability that the attribute $\mathbf{x}$ is in class C_k

- $p(C_k)$ is the prior, which is the probability of the class C_k

- $p(\mathbf{x} \mid C_k)$ is the likelihood, which is the probability of the attribute $\mathbf{x}$ given the class C_k

- $p(\mathbf{x})$ is the evidence, which is the probability of the attribute $\mathbf{x}$

The same equation can be written as

$$\text{Posterior} = \frac{\text{Prior} \times \text{Likelihood}}{\text{Evidence}} \tag{2}$$

Support Vector Machines (SVM)

SVM is binary classification algorithm [41, 42]. Given labeled training data, the algorithm outputs the dataset on a space and divides the dataset into two clusters with the maximum margins between them, then tries to classify an input to one of the two categories. SVM is typically used when accuracy is the priority when categorizing a dataset.

Maximum Entropy (ME)

Maximum Entropy (ME) is a probabilistic classifier that performs an estimation method based on a uniform distribution by making as few assumptions as possible [43, 44]. ME requires more time to train compared to NB.

2.4 ONTOLOGIES

An ontology is a set of categories that contain a group of related instances. There are different types of ontologies [45], such as:

- Terminological ontologies: a set of words senses that contain a group of related words, such as WordNet [46].

- Topic ontologies: a set of topics that contain a group of documents, such as DMOZ.

- Data-model ontologies: a set of tables that contain a group of data records, such as a database schema.

In our work, we need to map terms extracted from a URI to a *topic ontology.* In related work, different ontologies have been used to classify webpages. Some ontologies that are commonly used for classification are DMOZ and Wikipedia. In this section, we describe DMOZ and Wikipedia.

2.4.1 DMOZ

DMOZ is also known as the Open Directory Project (ODP), which is the largest human-edited directory of the Web. It is considered a hierarchical classification in which each category may have sub-categories. DMOZ was based on user contributed submissions and maintained by a community of volunteer editors [47].

DMOZ is a very popular dataset that has been used in previous research. In 2011, Ainsworth et al. [48] and AlSum et al. [49] used DMOZ as one dataset and measured how much of the Web is archived. In our previous work [50, 51] we used DMOZ as one of the dataset resources and measured how much Arabic, English, Danish, and Korean webpages are archived. Using DMOZ for classification has a long history [52, 53]. It is one of the oldest resources available, which makes it a good source for URIs that may no longer exist but have been crawled and archived. DMOZ was also used as an ontology for research related to classifying webpages [54, 1]. Each category has a list of URIs, the website titles, and descriptions.

DMOZ was closed down on March 14, 2017 (Figure 20) and currently, the webpage `dmoz.org`, links to a mirror webpage (Figure 21). We downloaded the dataset locally in Feb 2017. In our local dataset, there are 3,573,189 categorized URIs, with 2,539,029 of them being unique. The DMOZ listing has been archived in the Internet Archive (Figure 22). However, some inner links may be missing or redirect to pages from another archived date.

The dataset consists of four fields, the category, the URI, the webpage title, and the description, (Table 8). In general, there are 16 main categories and 590,516 unique sub-categories. The main categories are "Arts", "Business", "Computers", "Games", "Health", "Home", "Kids and Teens", "News", "Recreation", "Reference", "Regional", "Science", "Shopping", "Society", "Sports", and "World". The "World" category contains foreign-language webpages. These categories may differ slightly because the directory is updated over time.

Table 8 An example of a URI and its category

Category	Computers/Computer science/Academic departments/ North America/United States/Virginia
URI	`http://www.odu.edu`
Title	Old Dominion University
Description	Norfolk, Virginia
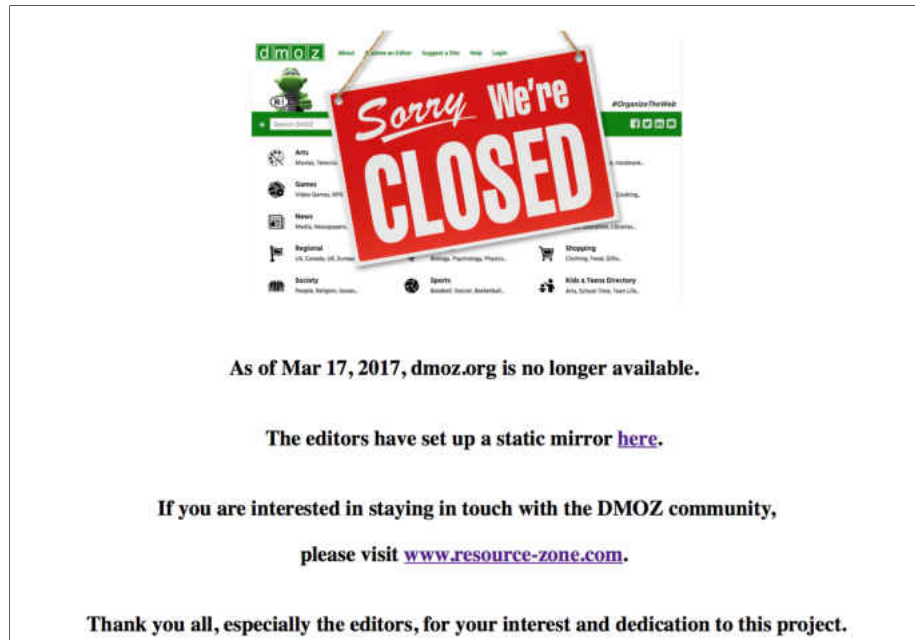
Fig. 20 DMOZ is no longer available, Source: `http://www.dmoz.org`
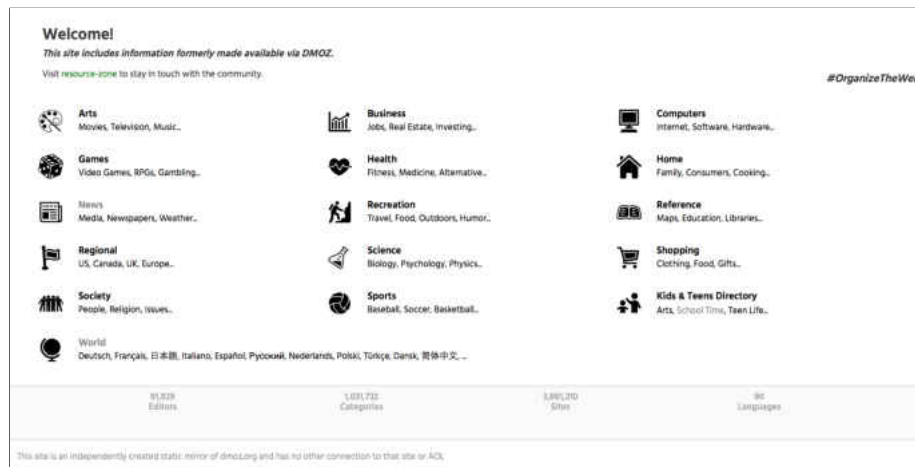
Fig. 21 DMOZ mirror website, Source: `http://dmoztools.net`

Fig. 22 DMOZ webpage from the Internet Archive, Source:
`https://web.archive.org/web/20080105042131/http://www.dmoz.org:80/`

2.4.2 WIKIPEDIA

Wikipedia is a web-based encyclopedia, launched in 2001 and available in 301 languages. Wikipedia has been used as a corpus, for classifying and clustering, for creating annotations, etc. [55]. In 2010, Glott et al. [56] estimated that 68.25% of Wikipedia responders are readers, compared to 31.75% contributors.

In using Wikipedia to classify entities Gattani et al. [57] used texts from Twitter as a dataset, then extracted its keywords and matched it to Wikipedias links to classify and tag the tweet.

Wikipedia contains 38 main categories and 898 second level sub-categories. Each sub-category contains other sub-categories. The main categories are: "Academic disciplines", "Arts", "Business", "Concepts", "Crime", "Culture", "Economy", "Education", "Entertainment", "Events", "Food and drink", "Geography", "Government", "Health", "History", "Human behavior", "Humanities", "Knowledge", "Language", "Law", "Life", "Mathematics", "Military", "Mind", "Music", "Nature", "Objects", "Organizations", "People", "Philosophy", "Politics", "Religion", "Science", "Society", "Sports", "Technology", "Universe", and "World". These categories may differ slightly because it is updated over time.

Articles in Wikipedia contain a summary about the topic, external links, references, categorization of the article, and an information box. References are links that appeared within the article, however external links are links to webpages outside

Wikipedia, and usually do not appear in the article.

For instance, the Wikipedia webpage on Old Dominion University shown in Figure 23 contains a summary about the topic, external links, references, categorization of the article, and an information box. Figure 23(b) shows the end of the Wikipedia webpage where a list of 65 references are mentioned within the article. Next, we see two external links: "Official website" that links to the Old Dominion University webpage `https://odu.edu`, and the external link "Old Dominion Athletics website" which links to `http://odusports.com`. Finally, we see the categorization section where the article is characterized to multiple categories (Table 9).

Table 9 An example of an external link and its category

Official website as an external link	`https://odu.edu`
Categories	Old Dominion University Universities and colleges in Virginia Educational institutions established in 1930 and Public universities ...

2.5 RECOMMENDATIONS

Given a set of candidate webpages we need to recommend several alternatives to rank the candidates and select the top *N*. In this section we discuss recommendation systems and their methods.

Recently, recommendation lists have become popular in different areas such as books, movies, articles, and in any search query in general. Many major companies' websites provide recommendations for users including Google, Facebook, Twitter, LinkedIn, Netflix, Amazon, Microsoft, Yahoo!, and many others. This allows the user to look at the results of other similar search queries.

Recommender systems produce a list of recommendations in one of the following ways: stereotyping, content-based filtering, collaborative filtering, co-occurrence, and hybrid. Different recommendation systems were discussed and compared in several surveys [58, 59, 60].

The type of recommendation system is based on the information available. Often,

(a) Wikipedia webpage `https://en.wikipedia.org/wiki/Old_Dominion_Uni versity` resulted after searching for `odu.edu`

(b) The end of the Wikipedia webpage `https://en.wikipedia.org/wiki/Ol d_Dominion_University`, showing part of the references, external links including `http://odu.edu` as an "official website", and the categories that this Wikipedia webpage belongs to

Fig. 23 Searching for the request `http://odu.edu` in Wikipedia resulted in finding the Wikipedia webpage
`https://en.wikipedia.org/wiki/Old_Dominion_University`

the available resources are user's demographics (e.g., age, gender, location), the user-item preference (e.g., ratings, purchases), and item features (e.g., keywords).

2.5.1 STEREOTYPING

In 1979, Rich [61] introduced the recommender system named Grundy, which recommended novels to its users. They defined stereotypes as collections of characteristics. For example they assume that men are interested in suspense novels and not in romance novels. Therefore, they would recommend this type of novels to all men. However, it is known that this is not true for all men and this recommendation method will limit the number of books recommended to the user.

Other researchers [62] used this method for their research paper recommender system, Docear[9]. Docear is a literature management software that allows managing PDF files, annotations, and references in mind maps. In this work, the author assumed that all of their software users are researchers or students and would recommend research papers based on their interests.

2.5.2 CONTENT-BASED FILTERING

Pazzani et al. [63] discussed content-based filtering. Content-based filtering approaches use a range of features of an item to recommend items with similar properties. Items are usually textual such as webpages, news articles, television programs, songs, buying items, or tagging an item. This method allows the creating of a profile for each user that describes the types of items the user likes. Then, the method compares the items to determine what to recommend.

Pandora Radio[10] is an example of a content-based recommender system. It uses the user's first choice of song as a seed, and it then recommends similar playlists. The advantage of using this method is that it allows user-based personalization. However, all of the playlists have to be categorized in advance in order to recommend similar playlists.

2.5.3 COLLABORATIVE FILTERING

The collaborative filtering approach is to build a model from a user's past behavior. It requires a large amount of data from users to make accurate recommendations.

---

[9]http://docear.org
[10]https://pandora.com/

In this case, the recommendations are not based on item similarity but user similarity.

In 1992, Goldberg et al. [64] introduced collaborative filtering. The system they introduced was called Tapestry, which is an experimental mail system. The motivation for Tapestry comes from the large volume of incoming e-mail. The system requires people to record their reactions to documents they read. Then system would recommend documents to users. This method is considered content-dependent.

YouTube uses collaborative filtering. Covington et al. [65] mention that the system consists of two neural networks, candidate generation and ranking. Candidate generation takes as input a user's watch history and selects videos in the range of hundreds. For ranking, they used a deep neural network with similar architecture as candidate generation to assign an independent score to each video impression using logistic regression. Davidson et al. [66] also described YouTube's recommendation system and some of the challenges that they addressed. Such challenges are poor metadata, size, and the short life cycle going from upload to viral.

Collaborative filtering approaches often suffer from three main disadvantages: cold start, scalability, and sparsity. Cold start is when the user first starts using the system and the system does not have previous information about the user, so the recommendations are not accurate. In order to make accurate recommendations it requires a large amount of data from the users [59]. Scalability is a disadvantage because a large number of candidates to choose from will require in a large amount of computational power. Sparsity is when users rate a few number of items, so the system can learn a little information about the items.

2.5.4 CO-OCCURRENCE

Co-occurrence recommendations are items that are recommended that frequently co-occur with some source items. This method was proposed in 1973 by Small [67]. Amazon uses this method among other methods. When a customer browses a product on Amazon, items frequently bought with that item are recommended: "customers who bought item X also bought item Y".

Pinterest also uses this type of recommendation system among other methods. Liu et al. [68] showed that over 40% of user engagement on Pinterest are from related Pins. They discussed the evolution of the system and the challenges that occurred while building it. The major pieces of related Pins were candidate generation, memorization, and learning to rank.

2.5.5 HYBRID

Hybrid recommender systems are the result of combined approaches to improve performance [69]. Netflix, Amazon, and Pinterest are all examples of the use of hybrid recommender systems. Netflix makes recommendations using collaborative filtering by comparing the watching and searching methods of similar users, and content-based filtering by offering movies that share characteristics with other movies that a user has rated highly.

2.5.6 EXAMPLE CASES OF USING RECOMMENDATIONS

When searching for a book on Amazon, Amazon provides the search result as well as other similar products. For example, when searching for the book *Beautiful Data*, Amazon shows the book as the top result (Figure 24(a)) as well as my other recently viewed items and featured recommendations that are similar (Figure 24(b)). When I select one of the books, more recommendation selections appear, such as what was frequently bought together (Figure 24(c)), what other customers who bought this item also bought (Figure 24(d)), customers who viewed this item also viewed (Figure 24(e)), and my recent viewed items and feature recommendations which also appear on the similar books page (Figure 24(f)).

We notice that Amazon makes recommendations based on similar items, the user's search history, featured recommendations, the history of other users, and what others viewed or bought. This means that Amazon has to preserve each item's history and user history.

Other popular services, such as Netflix, also make recommendations based on the user's history. When a user browses Netflix, the user will find a recommendation list based on what is currently popular, what is trending now, and what was previously watched (Figure 24), as well as others such as watch again, top picks, added recently, different categories such as kids TV, children and family movies, and many more. Netflix uses a combination of the user's history and popular items.

In general, recommendation systems can be personalized by content-based filtering (results are based on past ratings and purchases), collaborative-based filtering (users are categorized by their demographic profiles or similar rating preferences), knowledge-based filtering (knowledge about the price, quality), and non-personalized filtering (results are identical to all users, based on popularity in terms of both ratings

(a) Amazon found requested book and recommend similar requests

(b) Amazon recommendation based on my recent viewed items and featured recommendations

(c) Amazon recommendation based on what was frequently bought together

(d) Amazon recommendation based on what other customers who bough this item also bought

(e) Amazon recommendation based on what other customers viewed who viewed this item

(f) Amazon recommendation based on my recent viewed items and featured recommendations

Fig. 23 Amazons recommendation examples

and sales).
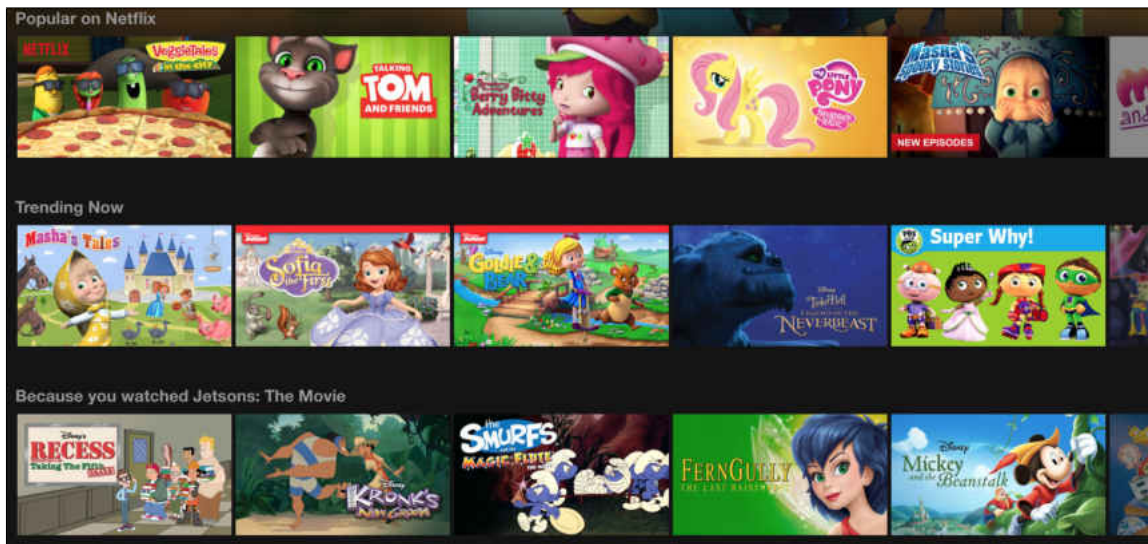
Fig. 24 Netflix recommendation based on previous watched movies

In our work, we will be building a recommendation system using *content-based filtering*, which takes information from a URI and known ontologies of webpages and recommends archived webpages to the user, while focusing on some features such as hard to find webpages. In this work, we do not have access to user preference or past queries of the archive.

2.6 SUMMARY

In this chapter, we introduced some basic concepts related to our work. First, we presented Web archiving and explained why it is important. We also described the Memento framework, the Internet Archive interfaces, and defined cold spots. This relates to RQ3. After that, we discussed the basic terms related to RQ1, which focuses on obtaining information from the URI only. We discuss URIs, their structure, status code, and defined cool URIs and empty URIs. Next, we introduced concepts relating to RQ2, which is mapping terms to an ontology. We do that by introducing classification and some common machine learning algorithms. We also gave an overview of ontologies and described DMOZ and Wikipedia. Finally, we defined terms relating to RQ4, which focuses on recommending candidate webpage, where we described recommendations and listed its popular usage. In the next chapter, we will show previous work that is related to our work.

CHAPTER 3

RELATED WORK

In this chapter, we provide an overview of the research that has been established in several fields related to the problems we investigate in this work. We present work related to splitting compound words (Section 3.1), clustering (Section 3.2.1), and classifying (Section 3.2.2). After that, we present work related to our methods to filter candidate webpages, which includes existence in the archive (Section 3.3), aboutness (Section 3.3.7), and soft 404s (Section 3.4). Finally, we present related work to our method to rank the candidates, which includes memento damage (Section 3.5.1), retrievability (Section 3.5.2), and detecting the creation time of webpages (Section 3.5.3).

3.1 SPLITTING COMPOUND WORDS

To address both RQ1, "Given a URI, how much information can we learn about the referenced webpage without examining the content of the page?", and RQ2, "Given a set of terms that describes a webpage, how do we map the terms to an ontology of webpages?", we will evaluate different methods to split URIs into tokens and classify them and then to find other recommendations. In this section, we present related work that uses the URI to gain information about the webpage it holds. Using the information that the URI holds includes splitting compound words, such as splitting the URI `http://www.norfolkbotanicalgarden.org` into the tokens {norfolk, botanical, garden}.

Chi et al. [70] proposed methods to segment compound words and detect partial words. This work introduced some basic terms that we can use in our work such as:

- Document framework of a webpage: a set of information that contains the webpage title, the URL, hyperlink addresses, embedded object file names, and their alternate texts.

- Word: a consecutive string of characters separated by delimiters.

- Dictionary word: a word that exists in a given dictionary.

- Word set: all the words extracted from a document framework.

Two steps were performed to solve the problem of segmenting compound words, using a combined forward and backward matching to split compound words. First, they try to identify the components of the compound words. Second, they match each part to a dictionary, including handling the cases of acronyms, abbreviations, and meaningless words. To identify the prefix in a compound word, they search for matching words starting from the first letter, which is called forward matching. For identifying the suffix in a compound word, they searched for matching words starting from the last letter, which is called backward matching. In both cases, the priority is to search for the longest matching word. To solve the second issue of trying to match each part to a dictionary, they created and maintained a table of acronyms and abbreviations. Their results showed an accuracy of 90%.

Koehn et al. [71] considered splitting German compound words based on words that exist in a training corpus. The corpus used was the European Parliament Proceedings, consisting of 20 million words in German. They tested different methods for splitting words such as how frequently the word appears in training set, translating German compound words into English, then translating it back to German. They restricted their work to words of three or more letters and excluded parts of words based on their part-of-speech. Also, if more than one split option occurred, then they would choose the highest score based on frequency. Their results showed an accuracy of 99.1%.

Segaran et al. [72] described a word segmentation solution for compound words, including these that may have multiple options such as `chosespain.com` where both {choses, pain} and {chose, spain} are valid splits. They use some features to split words such as using N-grams and term frequency.

Khaitan et al. [73] described an application for splitting domain names called Domain Keyword Extraction, which uses compound splitting that allows selecting relevant keywords from misspelled or non-existing domains. First, they performed domain sanitization by removing prefixes and suffixes from domain type errors, such as eliminating `.com` and `www.` from `www.bankofamerica.cmo` resulting in `bankofamerica` and eliminating `.cm` and `www.` from `www.sprintstore.cm` resulting in `sprintstore`. Note that `.com` was misspelled in this case and was detected and removed. Next, they split words based on features added, such as length of words, N-grams, stop-words, and frequency. They used the Wikipedia corpus for training using only

the titles of the pages. After scoring each component of the compound words, they filtered any part that had a score less than 0.2. Using the hybrid method, which uses all the features together, got the highest F_1 score.

Baykan et al. [54, 1] investigated using the URI to classify a webpage and identify its topic. Using different dictionaries including DMOZ, they split the URI using four methods:

- **Tokens**: The URI is converted to lowercase, then split by any non-alphabetic characters. Finally, they removed any tokens less than length 2. An example is the URI `http://www.allwatchers.com/Topics/Info_3922.asp`, tokenized to {allwatchers, com, topics, info, asp}.

- **N-grams as features**: This approach splits the URI into the same tokens as the method above. Then it uses the letter N-grams, including 4-, 5-, 6-, 7- and 8-grams. An example of using a 5-gram as features is the URI `http://www.allwatchers.com/Topics/Info_3922.asp` tokenized to {allwa, llwat, lwatc, watch, atche, tcher, chers, com, topic, opics, info, asp}.

- **N-grams from the URI**: The word "HTTP", punctuation, and numbers are removed, and the case is changed to lowercase. Then the letter N-grams are used, including 4-, 5-, 6-, 7- and 8-grams. An example of using a 5-gram from the URI is `http://www.allwatchers.com/Topics/Info_3922.asp` being transformed to {allwatcherscomtopicsinfoasp}, then tokenized to {allwa, llwat, lwatc, watch, atche, tcher, chers, hersc, ersco, rscom, scomt, comto, omtop, mtopi, topic, opics, picsi, icsin, csinf, sinfo, infoa, nfoas, foasp}

- **Encoding positional information**: In this approach, each token is duplicated, and its position is added. An example of using a 3-gram is the URI `www.epfl.ch` tokenized as {www, www_1, epf, epf_2, pfl, pf_2, ch_, ch_3}. Using this approach, the machine learning algorithm will not see the number, but will not treat the tokens that appeared in different positions the same, as they will have different dimensions for each.

To evaluate the different tokenization methods, Baykan et al. used the DMOZ dataset (ODP) with Naïve Bayes to classify the tokenized URIs. They found that the all-gram, combination of 4-, 5-, 6-, 7-, and 8- grams, was the best approach. The result of evaluating the different tokenizations used is shown in Table 10. They also

found that the token-based method may not work well, due to the high percentage of empty URIs, which are URIs that include only stop-words or have never been seen in the training set, such as `http://www.yuzutree.com` or `www.mlx.co.uk`. In this case, breaking the URI to small N-grams helps with some empty URIs.

Table 10 Macroaveraged F_1 Values of Various Features with Naïve Bayes [1] (Table II)

Feature	F_1
Tokens	76
3-grams	75
4-grams	80
6-grams	81
8-grams	79
4-5grams	81
4-5-6grams	81
4-5-6-7grams	81
4-5-6-7-8grams	82

In later work, Baykan et al. [74, 2] proposed a method to identify the language of the website using the URI only. The language detection model is useful for our work if we want to expand to include different languages. They used some features such as words, various sized N-grams, and custom-made features. The classifiers were built for English, German, French, Spanish, and Italian by applying machine learning algorithms and features. The features used are words-as-features, various sized N-grams, and custom-made features. Words-as-features is breaking the URI into tokens then having a counter associated with each token to mark how many times it has appeared as a token in this specific language. N-grams as features is creating N-grams of the URI to tokens then having a counter in the same manner as the previous method. Custom-made features are the combination of the use of a dictionary, top-level domains, and IP addresses of the hosting Web servers. They compared their approaches with two methods, classification by country code top-level domains and classification by IP addresses of the hosting Web servers. Two datasets are used, DMOZ containing 150k URIs for each of the five languages (English, French, German, Spanish, and Italian), and by querying a commercial Search Engine Results (SER) obtained from Microsoft's Live Search[1] containing 100k URIs for each language. In their earlier work [74], they also had a third dataset which includes a random

---

[1] `http://search.live.com`

sample of 1,260 pages from a web crawl. A classifier was trained with a 10-fold cross-validation on the dataset. In this work, they found that the best performing classifier is for German with a score of 97.7 using Decision Trees (DT) with custom-made method, shown in Table 11. On the other hand, the lowest score was for English with a score of 94.2 using Support Vector Machines (SVM) with all-grams.

Table 11 Comparison of Various Approaches for Language Identification on ODP + SER Test Set [2] (Table XIV)

Lang.	Classifier	P	R	p(-|-)	F1
En.	ccTLD	96.6	22.8	99.2	36.9
	IP	82.1	85.9	81.3	84.0
	SVM with allgrams	**92.2**	**96.2**	**91.9**	**94.2**
	DT with custom-made	92.3	95.9	92.0	94.1
	RE with all grams	90.8	93.1	90.6	91.9
	SVM with all grams + IP	80.2	98.2	75.7	88.3
Ge.	ccTLD	98.5	85.4	98.7	91.5
	IP	92.8	92.7	92.8	92.7
	SVM wit allgrams	98.5	95.8	98.5	97.2
	DT with custom-made	**97.3**	**98.1**	**97.3**	**97.7**
	RE with allgrams	97.0	92.8	97.2	94.9
	SVM with all grams + IP	92.7	99.0	92.2	95.7
Fr.	ccTLD	99.8	36.8	99.9	53.7
	IP	97.5	64.6	98.4	77.7
	SVM wit allgrams	95.8	93.0	96.0	94.4
	DT with custom-made	96.2	92.6	96.3	94.4
	RE with allgrams	94.2	91.7	94.4	92.9
	SVM with all grams + IP	**94.6**	**96.6**	**94.5**	**95.6**
Sp.	ccTLD	99.9	37.2	99.9	54.2
	IP	99.7	50.0	99.8	66.6
	SVM wit allgrams	95.8	94.2	95.9	95.0
	DT with custom-made	90.2	94.0	89.8	92.1
	RE with allgrams	94.7	91.8	94.8	93.2
	SVM with all grams + IP	**95.8**	**96.5**	**95.7**	**96.2**
It.	ccTLD	99.8	61.3	99.9	75.9
	IP	99.1	73.5	99.3	84.4
	SVM wit allgrams	97.7	94.5	97.8	96.1
	DT with custom-made	98.5	94.3	98.6	96.4
	RE with allgrams	95.5	93.7	95.6	94.6
	SVM with all grams + IP	**97.3**	**98.0**	**97.2**	**97.6**

Raju et al. [75] proposed algorithms called URL-KEX, which is extracting advertising keywords from the URL string alone, and Page-KEX, which is extracting

keywords from the webpage. URL-KEX operates in three stages: (a) removing keywords from the URL string, (b) keyword synthesis to generate more keywords, and (c) ranking by computing the relevance score for each keyword and ranking in decreasing order of relevance. In this work, keyword extraction is performed using the following steps. First, the URL is split into hostname, path, and query. Next, each delimiter that is a non-alphanumeric is split into individual units called segments, while removing stop-words such as "html". After that, compound words are split, keeping in consideration to not split valid words further. This is accomplished by using a dynamic program to split the word into valid terms while favoring the split with the highest number of occurrences. Note that a list of common words such as "homepage" is filtered out. In the keyword synthesis step, the keywords from different segments are combined. Finally, in the ranking step, the score of the keyword is based on the parent segment's weight and keyword length. Usually, the query segment is more informative than the path, and the path is more informative than the hostname. In this work, they found that using Page-KEX, at least one keyword is produced for 58% of the URLs and at least two keywords are produced for 37% of the URLs. On the other hand, due to the limitation of text in the URL, URL-KEX was able to produce one keyword for 44% of the URLs and two keywords for 26%.

Lin et al. [76] proposed a method for malicious URL filtering based only on the URL string. In this work, they used two feature sets to separate benign URLs from malicious ones. The first includes lexical features that describe the dynamic information of URLs using the "words". In this method to split the URI they used different delimiters based on URL components. For the domain name, they used dash ("-") and slash ("/") as delimiters. For the path, they used the dash ("-"), dot ("."), underscore ("_"), and backslash ("/"). Finally, for the argument, they used the ampersand ("&") and equal sign ("="). They replaced the IP address with an Autonomous System (AS) number and replaced the digits in words with a regular expression. They kept only the words generated in the last 24 hours by giving every word a timestamp as they occur, and others are considered expired. The second feature set includes descriptive features that describe some statistical characteristics of URLs. Here they split the path component into sub-directory, filename, and file extension to obtain more detailed information. They removed "www", "www" with any number (such as www1), country code top-level domain (ccTLD), and generic top-level domain (gTLD). Some features are added, including the following: length

of the URL, length ratio of different components, number of letter-digit-letter and digit-letter-digit occurrences, delimiter count, and longest word length. To train the data they used the Passive-Aggressive (PA) [77] algorithm and the Confidence Weighted (CW) algorithm [78]. Using this approach resulted in misses in less than 9% of malicious instances.

Berardi et al. [79] split compound words in Twitter hashtags. They assigned an indexing function for hashtags that automatically recognizes distinct words. The indexing and retrieval system is called CipCipPy[2]. The method they used for splitting the words is based on the Viterbi algorithm [80], which computes the most probable path based on observed events. In this work, they also proposed an expansion of tweets based on the title of any referred webpage. Also, they introduced a method to rank tweets based on their quality. In this work, they used an English corpus[3] as their dataset.

Cui et al. [81] proposed using hashtags as an indicator of events. They split the hashtags to extract keywords. The method used to split compound words in the hashtag was using a dictionary containing more than 80,000 common English words from 12Dicts [82] to obtain minimal words. Words not in the dictionary are split into individual letters.

Zhang et al. [83] proposed a method to automate Twitter topic summarization. This work includes splitting compound words in a hashtag. They identified two types of hashtags. The first are those with mixed case characters such as "#CyberMonday", and this case is easy to split. The second case are those with all lowercase characters like "#letsbehonest". In this case they used the Chinese text segmentation method for text retrieval proposed by Wu et al. [84]. They used both unigrams and bigrams from all tweets. Next, they scored every possible split option by scoring it with an N-gram. The accuracy of this method was 97%.

In our work, we need to extract all possible tokens in the URI not only focusing on the keywords. We are going to present a method of extracting tokens that is similar to some of the work mentioned above. Our method combines multiple methods such as filtering the URI and then splitting the tokens by matching the tokens to a dictionary and using N-grams. We will evaluate our method and compare the results to the tokenization methods proposed by Baykan et al. [54, 1].

---

[2]`http://tag.isti.cnr.it/cipcippy/`
[3]`http://norvig.com/big.txt`

3.2 CLUSTERING VS. CLASSIFICATION

In our work, after gathering information from the URI, we will need to classify the URI in a clustered list of URIs. This step is necessary to answer RQ2, "Given a set of terms that describes a webpage, how do we map the terms to an ontology of webpages?". Clustering and classification are two different techniques (Section 2.3.1).

3.2.1 CLUSTERING

Clustering is the process of grouping unlabeled objects based on a certain similarity. There has been much research done on clustering webpages. Arranging a significant amount of data in related groups is essential but a hard task; however, it helps us to classify them better.

Wen et al. [85] propose to cluster similar URI queries of Frequently Asked Queries (FAQ) according to their contents as well as user logs, to recommend URIs to FAQ of a search engine. They use four features of query distance: (a) based on keywords or phrases of the query, (b) based on string matching of keywords, (c) based on common clicked URIs, and (d) based on the distance of the clicked documents. They applied this work on the Encarta encyclopedia[4], which is organized in a hierarchy. Their results show that the clusters provide useful information for the FAQ.

Baeza-Yates [86] presented an algorithm for clustering queries to a search engine. Given a query submitted to a search engine, their method suggests a list of ranked similar queries based on previously issued queries. The clustering process is based on term-weight vector representation of the queries. The ranking is based on the similarity of the queries to the input query and the support, which measures how relevant the query in the cluster. In this work, they used existing query log from the Chilean Web[5] to create a list of ranked related recommendations.

3.2.2 CLASSIFICATION

Classification is the process of comparing representations of documents with representations of labeled categories and computing similarity to find to which category the documents belong. It is an essential process in our work because we need to classify the URI to find similar topic webpages. In our work, we are proposing using

---

[4]`http://encarta.msn.com/`
[5]`https://TodoCl.cl`

the classification technique to recommend similar webpages, using existing ontologies such as DMOZ. We need to understand and group our existing list of URLs. After that, we need to classify the requested URI by trying to map the URI to one of the pre-classified lists of URIs to select similar recommendations from them.

There are different types of webpage classification (Section 2.3). In our proposed work, we will focus on subject classification. We will also use different classification methods and compare results. Some of the methods that we will evaluate are the Baykan et al. [54, 1] method to classify the webpage using the URI and the Xue et al. [6] method where they classified text documents. Although Xue et al. [6] used text classification, not URI-based classification, the method proposed was on a hierarchical structure, which is the same type of structure we are going to use. These two related works and others are described below.

In 1997, Koller et al. [87] introduced a method where documents are classified hierarchically using very few words because of computational cost and complexity. This is accomplished by creating a classifier on each node in the classification tree. Other work included classifying a webpage using its URI without the need to use its content to improve the classification speed. In 1999, Attardi et al. [88] proposed automatic webpage categorization by using the link and context analysis. They proposed a method where they extract information about the webpage from the content of the URL referring to it.

Kan [89] researched categorizing the webpage using the URI by first using segmentation expansion then classifying. The first step is called information content reduction. Using the URL, it examines all possible partitions of the chunk and calculates the sum of the information content (IC) of each partitioning. The dataset used was the WebBase project [90] which holds the frequency of tokens on over 39 million webpages. The second step is called title token based finite state transducer (FST), where URIs are split and segments are expanded based on previously viewed webpage titles. An example is expanding "cs" to "computer science" based on a set of rules of scoring the title to match the segment. Next, the resulting tokens are fed to a machine learning algorithm to classify after changing tokens to IDs to be accepted to the SVM. The dataset tested had webpages in the following categories: student, faculty, course, and project. The features that were added to evaluate this work included URL text, anchor text, title text, and page text. Here they measured performance using F_1. Their findings include that proper use of URL alone is almost

as effective as using the page text itself, and it exceeds the performance of systems using the title or the anchor words. Also, IC and FST improve the result in about 12.7% and 18.6% of the cases, respectively.

In later research, Kan and Thi [91] added more features from the resulting segments before using them in the classifier. Some of the features were the URL component, URL length, sequential N-grams, and the precedence. Using the WebKB corpus [92] the results indicate that these features improve the results over existing URL features.

Kustanowitz et al. [93] proposed a three-component framework for search result categorization techniques, which are lean vs. rich, full-feature vs. fast-feature, and online vs. offline. The lean categories are simple categories with modest breadth and depth. The rich categories are extensive classification. Online categorization can be done for several reasons such as when searches are generated if the mapping of the page to the hierarchy is not essential. Offline categorization is required if no database exists to map search results to categories. The fast-feature requires only information provided in search result set. The full-feature requires the full text of the link destination. The online fast-feature method includes using the title, snippet, URI, domain, size, DMOZ, and pre-existing database map, to categorize Web search results. The offline fast-feature method is used for URL directory hierarchy parsing. They found that online fast-feature techniques show promise for quick categorization at query time. In this work, they try to improve categorization rates by making suggestions about how websites designers should redesign their websites to support fast categorization of search results.

Xiaoguang et al. [5] used neighboring websites to boost the classifying results of a webpage, such as a parent, children, sibling, and spouses. As shown in Figure 25, the parent webpage links to a target webpage, children webpages are the pages that the target links to, a sibling webpage is the webpage that had the same parent as the target, and a spouse webpage is the webpage that has the same child as the target. Each is divided into two subsets, labeled pages and unlabeled pages. They found that the method proposed can boost the classification accuracy of common textual classifiers by around 70%-90%. They also found the sibling pages are the most important.

Qi et al. [37] reviewed Web classification research concerning its features and algorithms. They described the different types of classification and how it is essential
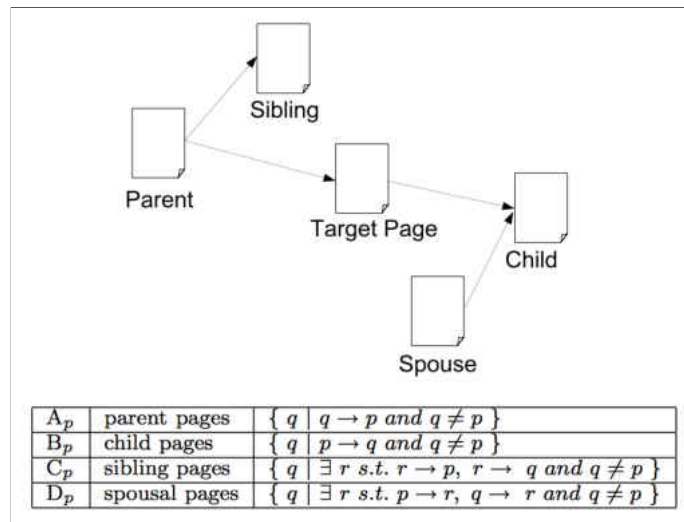
Fig. 25 Four kinds of neighboring pages of p [5] (Figure 1)

to many tasks in Web information retrieval, such as focused crawling and maintaining Web directories. They mentioned that since the Web is changing its content over time, its classification is more challenging than text classification. Also, they surveyed the published approaches to webpage classification from different viewpoints. The different features that can be used for Web classifications are using on-page classification such as textual content and tags and visual analysis, using features of neighbors. They discussed the fact that features from neighboring pages provide significant supplementary information to the page being classified. In addition, combining different methods can improve the classification as well.

Devi et al. [94] used the URI to classify the webpage and included some features, such as the URI length, content, token sequence, and precedence. In this work, they split the URI tokens and created a binary tree for the tokens. They classified URIs as student, faculty, and project pages. They found that this approach shows better performance than other methods.

Baykan et al. [54, 1] also investigated using the URI to classify the webpage and identify its topic (Section 3.1). They found that there is a relationship between classification and the length of the URI, where the longer URI, the better result. They used the following machine learning algorithms: Support Vector Machine (SVM), Naïve Bayes (NB), Maximum Entropy (ME), and Boosting by combining several algorithms. The highest scores were achieved by ME. The results of performing

Binary ME classifiers that are trained on the DMOZ dataset using all-grams method and are tested on Yahoo!, Wikipedia, Delicious, and Google are shown in Table 12, where the dataset is also grouped by category in the directories. It shows that Google got the highest F_1, with a score of 87, followed by Wikipedia with 85, DMOZ with 83 (shown as ODP), and both Yahoo! and Delicious with 79.

Table 12 Performance when Binary ME Classifiers are Trained on the ODP Dataset Using All-Grams, and are Tested on the Other Four Datasets [1] (Figure 37)

	ODP			Yahoo!			Wiki			Delic.			Google		
Topic	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Adult	90	85	88	-	-	-	-	-	-	73	71	72	-	-	-
Arts	81	83	82	77	80	78	84	90	87	79	82	81	87	88	87
Busin. (& Econ.)	80	84	82	80	84	82	78	87	82	80	77	78	89	85	87
Comp. (& Intern.)	82	80	81	79	78	79	90	93	91	81	82	81	87	87	87
Education	-	-	-	74	77	75	-	-	-	-	-	-	-	-	-
Entertainment	-	-	-	85	81	83	-	-	-	-	-	-	-	-	-
Games	89	88	88	-	-	-	93	90	91	80	76	78	91	86	88
Government	-	-	-	81	79	80	-	-	-	-	-	-	-	-	-
Health	85	79	82	83	81	82	78	81	79	86	85	85	90	86	88
Home	86	80	83	-	-	-	87	85	86	77	76	76	89	86	88
Kids & Teens	82	79	80	-	-	-	78	78	78	69	74	71	86	81	84
News (& Media)	81	82	81	77	78	77	74	81	77	78	78	78	83	83	83
Recreation	80	82	81	83	80	81	86	86	86	78	78	78	89	83	86
Reference	87	83	85	75	80	77	84	83	84	79	78	79	88	84	86
Science	83	81	82	80	81	80	91	89	90	83	83	83	87	83	85
Shopping	79	82	81	-	-	-	86	91	88	83	85	84	88	82	85
Social Science	-	-	-	74	75	74	-	-	-	-	-	-	-	-	-
Society (& Cult.)	81	81	81	76	77	76	-	-	-	80	81	80	87	85	86
Sports	87	84	85	-	-	-	92	88	90	86	84	85	91	91	91
Average	**83**	**82**	**83**	**79**	**79**	**79**	**85**	**86**	**85**	**79**	**80**	**79**	**88**	**85**	**87**

Xue et al. [6] used text classification on a hierarchical structure. They proposed a deep classification method, where given a document, the entire categories are divided into two kinds according to their similarity to the document, related categories and unrelated categories. They had two steps, the search stage and the classification stage. The method's algorithm steps are shown in Figure 26. The figure shows that after the search stage ends a small subset of candidate categories in a hierarchy structure would be the result. Then the output of the first stage would be the input of the second stage. For the first search stage, two strategies have been proposed. The first strategy is to use term frequency (TF) and Cosine similarity and find the
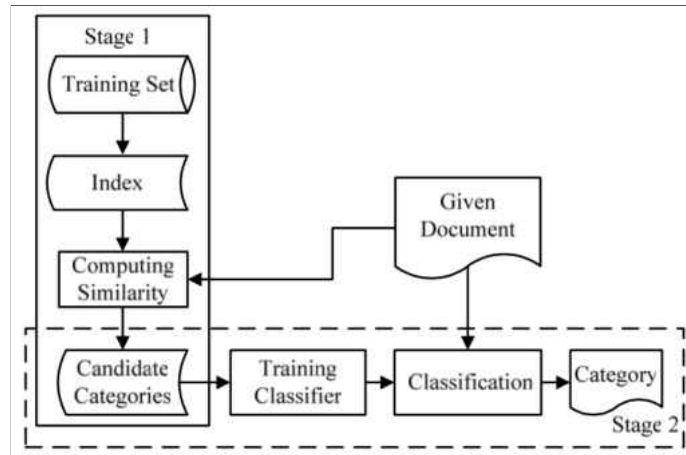
Fig. 26 Deep Classification algorithm steps [6] (Figure 1)

top N documents and get their categorization. The second strategy is category-based, where documents in each category are grouped, then they calculated the TF and cosine similarity for each category and returned the closest category. For the second stage, the resulting N category candidates are structured as a tree, then the tree is pruned by removing the category if it has no candidate in it. Three strategies are proposed to accomplish this step. The first is a flat strategy, where they considered candidates as a flat structure, without the use of ancestors. The second is pruned top-down strategy, where they used a classifier from the root to the last level. The third is the ancestor-assistant strategy, where they grouped the ancestors if there are no common ancestors with another candidate. This will help provide more candidates if there is a small sample size. They also used Naïve Bayes as a classifier because of the large sample and the speed desired. They used 3-gram because of the close similarity between categories. As a dataset they used DMOZ and ignored the regional and world categories. The remaining dataset was 1.3 million URIs. They used 130,000 documents for testing, 200 randomly picked documents for validation, and the remaining were used for training. For evaluation, they used the Mi-F_1 metric, which evaluates the performance for each level. An example is if the URI is classified to "computers/programming/language", they would check the accuracy of the result in "computers" then "computers/programming" until all levels are completed. In this work, they tested three methods of classification: the top-down level-based using SVM, the category based strategy, and the deep classification with the following options, the category based strategy, ancestor-assistant strategy,

and 3-gram NB. They found that the deep classification performs the best of the three using the Mi-F_1, where it resulted in 77% improvement over top-down based.

Rajalakshmi et al. [95] proposed an approach where N-gram based features are extracted from URIs alone, and the URI is classified using Support Vector Machines and Maximum Entropy Classifiers. In this work, they used the 3-gram features from the URL on two datasets, ODP with 2 million URLs and WebKB dataset with 4K URLs. Using this method on WebKB dataset resulted in an increase of F_1 by 20.5% compared to the related work [89], [91], and [96]. Also, using this method on ODP resulted in an increase of F_1 by 4.7% compared to the related work [97], [91], and [54].

Abdallah et al. [98, 99] introduced classifying the URI using N-gram language models. The all-grams approach was more successful than the earlier approaches but does not scale very well for large datasets. In this work, they found that the N-gram approach can capture the morphological variations in words. They showed that taking the lengths of the URIs into the consideration of the classification model did not improve its performance. Their main finding was that N-gram URI-based classification has high accuracy and performs well on large datasets.

3.3 ARCHIVING

To address RQ3, "Given a set of webpages discovered through the ontology, what metrics do we use to filter the best candidates for recommendation?", and since archiving is one of the methods to filter candidates, we will discuss related work to Web archiving and some of the evaluation methods that have been performed.

3.3.1 ARCHIVING API TOOLS

In recent years, there has been limited work in extending Web archive functionality. AlSum [49] created a Web archive service called ArcSys, which is a framework for tighter integration between the past and present Web. It helps third-party developers, researchers, and others to gain knowledge from the Web archive. ArcSys is a service framework that extracts, preserves, and exposes APIs for the Web archive corpus. It is built into three levels. The first level is ArcContent, which exposes the Web archive content processed through various filters. The second level is the metadata level which is extracted from the archived Web data and made available to users. The third level is to use the URI HTTP redirection status to enhance the user

query. They use Hadoop to process WARC files and provide metadata for webpages, such as incoming anchor text, HTML title, and page rank.

Warcbase is an open-source platform for managing Web archives built on the distributed datastore HBase [100, 101]. The system stores and manages raw content as well as metadata and extracted knowledge. It also uses Hadoop to process WARC files for extended service. This project is currently called the Archives Unleashed Toolkit (AUT)[6], formerly "Warcbase".

In addition, a similar framework to Warcbase is ArchiveSpark. ArchiveSpark was proposed by Holzmann et al. [102, 103, 104] and is a framework based on Apache Spark that builds a research corpus from the Web archive. The input files for ArchiveSpark are a WARC file with its corresponding CDX dataset. ArchiveSpark is considered a fast tool compared to other approaches.

In the area of selecting a list of websites for creating recommendations, AlNoamany et al. [105] used storytelling to enrich the live Web experience using Web archives. They identified, evaluated, and finally selected webpages from an archive collection. They used some filtering techniques such as detecting off-topic webpages, detecting near duplicates, detecting non-English webpages, and detecting damaged webpages. Then they arranged the pages and visualized them with the Storify service [106]. Some of the filtering methods will be discussed in Section 3.3.7 and Section 3.5 and used in our work.

3.3.2 EXPECTED LIFETIME OF A WEBPAGE

It is known that webpages disappear from the live Web, and there has been some work on measuring how long is the expected lifetime of a webpage. In 1997, Brewster Kahle [22] estimated that the expected lifetime of a webpage is 44 days. Lawrence et al. [107] collected computer science journals in May 2000, and found that the percentage of links to papers related to computer science that were invalid was 23% for papers published the previous year in 1999 and 53% for papers published six years earlier. Wallace Koehler [108] estimated that the half-life of random webpages is approximately two years. Nelson and Allen [109] found that in digital libraries 3% of the URIs were unavailable after one year. Diomidis Spinellis [110] performed a study on the papers published in *Communications of the ACM* and the *IEEE Computer Society*. He found that 28% of all URIs were unavailable after five years.

---

[6]http://archivesunleashed.org

He also estimated the half-life of a URL to be four years. Klein et al. [111] found that one in five Science, Technology, and Medicine (STM) articles have invalid references to them. Using the same dataset, Jones et al. [112] found that three out of four URI references lead to changed content. In this work, knowing that webpages disappear from the live web and having those webpages appear only in the archive made it an important to find and recommend webpages in the cold spots (Section 2.1.4).

3.3.3 USER INTENTION OF SEARCHING THE ARCHIVE

Webpages link one page to another and having them both in the live Web and in the archive makes them easy to find. However, over time webpages and their inner links may disappear from the live Web, making them hard to find.

AlNoamany et al. [32] analyzed who and what links to the Internet Archive and detected that users access the archive when they do not find the webpage they requested on the live Web. They found that about 65% of the requested archived pages no longer exist on the live Web. Also, more than 82% of human sessions connect to the Wayback Machine are via referrals from other websites. On the other hand, only 15% of robots have referrers.

Since we are working on what the user requests from the archive, knowing what the users are searching in the archive is helpful. Costa et al. [113] addressed the issue of why, what, and how users search the Web archives. This research aimed to understand the information needs of Web archive users. This work revealed that the users' intent could be navigation, in which users usually do not restrict archive search by date, that 21% of queries had URIs only, and that half of the users' searches are for names, people, places, and things. In addition, users typically search for the oldest document. This work also described the necessary information of what is looked for and that recommending old webpages, such as the ones that are in cold spots, is beneficial for the user.

3.3.4 MEASURING ARCHIVED WEB RESOURCES

Bias in the Internet Archive was addressed by Thelwall et al. [114]. They found that there are indeed large national differences in the archive's coverage of the Web. However, they mentioned that the bias in coverage is not intentional but that researchers need to be aware of this issue.

McCown et al. [115] addressed coverage as well but limited it to search engine

caches such as Ask, Google, MSN, and Yahoo. They also examined the overlap of the various caches with the holdings of the Internet Archive.

Ainsworth et al. [48] and AlSum [49] measured how much of the Web is archived. They used DMOZ, Delicious, Bitly, and search engine indexes to measure the number of archived copies available in various public Web archives. They found that 35-90% of URIs had at least one archived copy. Also, they found that 17-49% have two to five copies, 1-8% have six to eight copies, and 8-63% have at least ten copies. They also found that 14.6-31.3% of URIs are archived more than once per month. The percentage of archived webpages ranged from 16-79% in 2010 and increased to 33-95% in 2013.

In previous work [50], we addressed how well Arabic language webpages are archived. We used language detection tools to check if the webpage is in Arabic. We found that 46% of the URIs are not archived, and 31% are not indexed by Google. We also found that only 14.84% of the URIs had an Arabic country code top-level domain (e.g., .sa). Later, we extended the work to include other languages such as English, Danish, and Korean [51]. We found that English has a higher archiving rate than Arabic, with 72.04% archived. However, Arabic has a higher archiving rate than Danish and Korean, with 53.36% of Arabic URIs archived, followed by Danish and Korean with 35.89% and 32.81% archived, respectively. Also, we found that the presence of a webpage in a directory such as DMOZ positively impacts indexing and archiving.

3.3.5 SEARCHING THE ARCHIVE WITHOUT INDEXING IT

Kanhabua et al. [116] proposed a search system to support retrieval and analytics on the Internet Archive. They used the live Web, such as Bing, to search for the query and match the result to the archive, using a user-friendly interface. The features they added to the archive were (a) keyword search, (b) query auto-completion using Wikipedia entity index, (c) query suggestion using Wikipedia articles and building an entity graph to find related queries, (d) multilingualism supporting English and German languages, and (e) a list of ranked results. They also measured the coverage of the archived content retrieved by the current search engine and found that on page one of Bing results, 94% are available in the Internet Archive. On the other hand, on pages two to five, only 87-89% are archived. They also measured the amount of the search result change over time for different entity categories over the period of

2, 5, and 7 months, and found that after two months, approximately 47% of the top 100 URLs are not returned anymore, and after seven months this increases to 60%.

In our proposed work, we will not consider recommendation results from the live Web until there is no other option available, due to the cost of complexity, time, and storage. Also, we will focus on webpages that are in the cold spots.

3.3.6 RECOVERING MISSING WEB RESOURCES

Link rot is when a link points to a Web resource that has become unavailable. The HTTP response code for this is `404 Page Not Found` (Section 2.2.4).

Francisco-Revila et al. [117] developed a tool called Walden's Paths Path Manager that allows users to construct trails using webpages which are usually authored by others. They also addressed discovering relevant and significant changes to websites.

Martinez et al. [118, 119, 120] introduced a method to recover link rot based on anchor text, the webpage containing the link, or a combination of both. Then the selected information is processed and submitted to the search engine. They proposed an algorithm based on information retrieval techniques to select the most relevant information and ranked the candidate pages for the search engine to get the best replacements. They found that expanding the query with the anchor text data can improve the retrieval performance.

Harrison and Nelson [121, 122] developed a server-side system called OPAL which uses search engines to locate a missing webpage. They used cached copies on the Web and after that OPAL creates a lexical signature (LS) which is a small set of terms extracted from a document that captures the aboutness of that document. The resources used to find the cached copies were search engines such as Google, the Internet Archive's Wayback Machine, and research projects (e.g., CiteSeer and NSDL). Then the LSs are used to search for similar versions on the Web.

Klein and Nelson [123] also used LSs to re-discover webpages at different URLs as a replacement for missing webpages. First, they send the requested URL to the live Web, and if the webpage is not live, they request it from the archive. Next, they create the LS from the archived resources. Finally, they use the LS to issue a query to one or more Internet search engines such as Google, and receive the correct URL of the webpage that is considered missing by matching it to the archived URL. In this work, they distinguish between two types of overlap in LSs, rooted and sliding. Rooted overlap is the overlap between LSs of all the URL occurrences. Sliding overlap

is the overlap between LSs of two consecutive years of the URL occurrences. The two different types of overlap showed that LSs decay over time, and that LSs should be created recently since the content of a webpage might change over time. They created LSs of websites from the last 12 years. The number of LS terms created for each webpage was 2-10 terms. Their results showed that the best number of LS terms to return the correct URL in the top ten of search engine results was with 5-, 6-, and 7-term LSs. They found that 50% of the URLs got the correct result in the number one result. However, 30% of the URLs were not discovered. This work did not recommend other webpages; however, this revealed that each URI-M may have a different classification over time.

Klein et al. [12] recommended missing webpages from the live Web using four retrieval methods: (1) LS, (2) webpages' titles, (3) tags, and (4) link neighborhood lexical signatures (LNLS). Using these four methods helped to get a replacement of missing webpages. Various datasets were used, including DMOZ. By comparing the different methods, they found that 70% of the webpages were recovered using the title method. Titles were at least as well performing as lexical signatures and were easier to obtain. They found that 7- and 5- term LSs performed best. The tag-based method that was used from Delicious performed poorly when used by itself. The most expensive to generate are LNLS, so it was used as a last resort for the rediscovering of missing webpages. The result increased to 77% by increasing the complexity and combining the other three methods. This shows that the webpages have moved to a different location and need rediscovering. They introduced Ghost tags as "tags that describe previous versions of webpages better than current ones". In their work, the user will get a single alternative when a page is not found on the live Web. However, in our proposed work, we want to recommend several other similar webpages and we might need to adopt some of the methods proposed when needed.

Popitsch et al. [124, 125, 126] introduced a tool called DSNotify to handle broken links in linked data environments. The tool detects broken links and assists the data source in fixing them, e.g., when they are moved to other Web locations. The tool is used in several ways: (1) to function as a detection and correction module in existing software, (2) as a standalone notification service that keeps subscribed clients informed about changes in a data source, and (3) as an indirection service that automatically forwards requests for moved resources to their new location.

Huurdeman et al. [7, 127, 128] detailed their approach to recover pages in the

unarchived Web based on the existence of links and anchors of crawled pages. The data used was from the Dutch 2012 National Library of the Netherlands[7] (KB). The link extraction used Hadoop MapReduce and Apache Pig to process all archived webpages and used JSoup to extract links from their content. Both external links (inter-server links), which are links between different servers, and site internal links (intra-server links), which occur within a server, were included in the dataset. The Aura is defined as the set of webpages that are not in the archive but are known to have existed based on references found in archived webpages. There are four main classifications of URLs found in this dataset, shown in Figure 27:

- Intentionally archived URLs in the seed list. 92% of the dataset, and 10.1 million webpages.

- Unintentionally archived URLs due to crawler configuration. 8% of the dataset, and 0.8 million webpages.

- Inner Aura, unarchived URLs where the parent domain is included in the seed list. 5.5 million webpages, where 20% are pages of depth 4.

- Outer Aura, unarchived URLs where the parent domain is not on the seed list. 5.2 million webpages, where 29.7% are pages of depth 2.
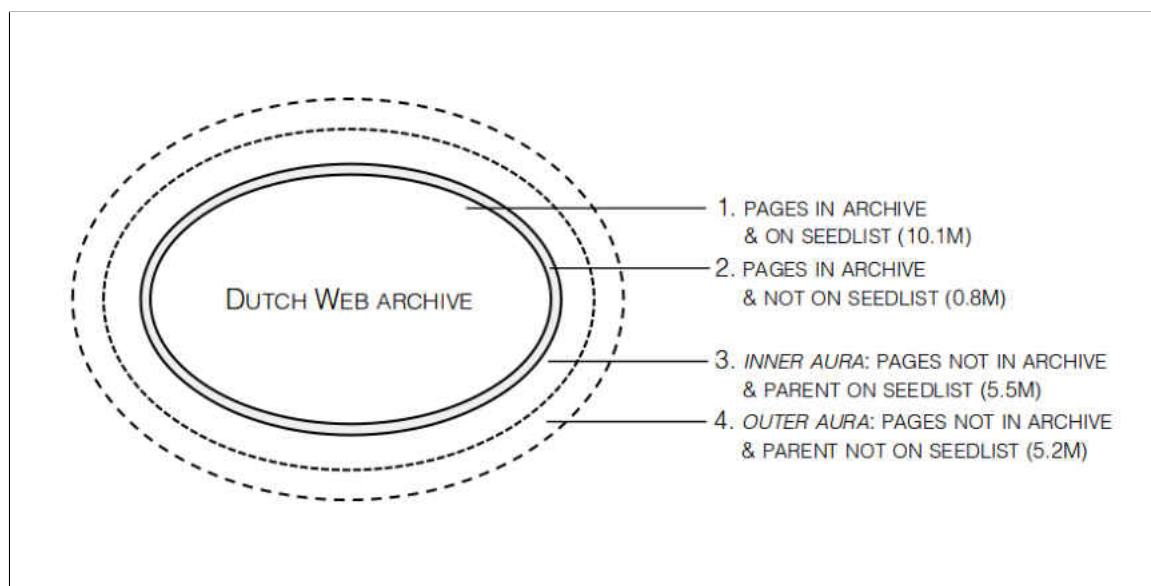
Fig. 27 Classification of URIs [7] (Figure 1)

---

[7] https://kb.nl/en

Their findings included that the archived pages show evidence of a large number of unarchived pages and websites. They also found that only a few homepage webpages have rich representations. Finally, they found that even with a few words to describe a missing webpage, they can be found within the first hit.

Since there are a large number of unarchived pages, we assume that there will be many requests to the archive that may result in `404 Page Not Found` status. In our work, we want to recommend other similar webpages that are in the archive.

3.3.7 OFF-TOPIC MEMENTOS

In addition to filtering the candidate URIs based on archiving, we can also filter the candidates based on the aboutness of the webpage, which is measuring how much a text is about a certain topic. This method can also be used to address RQ4, "Given a set of candidate webpages for recommendation, how do we rank the candidates?". The content similarity could be the similarity of URI tokens or the similarity of the webpage content.

AlNoamany et al. [129] presented a method to detect off-topic pages within TimeMaps in Web archives where in some cases the archived webpage topic changes over time. In this work, they evaluated six different methods to detect when the page has gone off-topic through subsequent captures. The methods are (a) cosine similarity, (b) Jaccard similarity, (c) the intersection of the 20 most frequent terms, (d) Web-based kernel function, and (e) the change in size using the number of words and content length. They used three Archive-It collections to evaluate the proposed methods at different thresholds. They found that combining cosine similarity at threshold 0.10 and change in size using word count at a threshold of -0.85 performs the best, with an accuracy of 0.987 and an F_1 0.906. They found the average precision of detecting the off-topic pages is 0.89. In later work, Jones et al. [130] created a tool called Off-Topic Memento Toolkit (OTMT), which detects off-topic mementos in a web archive collection. They used the following methods to detect off-topic mementos: byte count, word count, cosine similarity, Jaccard distance, Sørensen-Dice distance, Simhash using raw text content, Simhash using term frequency, and Latent Semantic Indexing via the gensim library. Using a gold standard dataset generated manually, they found that the word count has the best F_1.

3.4 SOFT 404s

Bar-Yossef et al. [131] proposed a method to identify soft 404s. They listed three reasons why a webpage may be not publicly available over the Web. The first reason is the URI could be malformed. The second reason is that its host is down or does not exist. The last reason is that it does not exist on the host. When fetching a page that is not found, an error message should appear. However, soft 404s may appear, which are hard to detect. To detect the soft 404s, they proposed sending a second request to the site with a string of random characters appended the URI to compare the content similarity of the two responses. Assuming that soft 404s responses are similar regardless of the request, such sites could be isolated.

Lee et al. [132] proposed another approach to detect soft errors by analyzing redirection logs collected during crawling operation. Three main observations were made based on the logs. A soft error redirection from a webpage to a target is likely to have a large number of redirections to the same target. A redirection from a host with a substantial number of redirects to the same target is likely to be a soft error redirection. A legitimate redirection to a server is likely to have a large number of distinct target hosts reached by redirections from a host. Based on this observation, a scoring system has been designed to determine soft errors. Using this method not only determines soft errors redirections, but also determines 403 and 5xx errors redirections effectively.

Meneses et al. [133] developed two methods that use a Naïve Bayes classifier based on known valid responses and known 404 responses. The classifier was able to predict soft 404 pages with a precision of 99% and a recall of 92%.

3.5 RANKING FEATURES

To also address RQ4, we need to rank the results based on several dimensions. Some of the dimensions are presence in a cold spot, datetime of the webpage, and memento damage.

3.5.1 MEMENTO DAMAGE

One of the features we can use to rank the candidate URIs is the archival quality. Archival quality refers to measuring memento damage by evaluating the number and importance of missing resources in a webpage. The missing resources could be text,

images, video, audio, style sheet, or any other type of resource on the webpage.

Brunelle et al. [10] proposed a damage rating algorithm to measure the relative value of embedded resources and evaluate archival success. The algorithm is based on a URI's MIME type, size, and location of the embedded resources. Different resources are more important than others depending on the webpage. They found that almost 2.05 significant embedded resources per memento on average are missing over time. In the Internet Archive the average memento damage reduced from 0.16 in 1998 to 0.13 in 2013. Related to this work, Siregar [134] created a Memento Damage tool[8] to estimate the damage of the memento or a live webpage using an online service or a local service. This calculation is used in one of the features we use to rank the candidates (Section 5.6.1).

3.5.2 RETRIEVABILITY

Another feature that we will use to rank the candidate URIs is webpage popularity, which considers how often the webpage has been archived, its popularity on the live Web, and frequency of requests in the archive. A particular case of popularity are webpages in "cold spots", which are pages that are not on the live Web and may not have frequently been requested from the archive.

Related work has been proposed to measure the retrievability of the documents. Azzopardi et al. [135] quantified the webpages that are hard to find or impossible to retrieve. The retrievability score counts how often a document is retrieved as one of the top K documents by a given set of queries. They showed how the retrieval system affects the users' ability to access information. They found that in a TREC-style evaluation, 80% of the collection was not retrieved due to the bias it exhibits over the collection of documents. They also found that the least retrievable documents within the collection are significantly harder to find than the rest of the collection.

In a similar work, Traub et al. [136] investigated how well retrievability studies represent the search behavior of real users and how they can be applied to an extensive collection of digitized documents. Here they calculated retrievability score for every document in the collection for three different IR models. This work was applied to three datasets: National Library of the Netherlands (KB) (2,732,139 queries), a real user query log collected between March and July 2015 (957,239 queries), and a simulated dataset (4,000,000 queries). They used Gini coefficient and Lorenz curves,

---

[8]http://memento-damage.cs.odu.edu/

which are used to measure and visualize a degree of inequality in societies [137]. They found that simulated queries differ from real queries regarding term frequency, the prevalence of named entities, and retrievability results. In both collections, they found a substantial fraction of documents were never retrieved. However, there was no reliable evidence of technical bias in retrievability.

3.5.3 CREATION TIME OF WEBPAGES

Another feature that we can use to rank the candidate URIs is temporal similarity, which can refer to how close the candidate webpage's creation date, memento-datetime, or content date are to the requested URI (Section 2.1.2).

SalahEldeen et al. [8] created a tool called Carbon Date[9] to estimate the creation date of a Web resource. They illustrated a timeline of resources along with how they estimated the age of the resource in Figure 28. They used a group of sources of evidence and returned a machine-readable structure with their respective values. The resources used to determine the creation time were:

- Bitly[10]: a URI shortening service, the tool uses the first time someone shortened the URI.

- Topsy: a social analytics service, the tool uses the first time someone tweeted the URI. This tool is no longer operational [138].

- Memento aggregator[11]: gathers information on mementos, the tool uses the first time the URI appeared in a public Web archive.

- Google's time of the last crawl.

- Last-Modified HTTP response header of the resource itself.

The method was able to automatically estimate a creation date for 75.90% of the resources, where 32.78% of the dates were accurate. Since topsy is no longer working, a recent follow-up on this tool was made by Atkins [139]. The new service added additional detection methods, such as Bing searching, Twitter searching, and pubdate parsing [140].

---

[9]http://carbondate.cs.odu.edu
[10]https://bitly.com
[11]http://timetravel.mementoweb.org
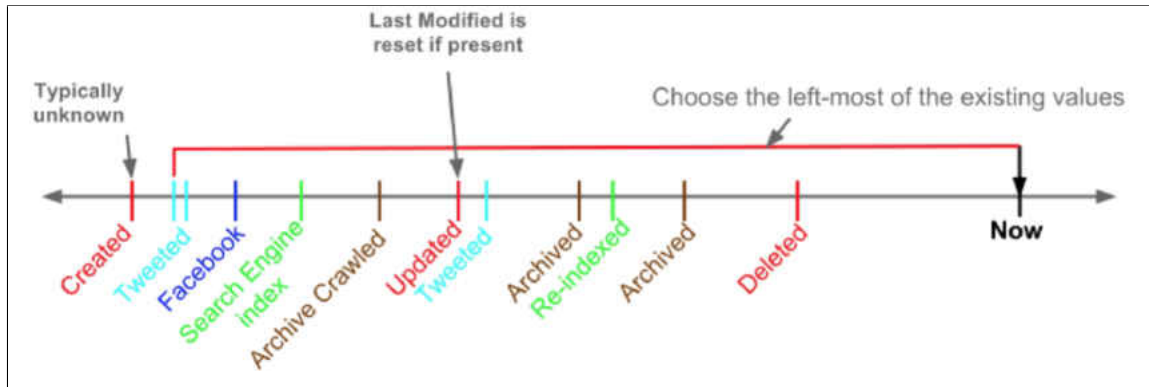
Fig. 28 The timeline of a shared resource and the proposed process of carbon dating [8] (Figure 1)

3.6 SUMMARY

In this chapter, we presented an overview of the research that is related to our work. We presented work related to splitting compound words and how they split URIs specifically. Then we presented related work on clustering and classifying documents. Next, we presented archiving and showed some archiving API tools. We also presented archiving in different angles such as the expected lifetime of a webpage, the user intention of searching the archive, measuring the archived Web resources, searching the archive without indexing it, and recovering missing resources. After that, we discussed off-topic mementos and defined soft 404s. Finally, we also presented the related research on memento damage, retrievability, and webpage creation time. In the next chapter, we describe the datasets that we are going to use.

CHAPTER 4

DATASETS

In this chapter, we describe the three main datasets we use: DMOZ, Wikipedia, and a Wayback Machine access log. DMOZ and Wikipedia are used as ontologies that help classify the requested URIs and provide recommendations. The Wayback Machine access log consists of real requests made to the archive that we used to evaluate the framework. Some features of the datasets could affect the archiving rates of the webpages, such as the language of the webpage (Section 3.3.4), so in this work we will focus only on English webpages.

4.1 DMOZ

DMOZ, or the Open Directory Project (ODP), is a widely used dataset. We use DMOZ as one of the ontologies to classify a requested webpage (Section 5.3). We have archived 118 DMOZ files of the type Resource Description Framework (RDF) [141], from 2001 to 2017 (Table 13). The dataset consists of four fields: the category, the URI, the webpage title, and the description (Section 2.4.1, Table 8). Each URI and its fields are called an instance. We collect all instances from DMOZ. Since we will focus on English language webpages, we first filtered out the World category. Next, starting from the latest archived dataset (March 12, 2017) to the earliest (January 22, 2001), we collected the instances that include a unique URI. After that, we converted all the URIs to SURT format (Section 2.2.2). Table 14 shows the number of collected instances and sub-categories for each of the 17 categories. There were 6,522,125 instances after filtering out instances that contain an empty field.

Since we are going to get recommendations from DMOZ, we wanted to analyze the dataset. We checked the top level domains (TLDs), the depth of URIs, and if URI patterns occur.

Table 13 Archived DMOZ dataset over the years

Year	Number of files	Archival Date MM-DD
2017	2	03-12, 02-26
2016	4	03-06, 03-20, 04-17, 09-11
2015	3	08-30, 09-20
2014	4	04-06, 07-20, 09-14, 10-19
2013	4	01-26, 04-21, 08-18, 09-08
2012	5	01-29, 05-27, 07-15, 10-14, 10-21
2011	8	02-18, 03-08, 05-15, 06-05, 07-03, 07-17, 08-07, 09-25
2010	7	01-19, 03-04, 05-11, 06-22, 08-12, 09-02, 10-03
2009	11	01-06, 02-03, 03-03, 04-07, 06-01, 07-21, 08-17, 09-01, 10-06, 11-09, 12-02
2008	12	01-08, 02-05, 03-04, 04-01, 05-06, 06-03, 07-01, 08-05, 09-02, 10-07, 11-04, 12-02
2007	12	01-25, 02-12, 03-06, 04-03, 05-01, 06-05, 07-03, 08-07, 09-04, 10-02, 11-20, 12-04
2006	10	01-03, 02-07, 03-07, 04-04, 05-03, 06-06, 07-11, 08-01, 09-05, 10-03
2005	9	01-04, 02-01, 05-03, 06-07, 07-05, 09-06, 10-04, 11-01, 12-06
2004	12	01-05, 02-23, 03-01, 04-01, 05-04, 06-01, 07-06, 08-03, 09-07, 10-05, 11-09, 12-07
2003	8	02-10, 03-31, 06-09, 07-07, 08-04, 09-01, 11-03, 12-01
2002	5	02-14, 03-14, 04-25, 05-16, 09-19
2001	2	01-22, 10-20

4.1.1 DMOZ DIVERSITY MATRIX

Alexander Nwala created a diversity matrix to quantify URI diversity [142]. This matrix is called WSDL diversity index. The process includes URI processing such as scheme removal and parameters and fragment removal. Then the dataset is scored based on diversity. In our case since we only have unique URIs, the URI diversity score is 1.0. The diversity index also measures hostname diversity, and the score of hostname diversity in DMOZ dataset is 0.55. In addition, it measures the domain diversity, and the score of domain diversity in DMOZ dataset is 0.49.

Table 14 The number of instances for each category and the number of sub-categories in the DMOZ dataset

Category	Num. URIs	Num. sub-categories
Regional	2,348,257	297,140
Arts	658,942	57,959
Society	487,834	36,259
Business	469,668	22,465
News	421,800	2,581
Computers	297,789	12,580
Sports	278,706	28,761
Recreation	261,005	15,467
Shopping	250,538	7,393
Science	217,071	17,212
Adult	197,141	10,683
Reference	160,652	13,077
Games	151,459	20,233
Health	149,648	10,292
Home	81,059	3,553
Kids_and_Teens	63,333	5,793
Netscape	27,223	2,581
Total	6,522,125	564,029

4.1.2 DMOZ TOP LEVEL DOMAIN

In this section we wanted to determine the diversity of the TLDs in DMOZ. Shown in Table 15, we found that 61.85% URIs are commercial top level domain, `.com`, followed by `.org`, `.net`, `.edu`. Other top level domains include `.ca`, `.it`, etc. We found 2,824 unique TLDs in the others category that include effective TLDs, such as `co.uk`.

4.1.3 DMOZ URI DEPTH

Here, we want to know if the URIs we are recommending are only URIs of depth 0. The URIs are first canonicalized. Note that depth 0 includes URIs ending with `/inde x.html` or `/home.html`. Some of the extensions in the path that were removed include `index.html`, `index.htm`, `index3.php`, `default.html`, `default.htm`, `default.asp`, `homepage.html`, `homepage.htm`, `home.html`, and `home.html`. For example, the URI `https://batteries4everything.com/index.html` was considered depth zero. The

Table 15 Top level domain analysis for DMOZ dataset

TLD	Num. URIs	Percent
com	4,034,276	61.85%
org	586,152	8.99%
net	371,753	5.70%
edu	224,539	3.44%
gov	60,919	0.93%
us	11,382	0.17%
others	1,233,105	18.90%
Total	6,522,125	100%

depth is measured after URI canonicalization using SURT. Shown in Table 16, we found that 50.57% of the URIs in DMOZ are depth 0, essentially top-level webpages. We think this is because top level webpages are more often categorized and indexed in directories, rather than deep level webpages. We also found that the longest depth is 21.

Table 16 Depth analysis for DMOZ dataset

Depth	Count	Percent
0	3,298,369	50.57%
1	1,134,874	17.40%
2	905,849	13.89%
3+	1,183,033	18.14%
Total	6,522,125	100%

4.1.4 DMOZ URI PATTERNS

In this section we calculate the different URI patterns that occur in DMOZ. Table 17 shows the percentage of occurrence of the pattern in the hostname and the path. We analyze the following patterns:

- **Long strings, not separated by non-alphanumeric character**. Contains 10 or more contiguous letters. We chose 10 because it is likely that at least two words are grouped together since the average English word is 5 letters long [143, 144].

  - Example in the host: `http://radiotunis.com`

– Example in the path: `http://moviefreak.com/reviews/m/matrixrevo`
`lutions1a.htm`

- **Long strings in the domain, separated by non-alphanumeric character**. Contains 2 or more instances of 5 contiguous letters separated by non-alphanumeric character in the domain.

  – Example in the host: `http://bruno-groening.org/afrikaans/defaul`
  `t.htm`

- **Long slugs in the path**. A slug is usually the title of the webpage included in the end of the webpage's path, commonly used in news media webpages. To search for slugs we search for URI path that contains 2 or more instances of 5 contiguous letters separated by non-alphanumeric character in the path.

  – Example in the path: `http://www.arnosoftwaredev.blogspot.com/20`
  `05/01/sorting-algorithms-visualized.html`

- **Numbers**.

  – Example in the host: `http://911.com`

  – Example in the path: `http://comic-art.com/biographies/kelly001`
  `.htm`

- **Change in case**.

  – Example in the host: `http://AmericaInArabic.com`

  – Example in the path: `http://zeekoo.com/ZeeKooGids.php`

- **Query in the path**. HTTP query string, beginning with a "?".

  – Example in the path: `http://findagrave.com/cgi-bin/fg.cgi?page=`
  `gr&GRid=1795`

- **Port number in the hostname**.

  – Example in the host: `http://www3.gencat.cat:81/justicia/justiter`
  `m/index.htm`

- **IP address instead of the hostname**.

– Example in the host: `http://63.135.118.69/`

- **Percent-encoding**. Encoding to represent special characters in the URI.

  – Example in the host: `http://haus-wohnungsaufl%c3%b6sung.de`

  – Example in the path: `http://tinet.cat/%7ekosina`

- **Date string**.

  – Example in the host: `http://03-03-1952.homepage.t-online.de/universum.html`

  – Example in the path: `http://elmundo-eldia.com/1999/08/29/opinion/1001023218.html`

We found 42.65% of the URIs contain long strings in the hostname, and 20.01% of the URIs contain numbers in the path, as shown Table 17.

Table 17 URI patterns present in DMOZ

Pattern	% in hostname	% in path
Long strings, not separated	42.65%	13.21%
Long strings, separated	10.85%	-
Long slugs	-	7.82%
Numbers	4.37%	20.01%
Change in case	0.36%	8.18%
Query	-	4.72%
Port number	0.11%	-
IP address	0.07%	-
Percent-encoding	0%	0.50%
Date	0%	0.43%

4.2 WIKIPEDIA

Articles in Wikipedia contain a summary about the topic, external links, references, categorization of the article, and an information box. References are links that appeared within the article, however external links are links to webpages outside Wikipedia, and usually do not appear in the article (Section 2.4.2). In our work, we consider a match if the requested URI `http://odu.edu` is the "official webpage" (Figure 29), and we will use the article's category as the category we are going to use

for the requested URI. For instance, the article shown in Figure 29 is categorized as multiple categories, such as, *Old Dominion University, Universities and colleges in Virginia, Educational institutions established in 1930*, and *Public universities*. If the entity described in the article has an "official webpage", we consider the requested URI to be a match. We use "official webpage" as a keyword that we search for. We use Python Wikipedia packages [145, 146] to extract the information needed. For analyzing the dataset we used the external links found in the Wikipedia dump[1] from February 01, 2019, which contains 125,900,205 URIs. However, some URIs are links to the Internet Archive, so we collected the URI-Rs of these requests. Note that 3,703,149 unique URIs are external links to the Internet Archive. By removing duplicate URIs in the links to the archive, the resulting number of URIs is 123,101,129. Although some of the external links are not official links, we still want to analyze the URIs.

4.2.1 WIKIPEDIA DIVERSITY MATRIX

We measure the Wikipedia external links set using WSDL diversity index (Section 4.1.1). The URI diversity score is 1.0, since we only have the unique URIs in our dataset. The hostname diversity is 0.03, and the domain diversity is 0.02. The results indicate that there are a large number of redundant hostnames and domains in the Wikipedia external links dataset.

4.2.2 WIKIPEDIA TOP LEVEL DOMAIN

In this section we wanted to determine the diversity of the TLDs in Wikipedia's external links. Shown in Table 18, we found that 38.05% URIs are commercial top level domain, `.com`, followed by `.org` with 34.95%. Other top level domains include `.ca`, `.it`, etc.

4.2.3 WIKIPEDIA URI DEPTH

Here, we want to know if the URIs we are recommending are only URIs of depth 0. The depth is measured after URI canonicalization using SURT. Shown in Table 19, we found that only 3.40% of the URIs in Wikipedia's external links are depth 0, essentially top-level webpages, 30.21% of the URIs are depth 1, 34.19% of the URIs

---

[1] `https://dumps.wikimedia.org/enwiki/20190201/`

(a) Wikipedia webpage `https://en.wikipedia.org/wiki/Old_Dominion_Uni versity` resulted after searching for `odu.edu`

(b) The end of the Wikipedia webpage `https://en.wikipedia.org/wiki/Old_D ominion_University`, showing an external link `http://odu.edu` as an "official website", and its corresponding categories

Fig. 29 Searching for the request `http://odu.edu` in Wikipedia resulted in finding the Wikipedia webpage `https://en.wikipedia.org/wiki/Old_Dominion_University` that contains the requested URI as the official page in the External links section. We use other webpages in the same categories (at the end of the page) as candidate webpages.

Table 18 Top level domain analysis for Wikipedia's external links dataset

TLD	Num. URIs	Percent
com	46,836,894	38.05%
org	43,022,108	34.95%
gov	6,117,306	4.97%
net	2,836,749	2.30%
edu	1,915,075	1.56%
us	75,775	0.06%
others	22,297,222	18.11%
Total	123,101,129	100%

are depth 2, and 32.07% are depth 3 or more. We also found that the longest depth is 20. We think that this is because most of the external links points to deep webpages, as opposed to DMOZ that includes a higher level classification of the domain.

Table 19 Depth analysis for Wikipedia's external links dataset

Depth	Count	Percent
0	4,190,333	3.40%
1	37,194,370	30.21%
2	42,082,218	34.19%
3+	39,475,560	32.07%
Total	123,101,129	100%

4.2.4 WIKIPEDIA URI PATTERNS

Here, we calculate the different URI patterns that occur in Wikipedia's external links. Table 20 shows the percentage of occurrence of the pattern in the hostname and the path. We found 33.92% of the URIs contain long strings in the path and 77.28% of the URIs contain numbers in the path.

4.3 WAYBACK MACHINE ACCESS LOG DATASET

The goal of our work is to use requests to an archive and recommend other webpages. In our framework, we will use real requests made from users as inputs and provide a list of ranked webpages as recommendations. We will use the Wayback Machine access log dataset as it contains real requests to the Internet Archive's Wayback Machine. These requests could be a URI-M request or URI-T request

Table 20 URI patterns present in Wikipedia's external links

Pattern	% in hostname	% in path
Long strings, not separated	19.97%	33.92%
Long strings, separated	13.48%	-
Long slugs	-	39.98%
Numbers	1.69%	77.28%
Change in case	0.12%	42.32%
Query	-	44.39%
Port number	0.03%	-
IP address	0.16%	-
Percent-encoding	0.01%	15.78%
Date	0%	6.88%

(Section 2.1.1). This dataset has been used in other work such as in AlNoamany et al. [147, 32, 105].

Our Wayback Machine server access log dataset contains requests from 295 non-contiguous days between 2011-01-01 to 2012-03-26. Figure 30 shows a calendar heatmap to indicate the number of requests in the log. Light green means few requests, and dark green means more requests. The log does not include data from March through the end of July 2011. The size of the complete dataset is approximately 820 GB. Each file in the log represents a day's worth of records. The size of those files varies, some as low as 40 B and some as large as 2.5 GB.

Each file in the log is a text file where each request (line) contains some information, as shown in Listing 3. Listing 4 shows an example of a single record in the log, where the client IP has been anonymized for privacy reasons.
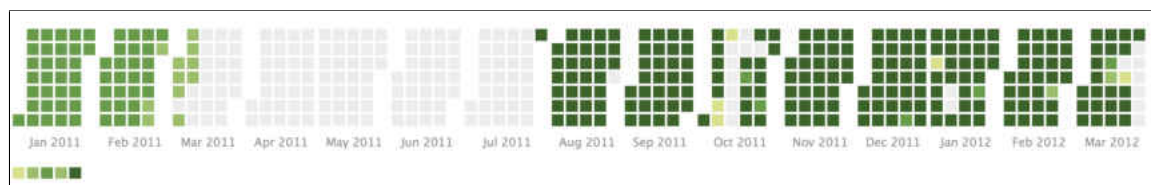
Fig. 30 Size of each day's datafile in the Wayback Machine Access Log
(graphic generated by Mat Kelly)

Listing 3 Log record fields

```
    Client IP, Access Time, HTTP Request Method, URI,
        Protocol, HTTP Status Code, Bytes Sent, Referring
        URI, User-Agent
```

Listing 4 A sample URI-M query log

```
    0.206.107.149 web.archive.org - [31/Jan/2011:00:00:15
        +0000] "GET http://web.archive.org/web
        /20070217154221/http://www.cnn.com/2005/SPORT/07/06/
        singapore.olympics/index.html HTTP/1.1" 200 12954 "
        http://en.wikipedia.org/wiki/
        Bids_for_the_2012_Summer_Olympics" "Mozilla/5.0 (
        Windows; U; Windows NT 6.0; en-GB; rv:1.9.2.13)
        Gecko/20101203 Firefox/3.6.13 ( .NET CLR 3.5.30729;
        .NET4.0C)" TCP_MISS:SOURCEHASH_PARENT/207.241.227.93
         4583
```

In Listing 4 the fields are as follows:

- Client IP (anonymized) = 0.206.107.149

- Server = web.archive.org

- Access Time = 31/Jan/2011:00:00:15 +0000

- HTTP Request Method (GET or HEAD) = GET

- URI = `http://web.archive.org/web/20070217154221/http://www.cnn.co`
  `m/2005/SPORT/07/06/singapore.olympics/index.html`

- Protocol (HTTP) = HTTP/1.1

- HTTP status code (200, 404, etc.) = 200

- Bytes Sent = 12954

- Referring URI = `http://en.wikipedia.org/wiki/Bids_for_the_2012_Sum`
  `mer_Olympics`

- User-Agent = Mozilla/5.0 (Windows; U; Windows NT 6.0; en-GB; rv:1.9.2.13)
  Gecko/20101203 Firefox/3.6.13

4.3.1 WAYBACK MACHINE ACCESS LOG FILTER

The Wayback Machine access log contains requests from both humans and robots. We need to separate those requests to use only the human requests as input to our framework. The purpose of filtering the requests is to have a list of URIs that are requested by humans. We predict that human requests have meaningful URI tokens. However, robot requests may be composed of random letters that are meaningless. This list will not be useful in our case. In addition, we want to find recommendations to requests made by users.

AlNoamany et al. [147] proposed a method to filter the Wayback Machine server access log dataset and separate human vs. robot requests. The steps include data cleaning, user identification, session identification, and robot detection. They found that regarding sessions, robots outnumber humans 10:1. This work was performed on a small dataset (week of Feb 8, 2012). To get the human requests only, we filtered out the robot requests and adopted some steps used in AlNoamany et al. [147].

In our work, we will use a sample from the requests made on Feb 8, 2012, similar to data selected in AlNoamany et al. [147]. There were 49,026,577 requests on that day. Before collecting a sample to use, we filtered out requests with the following characteristics and show the number of remaining URIs at each step:

1. did not result in an HTTP 200 status code (49,026,577 → 14,137,350)

2. with a non-HTML URI (14,137,350 → 9,637,218)

3. with a non-valid URI format (9,637,218 → 8,689,915)

4. with image extensions (8,689,915 → 3,875,245)

5. with a non-English TLD (3,875,245 → 2,624,527)

4.3.2 WAYBACK MACHINE ACCESS LOG DIVERSITY MATRIX

We measure the Wayback Machine access log set using WSDL diversity index (Section 4.1.1). The URI diversity score is 1.0, since we only have the unique URIs in our dataset. The hostname diversity is 0.92, and the domain diversity is 0.83. This results indicates that our sample is diverse.

4.3.3 WAYBACK MACHINE ACCESS LOG TOP LEVEL DOMAIN

In this section we wanted to determine the diversity of the TLDs in the Wayback Machine access log sample. Shown in Table 21, we found that 68.04% URIs are a commercial top level domain, `.com`, followed by `.org`. Other top level domains include `.ca`, `.it`, etc. We found 1,017 unique TLDs in the others category.

Table 21 Top level domain analysis for Wayback Machine access log sample

TLD	Num. URIs	Percent
com	1,785,827	68.04%
org	173,096	6.60%
net	152,026	5.79%
edu	24,425	0.93%
gov	10,743	0.41%
us	6,768	0.26%
others	471,642	17.97%
Total	2,624,527	100%

4.3.4 WAYBACK MACHINE ACCESS LOG URI DEPTH

Here, we want to know if the URIs we are recommending are only URIs of depth 0, similar to our analysis of the other datasets. The depth is measured after URI canonicalization using SURT. Shown in Table 22 we found that 86.83% of the URIs in the Wayback Machine access log sample are depth 0, essentially top-level webpages, and 6.21% of the URIs are depth 1. We also found that the longest depth is 16.

Table 22 Depth analysis for Wayback Machine access log sample

Depth	Count	Percent
0	2,278,764	86.83%
1	163,017	6.21%
2	99,575	3.80%
3+	79,420	3.03%
Total	2,624,527	100%

4.3.5 WAYBACK MACHINE ACCESS LOG URI PATTERNS

Here, we calculate the different URI patterns that occur in the Wayback Machine access log sample. Table 23 shows the percentage of occurrence of the pattern in the

hostname and the path. We found 40.10% of the URIs contain long strings that are not separated in the hostname and 1.08% of the URIs contain long slugs.

Table 23 URI patterns present in the Wayback Machine access log sample

Pattern	% in hostname	% in path
Long strings, not separated	40.10%	2.32%
Long strings, separated	11.21%	-
Long slugs	-	1.08%
Numbers	6.86%	6.02%
Query	-	2.19%
Port number	0%	-
IP address	0.12%	-
Percent-encoding	0%	0.22%
Date	0%	0.09%

4.4 HOW EACH DATASET IS USED

We are going to evaluate our model using a sample of requests from the Wayback Machine access log. We will use both DMOZ and Wikipedia to classify the requested URI and find recommendations, as shown in Figure 31.
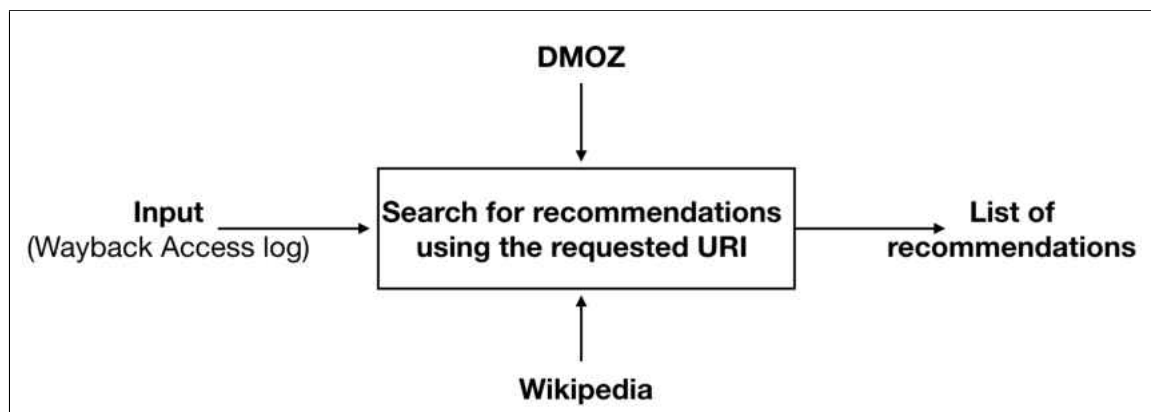
Fig. 31 A top level diagram showing how each dataset is used

4.5 COMPARING DATASETS

Here, we compare the three ontologies DMOZ, Wikipedia, and the Wayback Machine access log. Table 24 shows some of the main characteristics we are going to discuss. First, we compare the two ontologies used to classify the requested URIs.

Table 24 Comparing the three datasets

	DMOZ	Wikipedia	Wayback Machine access log
Hostname diversity	0.55	0.03	0.92
Domain diversity	0.49	0.02	0.83
.com	61.85%	37.14%	68.04%
Depth 0	50.57%	3.40%	86.83%
Depth 1	17.40%	30.21%	6.21%
Long strings in hostnames	42.65%	33.92%	40.10%
Numbers in path	20.01%	77.28%	6.02%

When comparing the hostname diversity we found that the score of hostname diversity in DMOZ dataset is 0.55, and the score of domain diversity in DMOZ dataset is 0.49. However, the score of hostname diversity in Wikipedia dataset is 0.03, and the score of domain diversity in Wikipedia dataset is 0.02. The results indicate that there are a large number of redundant hostnames and domains in the Wikipedia external links dataset, and almost half of DMOZ dataset is redundant.

Also, we found that in DMOZ 61.85% of the URIs are commercial top level domain (`.com`), however in Wikipedia we found that 37.14% URIs are commercial top level domain (`.com`).

In addition, in DMOZ dataset we found that 50.57% of the URIs are depth 0, and 17.40% are depth 1. However, in Wikipedia we found that only 3.40% of the URIs in Wikipedia's external links are depth 0, 30.21% of the URIs are depth 1, and the highest is 34.19% for URIs with depth 2. This indicates that DMOZ mostly covers the domain level, however, Wikipedia covers deeper webpages. Overall our dataset including both DMOZ and Wikipedia will be diverse, covering both domain level and deep level webpages.

For pattern analysis, we found that DMOZ has 42.65% of the URIs that contain long strings in the hostname, and 20.01% of the URIs that contain numbers in the path. However, in Wikipedia's external links we found that 33.92% of the URIs contain long strings in the path and 77.28% of the URIs contain numbers in the path. Although there is a high percent of numbers in the path, we found that 15.78% of the URI contains percent encoding and 44.39% contains query.

As for the Wayback Machine access log sample we found that the hostname diversity is 0.92, and the domain diversity is 0.83. This results indicates that our

sample is diverse. In addition, we found that 68.04% are commercial top level domain (`.com`). Also, we found that 86.83% are depth 0, meaning that most of the dataset used for analyzing the model is going to be of depth 0. In addition, we found 40.10% of the URIs contain long strings in the hostname. Also, we found that 6.02% of the URIs contain numbers in the path and 6.86% in the domain.

4.6 SUMMARY

In this chapter, we described and analyzed the datasets that we will use such as DMOZ, Wikipedia, and the Wayback Machine access log. DMOZ and Wikipedia datasets will be used as an ontology to classify incoming requests. The Wayback Machine access log will be used as input to our framework as requests for evaluation. We analyzed and compared the datasets, based on TLD, depth, and patterns.

CHAPTER 5

IMPLEMENTATION

In the previous chapter, we discussed the datasets we are going to be working with: DMOZ, Wikipedia, and the Wayback Machine access log. In this chapter, we present the framework and the implementation of our work, which includes detecting the URI aboutness by tokenizing the URI, classifying the URI, filtering candidates, and adding dimensions to select and rank recommendations. For each step, we present the different datasets we can use, the methods to accomplish the step, and the methods we use to test and evaluate the quality of the outcome.

When a user requests a webpage from the archive, that page may or may not be archived, and based on only the URI, we want the result to be a set of recommendations of similar webpages. We want to use only the URI because the content may not be available, to minimize the cost of dereferencing the webpage, and to enhance speed. The recommendation list is affected by the resources available and other features that we can add, but these may also affect the cost and time. There are dimensions we can add to enhance the recommendations outcome such as the archival quality, webpage popularity, temporal similarity, and URI similarity. In an ideal situation, we want to recommend webpages that might be in a cold spot, which are pages that are not live, are currently not popular, but are archived.
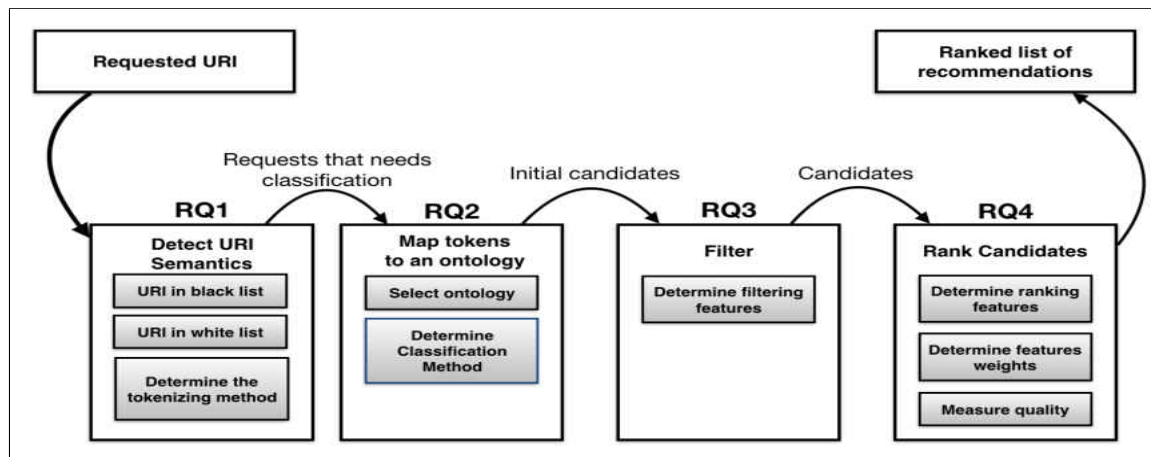
Fig. 32 Main steps performed to find recommendations for a requested URI

5.1 ESTABLISHING BASELINES

Here we describe the general method of addressing the main research question: "**Given a URI, how can we build a model for selecting and ranking possible recommendation webpages at a Web archive?**". That includes addressing other research questions, starting with a user requesting a URI to finally construct a ranked list of K archived recommendations that are similar to the requested URI. The main steps we will address are shown in Figure 32. After the user requests a URI, four main steps are performed, which are detecting the URI semantics, mapping tokens to an ontology, filtering candidates, and ranking candidates. Each of the four steps addresses a research question. Finally, the user will be provided with a ranked list of K recommendations.

The outline of the proposed steps is shown in Algorithm 1. The four main steps in our work are the following functions: first-level classification (Line 1), deep level classification (Line 4), filter candidates (Line 11), and score and rank candidates (Line 16). Each step in Algorithm 1 corresponds to one or more research questions proposed here.

- **RQ1: Given a URI, how much information can we learn about the referenced webpage without examining the content of the page?** Here, we determine the tokenizing method (Line 28). In addition, since the DMOZ dataset contains a list of categorized URIs (Section 4.1), we notice that some hostnames are categorized in multiple main categories and other hostnames appear in a single category multiple times. For this reason, we check if the requested URI is in the black list (Line 22), which is the list of hostnames that are hard to classify due to existing in multiple categories. In addition, we check if the URI is in the white list (Line 25), which is the list of hostnames that we know what main category it belongs to.

- **RQ2: Given a set of terms that describes a webpage, how do we map the terms to an ontology of webpages?** Here we select the ontology that we are going to use to classify the URI, for both first-level classification and deep level classification (Lines 1-9). The first-level classification is to determine the main category that the requested URI belongs to (1-3). Deep level classification is to determine the deep-level category that the requested URI belongs to (Lines 4-9). At the end of this section we will have a list of candidates.

- **RQ3: Given a set of webpages discovered through the ontology, what metrics do we use to filter the best candidates for recommendation?** Here we filter the candidates based on if the candidates are archived (Lines 11-14).

- **RQ4: Given a set of candidate webpages for recommendation, how do we rank the candidates?** Here we score and rank the candidates (Lines 16-18) by scoring and ranking the candidates based on several features (Line 17). Finally, we select the top N ranked recommendations (Line 18). Note that if the number of candidates is less then N, then we would collect, filter, and score more candidates (Line 31-35).
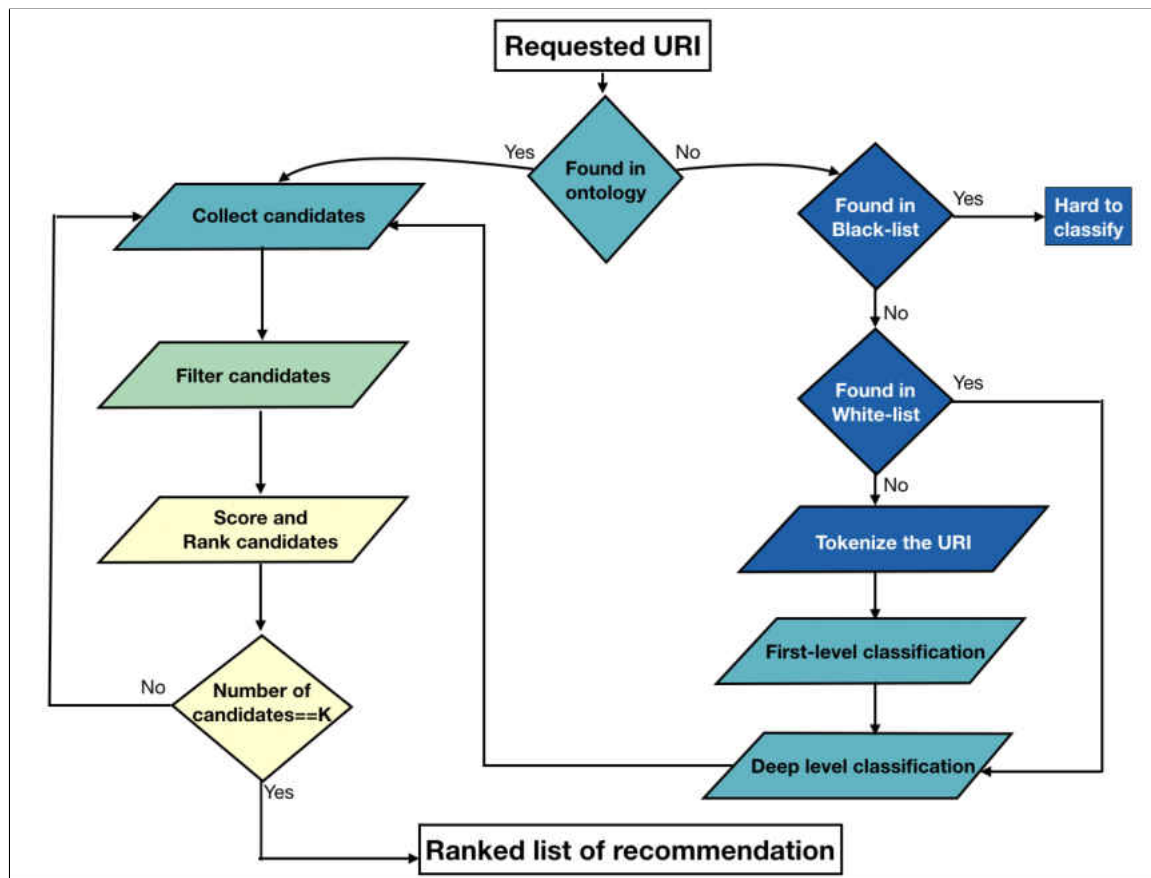
Fig. 33 Flowchart overview of the main steps to find recommendations for a requested URI, color coded by research questions, where (RQ1: dark blue, RQ2: light blue, RQ3: green, and RQ4: yellow)

Algorithm 1 Main steps to find recommendations for a requested URI

    ▷ (RQ1, RQ2) Step 1
1: **function** FIRST_LEVEL_CLASSIFICATION(*requested_URI*)
2:    ML (requested_URI)
3:

    ▷ (RQ2) Step 2
4: **function** DEEP_LEVEL_CLASSIFICATION(*requested_URI*)
5:    Index_dataset_by_category ()
6:    Cosine_similarity (*requested_URI*)
7:    Get_top_N_candidates (*Candidates*)
8:    Create_and_prune tree (*Candidates*)
9:    ML (*Candidates*)
10:

    ▷ (RQ3) Step 3
11: **function** FILTER_CANDIDATES(*Candidates*)
12:    **for** *Candidates* **do**
13:        **if** Candidate is archived **then**
14:            Archived_Candidates=Candidate
15:

    ▷ (RQ4) Step 4
16: **function** SCORE_AND_RANK_CANDIDATES(*Archived_Candidates*)
17:    Score (*Archived_Candidates*)
18:    Get_top_N_candidates (*Archived_Candidates*)
19:

    ▷ Main Function
20: **function** RECOMMENDINGARCHIVEDWEBPAGES(*requested_URI*)
21:    **if** requested_URI not in a_classified_ontology **then**
22:        **if** requested_URI in black_list **then**
23:            Print 'Hard_to_Classify'
24:        **else**
25:            **if** requested_URI in white_list **then**
26:                DEEP_LEVEL_CLASSIFICATION(*requested_URI*)   ▷ Step 2
27:            **else**
28:                TOKENIZE_URI(*requested_URI*)
29:                FIRST_LEVEL_CLASSIFICATION(*requested_URI*)   ▷ Step 1
30:                DEEP_LEVEL_CLASSIFICATION(*requested_URI*)   ▷ Step 2
31:    **repeat**
32:        COLLECT_ALL_CANDIDATES(*requested_URI*)
33:        FILTER_CANDIDATES(*Candidates*)   ▷ Step 3
34:        SCORE_AND_RANK_CANDIDATES(*Archived_Candidates*)   ▷ Step 4
35:    **until** Number of Candidates is N

Figure 33 shows a diagram overview of the framework. To accomplish the steps we need some ontologies to classify the URI. In our work, we used DMOZ and Wikipedia (Section 4.1 and 4.2). To test the entire model we used a sample of the Wayback access log (Section 4.3).

5.2 DETECT URI SEMANTICS

Here we try to address RQ1. We want to determine the semantics of the URI so we can classify the URI without dereferencing it. There are several methods to determine the semantics of the URI. In this section, we present different tokenizing methods of the URI and make some statistical analysis on the results. Next, we present some challenges with the methods. After that, we will evaluate the methods and compare the results to the related work.

In addition, we determine if the requested URI is not found in an existing ontology by checking if the requested URI is included in the black list. This list contains hostnames that belong to multiple categories in DMOZ and thus are hard to classify. If determine the requested URI is not in the black list, then we check if it is in the white list, which is a list of hostnames that belong to a single category and are easy to classify. After determining the first-level category, then we perform deep level classification.

Tokenize the URI

To classify the URI, we need to extract meaningful keywords, or tokens, from the URI. We adopt the three methods proposed by Baykan et al. [1]:

- **Tokens** The URI is split into potentially meaningful tokens. The URI is converted to lower-case and then split into tokens using any non-alphabetic character as a delimiter. Finally, the "http" (or "https") token is removed, along with any resulting token of length 2 or less. The code to tokenize using Baykan's method is shown in Listing 5, where we also show the total number of tokens. An example of using this method is shown in Listing 6. This method results in words, combined words, or characters. This method results in a large number of new tokens not seen in the training phase compared to other methods [1].

Listing 5 Creating Baykan et al. [54, 1] method to tokenize
the URI in Python

```python
def tokenize (url):
  #Step 1: lower case
  url=url.lower()
  #Step 2: Split by any non-alphabetic characters
  words=re.split("[^a-zA-Z]", url)
  #remove empty words from the list
  words =filter(None, words)
  #Step 3: Remove any tokens less than length 2, http,
      and https
  result = [item for item in words if len(item)>2 and
      item!="http" and item!="https"]
 return result


def main(argv):
  url=sys.argv[1]
  words=[]
  words=tokenize(url)
  print (words)
  print ("Number of tokens="+str(len(url)))
```

Listing 6 Baykan et al. [54, 1] tokenization example

```
>python method1_tokens.py "http://allwatchers.com/
    Topics/Info_3922.as"
["allwatchers", "com", "topics", "info"]
Number of tokens= 4
```

- **All-grams from tokens** The URI tokens are converted to all-grams. We
  perform the tokenization as above and then generate all-grams on the tokens
  by combining 4-, 5-, 6-, 7-, and 8-grams of the combined tokens. Where n-
  gram is the process of creating a sequence of n characters from the sample.
  The code to tokenize using Baykan's method to create all-grams from tokens
  is shown in Listing 7, where we also show the total number of tokens. An
  example of using this method is shown in Listing 8. This method results in
  a fewer number of new tokens not seen in the training phase compared to the
  tokenization method [1]. However, the resulting number of tokens is larger than

the tokenization method. Using the same example the number of tokens using this method is 38 compared to 4 tokens using the tokenization method.

Listing 7 Creating Baykan et al. [54, 1] method to create all-grams from tokens in Python

```
from nltk.util import ngrams
def word_grams(words, min=1, max=4):
  s = []
  if len(words)<min:
    s.append(words)
  else:
    for n in range(min, max+1):
      for ngram in ngrams(words, n):
      s.append("".join(str(i) for i in ngram))
    return s


def main(argv):
  url=sys.argv[1]
  words=[]
  words=tokenize(url)#using the tokenization method
  all_gram_feature=[]
  for r in words:
    all_gram_feature.extend(word_grams(words,4,8))
    print (all_gram_feature)
    print ("Number of tokens="+str(len(url)))
```

Listing 8 Baykan et al. [54, 1] all-grams from tokens example

```
>python method2_features.py "http://allwatchers.com/
  Topics/Info_3922.as"
["allw", "llwa", "lwat", "watc", "atch", "tche", "cher"
  , "hers", "allwa", "llwat", "lwatc", "watch", "atche
  ", "tcher", "chers", "allwat", "llwatc", "lwatch", "
  watche", "atcher", "tchers", "allwatc", "llwatch", "
  lwatche", "watcher", "atchers", "allwatch", "
  llwatche", "lwatcher", "watchers", "com", "topi", "
  opic", "pics", "topic", "opics", "topics", "info"]
Number of tokens= 38
```

- **All-grams from the URI** The URI is converted to all-grams without tokenizing first. Any punctuation and numbers are removed from the URI, along

with "http" (or "https"). Then the result is converted to lowercase. Finally, the all-grams are generated by combining the 4-, 5-, 6-, 7-, and 8-grams of the remaining URI characters. The code to tokenize using Baykan's method to create all-grams from the URI is shown in Listing 9, where we also show the total number of tokens. An example of using this method is shown in Listing 10. This method results in a fewer number of new tokens not seen in the training phase compared to the tokenization method and the all-grams from tokens [1]. However, the resulting number of tokens is larger than both the tokenization method and the all-grams from tokens. Using the same example, the number of tokens using this method is 105 compared to 4 tokens using the tokenization method and 38 using the all-gram from tokens method.

Listing 9 Creating Baykan et al. [54, 1] method to create all-grams from the URI in Python

```
def groupURI(url):
  url=url.strip("http")
  url=re.sub(r"[^a-zA-Z_]", "", url)
  return url
def main(argv):
  url=sys.argv[1]
  words=groupURI(url)
  all_gram_uri=[]
  all_gram_uri.extend(word_grams(words,4,8))#using the
      word_grams method
  print (all_gram_uri)
  print ("Number of tokens="+str(len(url)))
```

Listing 10 Baykan et al. [54, 1] all-grams from tokens example

```
    >python method3_URI.py "http://allwatchers.com/Topics/
        Info_3922.as"
    ["allw", "llwa", "lwat", "watc", "atch", "tche", "cher"
        , "hers", "ersc", "rsco", "scom", "comT", "omTo", "
        mTop", "Topi", "opic", "pics", "icsI", "csIn", "sInf
        ", "Info", "nfoa", "foas", "allwa", "llwat", "lwatc"
        , "watch", "atche", "tcher", "chers", "hersc", "
        ersco", "rscom", "scomT", "comTo", "omTop", "mTopi",
         "Topic", "opics", "picsI", "icsIn", "csInf", "sInfo
        ", "Infoa", "nfoas", "allwat", "llwatc", "lwatch", "
        watche", "atcher", "tchers", "chersc", "hersco", "
        erscom", "rscomT", "scomTo", "comTop", "omTopi", "
        mTopic", "Topics", "opicsI", "picsIn", "icsInf", "
        csInfo", "sInfoa", "Infoas", "allwatc", "llwatch", "
        lwatche", "watcher", "atchers", "tchersc", "chersco"
        , "herscom", "erscomT", "rscomTo", "scomTop", "
        comTopi", "omTopic", "mTopics", "TopicsI", "opicsIn"
        , "picsInf", "icsInfo", "csInfoa", "sInfoas", "
        allwatch", "llwatche", "lwatcher", "watchers", "
        atchersc", "tchersco", "cherscom", "herscomT", "
        erscomTo", "rscomTop", "scomTopi", "comTopic", "
        omTopics", "mTopicsI", "TopicsIn", "opicsInf", "
        picsInfo", "icsInfoa", "csInfoas"]
    Number of tokens=105
```

Note that the all-grams created from the tokens and from the URI contains grams that are not words, however, we do not filter them out because keeping them will increase finding a match using machine learning.

To show the effect of different tokenization methods on the number of resulting tokens, Table 25 shows the result of tokenizing `http://odu.edu/compsci`. In addition to testing the tokenization methods we also examine removing the TLDs from the URIs, removing numbers, and removing stop-words (Section 5.2.1).

Measuring Term Frequency

After splitting the dataset, tokens that appear in the test sample and not in the training sample may result in wrongfully classified URIs. Here we want to measure the amount of these "empty URIs".

Table 25 Tokenizing the URI `https://odu.edu/compsci` using different methods [1]

Method	Result
Tokens	odu, edu, compsci
All-grams from tokens	odu, edu, comp, omps, mpsc, psci, comps, ompsc, mpsci, compsc, ompsci, compsci
All-grams from URI	odue, dued, uedu, educ, duco, ucom, comp, omps, mpsc, psci, odued, duedu, ueduc, educo, ducom, ucomp, comps, ompsc, mpsci, oduedu, dueduc, ueduco, educom, ducomp, ucomps, compsc, ompsci, odueduc, dueduco, ueducom, educomp, ducomps, ucompsc, compsci, odueduco, dueducom, ueducomp, educomps, ducompsc, ucompsci

In Baykan et al. [1], the tokenization method has a low F_1 due to the high number of empty URLs (tokens not seen before) (32%). Examples of empty URIs are `http://xboxgames.com` and `http://yuzutree.com`. If the tokens 'xboxgames' and 'yuzutree' did not appear as tokens in any other URI, then they are considered as empty URIs. Here we check term frequency of the results of the different tokenization methods in order to estimate the issue and to recommend a solution, such as the usage of Wikipedia dataset as an additional ontology to cover as many URIs as possible.

- Tokenization: Using the DMOZ dataset (Section 4.1) with a total of 6,522,125 URIs, we converted the URIs to tokens (Listing 5). The number of tokens created for all the URI is 24,559,916, and the number of unique tokens created is 3,481,714. We checked the number of tokens that only appeared once, as this is important when classifying the URI. The goal is to have more frequent tokens so that they appear in both the training and testing dataset. If they appear only in the testing, then they can be classified. In this case we found 2,727,504 tokens that appeared only once, which is 78.34% of the unique tokens. However, we want to check if the URI only contains tokens that appear once. Since we included the top level domain in this method, there were no resulting URIs that contained only tokens which appeared only once. We found that none of the unique URIs are empty URIs because the TLD was included and redundant in the dataset.

- Tokenization, while removing the TLD and the URI extension: Using the DMOZ dataset, we filtered out the TLD and the URI extensions and tokenize the URIs. The number of tokens created for all the URI is 16,456,706,

and the number of unique tokens created is 3,468,686. We checked the number of tokens that appeared only once and found 2,725,413 tokens that appeared only once, which is 78.57% of the unique tokens and 16.56% of all tokens. We found that 40.90% of the unique URIs are empty URIs.

- All-grams from tokens, while removing the TLD and the URI extension: Using the DMOZ dataset, the number of all-gram from tokens created for all the URI is 258,171,414, and the number of unique tokens created is 18,988,783. We checked the number of tokens that appeared only once. In this case we found 10,777,371 tokens that appeared only once, which is 56.76% of the unique tokens and 4.17% of all tokens. We found that 0.2% of the unique URIs are empty URIs.

- All-grams from URI, while removing the TLD and the extension: Using the DMOZ dataset, the number of all-gram from URI created for all the URI is 620,919,919, and the number of unique tokens created is 52,348,990. We checked the number of tokens that appeared only once. In this case we found 32,873,677 tokens that appeared only once, which is 62.80% of the unique tokens and 5.29% of all tokens. We found that 0.1% of the unique URIs are empty URIs.

Resolve the Empty URIs using Wikipedia

Using a sample of the empty URIs using tokenization while removing the TLD and the URI extension method, we want to determine if Wikipedia can suggest related articles. We used the tokens and entered it in the Wikipedia search bar, then if there is a match Wikipedia will suggest related articles. By going through those suggestions we search for an article that contains the empty URI as an "official webpage" (Section 4.2). Finally, we will use the article's category as the category we are going to use for the requested URI. For example, the following URIs below are empty URIs and we use Wikipedia to find related webpages:

- `http://africover.org` Wikipedia suggested the following related webpages: Africover, Environmental issues in Africa.

- `http://00-buckshot.com/` Wikipedia suggested the following related webpages: Shotgun shell, Buckshot (rapper), Buckshot Jones, Taurus Judge, Buckshot Roberts, .410 bore, Black Moon (group), Buckshot LeFonque, The Buck Shot Show, Boot Camp Clik

- `http://1962valiants.com` Wikipedia suggested the following related webpages: Valiant, The Valiants, Vince Vance & the Valiants, Valiants Memorial, Vickers Valiant, Valiant tank, Johnny Valiant, The Valiants (firefighters), Plymouth Valiant, Chrysler Valiant

However, in some cases Wikipedia does not find any related webpages. For examples, the following URIs are cases where Wikipedia could not suggest any webpages:

- `http://2sweetkidz.com`

- `http://cableseal.com`

- `http://byzart.com`

Using this method, we found Wikipedia articles for 1,867,284 the empty URIs, which is almost 70%.

5.2.1 TESTING AND EVALUATING

To determine the best tokenization method, we classify the resulting tokens based on an ontology and check its accuracy. One ontology dataset we can use for classifying the URI is DMOZ. Here we use Naïve Bayes, and train the algorithm based on part of the dataset and test it on the remaining part. We chose NB because of its simplicity and nonindependent and overlapping features. In addition, NB performance in tokenization was proven to have close results to other methods such as ME and SVM [1]. We use F_1 to determine the best classification method. F_1 is the harmonic mean of the precision and recall [148, 149] (Equation 5). Precision is the positive predictive value (Equation 3). Recall is the sensitivity, which is calculated by determining the relevant instances retrieved and the total relevant instances (Equation 4).

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F_1 = \frac{2(Recall \times Precision)}{(Recall + Precision)} \tag{5}$$

where

- True Positives (TP) are when the correct category is predicted

- True Negatives (TN) are when the URI is correctly predicted as not belonging to a category

- False Positives (FP) are when the URI does not belong to a category but is predicted as belonging to the category

- False Negatives (FN) are when the URI belongs to a category but is predicted as not belonging to category

We must consider that when we are doing the classification step, we compare our result to Baykan et al. [1] (Section 3.2.2).

Best tokenization method

To determine the best tokenization method, we tested the classification of tokens on the DMOZ dataset, using machine learning. We took the DMOZ dataset and created a 10-fold cross-validation set, using 90% for training and 10% for testing. We employed a Naïve Bayes classifier to take tokens and return the top-level category. Naïve Bayes was selected because of its simplicity that assumes independence between the features. In the testing dataset we filtered out URIs that contain tokens not seen in the training set, as was also done in related work [1]. In the DMOZ dataset we took the URIs and their top-level category. Then for testing, we use the tokens from the test input to determine the likely category. We measured the F_1 score to evaluate the different tokenization methods. Table 26 shows the result of our evaluation. In addition to the base tokenization methods described above, we also tested the following alternatives for each method:

- remove TLD before tokenization

- remove TLD and numbers before tokenization

- remove TLD, numbers, and stop-words before tokenization

The stop-words were based on a set of stop-words in the Natural Language Toolkit (NLTK) [150, 151, 152]. We found that using the all-grams from the URI after removing the TLD and numbers had the highest F_1 score, which was comparable to results obtained in related work [95]. We use this method of tokenization going forward.

Table 26 Classifying at the first level, comparing F_1 score, Micro average, and Macro average for DMOZ dataset using different methods

Method		F_1 score	Micro average	Macro average
Tokens	**All URI tokens**	0.39	0.45	0.31
	URI tokens, without TLD	0.35	0.40	0.28
	URI tokens, without TLD and numbers	0.40	0.45	0.32
	URI tokens, without TLD and stop-words	0.39	0.43	0.30
All-gram from tokens	**All URI tokens**	0.51	0.53	0.45
	URI tokens, without TLD	0.51	0.53	0.46
	URI tokens, without TLD and numbers	0.51	0.52	0.47
	URI tokens, without TLD and stop-words	0.50	0.52	0.46
All-grams from URI	**All URI tokens**	0.56	0.55	0.48
	URI tokens, without TLD	0.55	0.59	0.46
	URI tokens, without TLD and numbers	**0.59**	**0.62**	**0.61**
	URI tokens, without TLD and stop-words	0.55	0.60	0.47

5.2.2 HOSTNAME BLACK LIST

In the DMOZ dataset, we found that are some hostnames that are so general that they are in many categories. This may cause the prediction of the category to be wrong. For this reason we determined the number of hostnames that are in many categories and are hard to classify. In order to create a list of hostnames that are hard to classify, we first extract the hostnames from the DMOZ URI list, then for each hostname we determine the list of main categories that it belongs to. Table 27 and Figure 34 shows the number of categories that a hostname appeared in and the number of hostnames. We found that there are a total of 3,586,362 unique hostnames in DMOZ. Of these, 3,254,322 appeared only once in one category, and 184,627 appeared more than once in one category. For example, the hostnames `geoc ities.com` and `members.aol.com` appeared in all 17 categories in DMOZ. However, the hostname `accuweather.com` appeared multiple times in the category Regional.

Table 27 The number of categories that a hostname appeared in and the number of hostnames

# of categories that the hostname appeared in	# of hostnames
1	184,627 (appeared more than once) and 3,254,322 (appeared once)
2	108,725
3	20,275
4	7,513
5	3,725
6	2,224
7	1,522
8	953
9	672
10	477
11	380
12	312
13	234
14	162
15	129
16	96
17	6
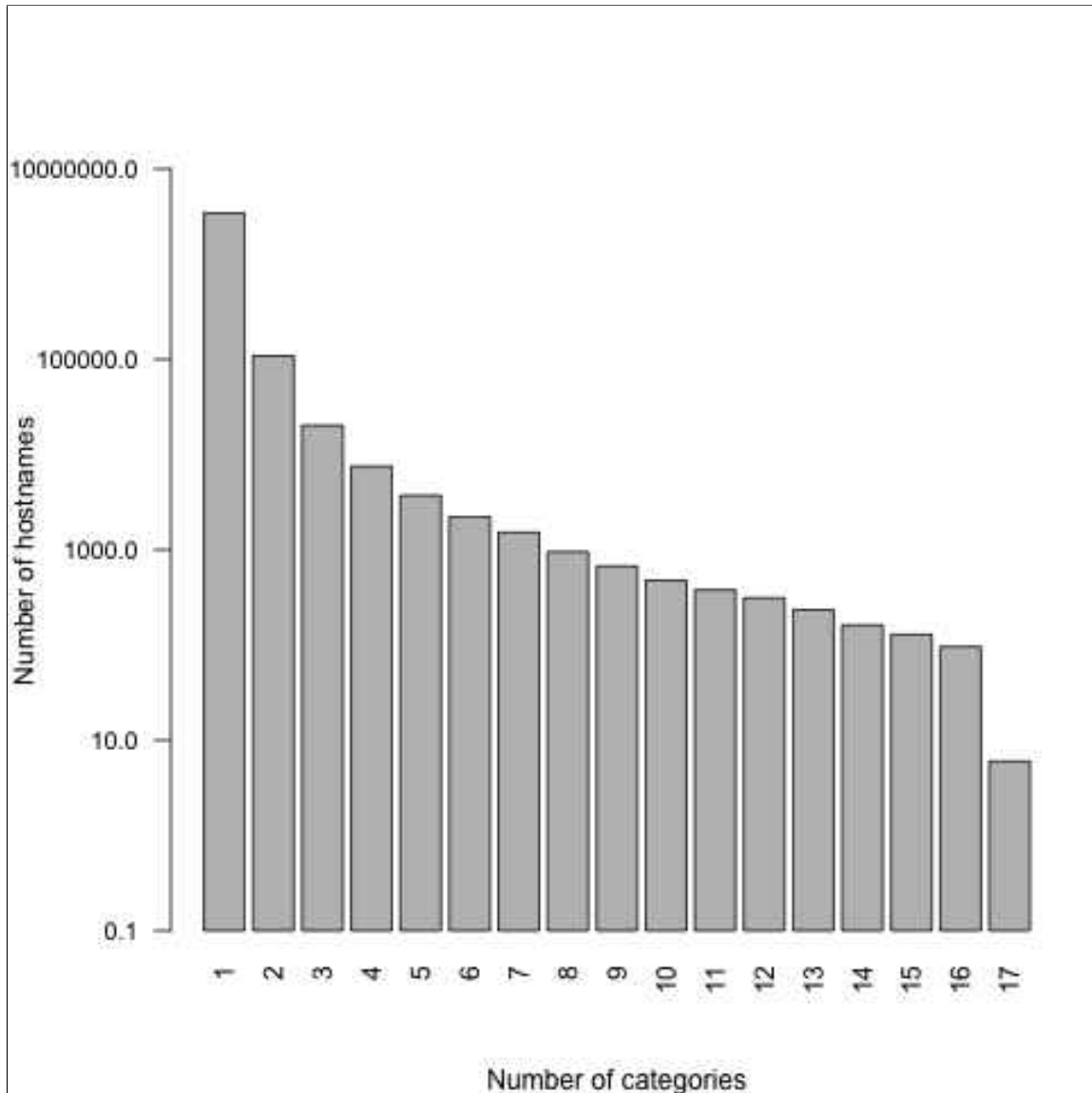
Fig. 34 The number of categories that a hostname appeared in and the number of hostnames

In order to create the black list we need to analyze this information further. In addition to a hostname appearing in multiple categories, we notice that the distribution of its occurrence within the category is different. For example, the hostname `cnn.com` appeared in 11 categories, however 99.87% of the occurrences were in the News category. So `cnn.com` is not hard to classify. This makes us look into the percentage of occurrences within the categories. Thus to create a black list we have to consider two factors: the number of categories that a hostname belongs to and the distribution of occurrences in each category. To find hostnames that are in the black list in DMOZ, we searched for hostnames that appeared in more than 10 categories,

and the difference between the percentage of occurrences for the top 3 categories is less than 15%. The total number of hostnames in this case is 954. If we encounter a requested URI with a hostname on the black list, our algorithm will stop and will not categorize or offer recommendations. The black list dataset is available for download at `https://github.com/oduwsdl/Recommending-Archived-Webpages`.

5.2.3 HOSTNAME WHITE LIST

Here we create a list of hostnames that are easy to classify. We have found that some hostnames exist multiple times in a single category. So instead of classifying the URI we predicted the first level classification for those requests. To create the white list we determine the hostnames that appear over 10 times in only a single category. In this case, we found 9,146 hostnames. In Table 28, we show the number of hostnames occurrences in each category. We found that 3,081 hostnames are from the Regional category and 2,825 hostnames are from the Adult category. We found 311 hostnames that appeared over 100 times in a single category. Examples of hostnames that are in the white list include: `movies.tvguide.com` categorized as Arts, `weatherunderground.com` categorized as Regional, and `cfpeople.org` categorized as Society. If we encounter a requested URI with a hostname on the white list, we will consider it classified at the first level and move on to the deep level classification step. The white list dataset is available for download at `https://github.com/oduwsdl/Recommending-Archived-Webpages`.

5.3 MAP TOKENS TO AN ONTOLOGY

In this section, we want to address RQ2. We would need to select an ontology to work with that satisfy our requirements. Next, we want to determine which classification method to use. Finally, we would test and evaluate our methods.

5.3.1 SELECT ONTOLOGIES

We have some different datasets that we may consider, such as DMOZ, Wikipedia, and search engine, such as Google or Yahoo! Directory. Note that search engines directories may update their lists over time by removing webpages that are no longer available, hence it will be impossible to find webpages that archived but not in the live web. In our work, we prefer DMOZ since it may contain webpages that no longer

Table 28 The number of hostnames occurrences in each category

Category	Number of hostnames occurrences
Regional	3,081
Adult	2,825
Arts	890
Sports	364
Games	355
Computers	355
Society	282
Science	233
Home	196
Health	167
Business	117
Reference	103
Recreation	94
Kids and Teens	70
News	7
Shopping	6
Netscape	1

exist on the live Web and since we have collected its dataset locally over the span of 17 years. In addition, external links in Wikipedia are considered classified and may contain webpages not found in DMOZ and could help classify webpages that DMOZ could not.

In our work, the first step is to determine if the requested URI is already present and categorized in a known ontology, and we use DMOZ and Wikipedia. Using DMOZ is straight forward; we check if the URI exists in DMOZ or not by matching the requested URI to the classified URIs in DMOZ. However, in Wikipedia we check if the requested URI is the official webpage and is categorized (Section 4.2).

To test how often this option might be available, we used the Wayback Machine access logs (Section 4.3). From the filtered set, we found 13.17% URIs in DMOZ or Wikipedia.

5.3.2 DETERMINE THE CLASSIFICATION METHODS

We need to determine the method to classify the webpage using the tokens we

extracted from the previous step. In related work, common machine learning algorithms, such as Naïve Bayes and Maximum Entropy work well. This method works for determining the first level classification. However, since we have a large dataset and a hierarchical structure on the data, we adopted the method proposed by Xue et al. [6] (Section 3.2.2) for deep classification. In summary, deep classification has two steps. First, we compute the term frequency-inverse document frequency (TF-IDF) [153] for each input, and then check the similarity using cosine similarity. Alternatively, we can use the TF-IDF by grouping the instances by category, then check the similarity using cosine similarity. Second, we select K candidates from the first step and perform NB to select the best candidate category. Note that this work was performed on documents, not URIs. In our case, we performed it on URI tokens.

First-Level Classification

For a request that did not appear in an ontology, we classify it using only the tokens from the URI. Out of the three different methods of tokenization (tokenize, all-grams from the tokens, and all-grams from the URI), Table 26 showed that the best tokenization method was using all-grams from URI while removing TLD and numbers.

Now that we have determined the best tokenization method, we will apply this for future requests. We trained the Naïve Bayes classifier on the entire DMOZ dataset and this will be used for classification at the first level. We take the requested URI, remove the TLD and numbers, and then perform the all-gram from URI tokenization described in the previous section. These resulting all-grams are used in the classifier to produce a first level classification.

Deep Level Classification

In this step we want to classify the requested URI `http://cs.odu.edu/compsci` to a hierarchical deep classification such as `Computers/Computer_Science/Academic_Departments/North_America_United_States/Virginia`. Known methods to determine hierarchical deep classification are the big-bang approach and the top-down approach [154]. Neither method is ideal with a large number of hierarchies and may result in error propagation. For this reason we adopt the method by Xue et al. [6]. In our case we are limited to the URI only and do not have the documents or any supporting details. The steps to perform deep level classification are the following:

1. **Index dataset**. In preparation to compute similarity between the requested URI and the category entries, we index DMOZ by category, creating a list of all URIs in each of the DMOZ deep level categories.

2. **Cosine similarity**. We compute the cosine similarity between the tokenized requested URI and the tokenized DMOZ URIs and their titles and description in each category. In this step each category of the index will get a similarity score to the requested URI, which is the average similarity to all entries in that category.

3. **Collect N candidates**. Next we select the top 10 candidate categories with the highest similarity score, similar to related work [6].

4. **Prune tree**. Each candidate category could be a leaf node or an internal node. We create a hierarchical tree and then prune it to get the final list of candidates that we can use machine learning to classify. First, we create a tree from the candidates by starting from the first node and then going down until all 10 candidates are presented, as shown in Figure 35(a). Next, in order to enhance the classification, the tree is pruned based on the ancestor assistance strategy. The ancestor assistance strategy includes the ancestors of a node if there are no common ancestors with another candidate, as shown in Figure 35(b).

5. **Classify**. To choose a single classification from the pruned tree we classify the requested URI based on two methods, using 3-gram tokens and all-grams. The 3-gram method had the best result when comparing documents [6], however in our work we compare URI tokens, so we expect the all-gram method to perform better.

Testing classifying deep classification

To test and evaluate deep classification we will measure the performance of each level of the hierarchical category. We also want to test the effect of pattern occurrences and classification. In addition, we want to determine the characteristics of URIs that are correctly classified, such as TLD, depth, and containing words that occur in a dictionary. This will be discussed in Section 6.1.

(a) Create hierarchical tree from the 10 candidate categories (the candidate categories are highlighted). The numbers represent the category ID

(b) Pruned tree using ancestor assistance strategy. The parents of nodes 88 and 100 are included because they have no shared ancestor with other candidates
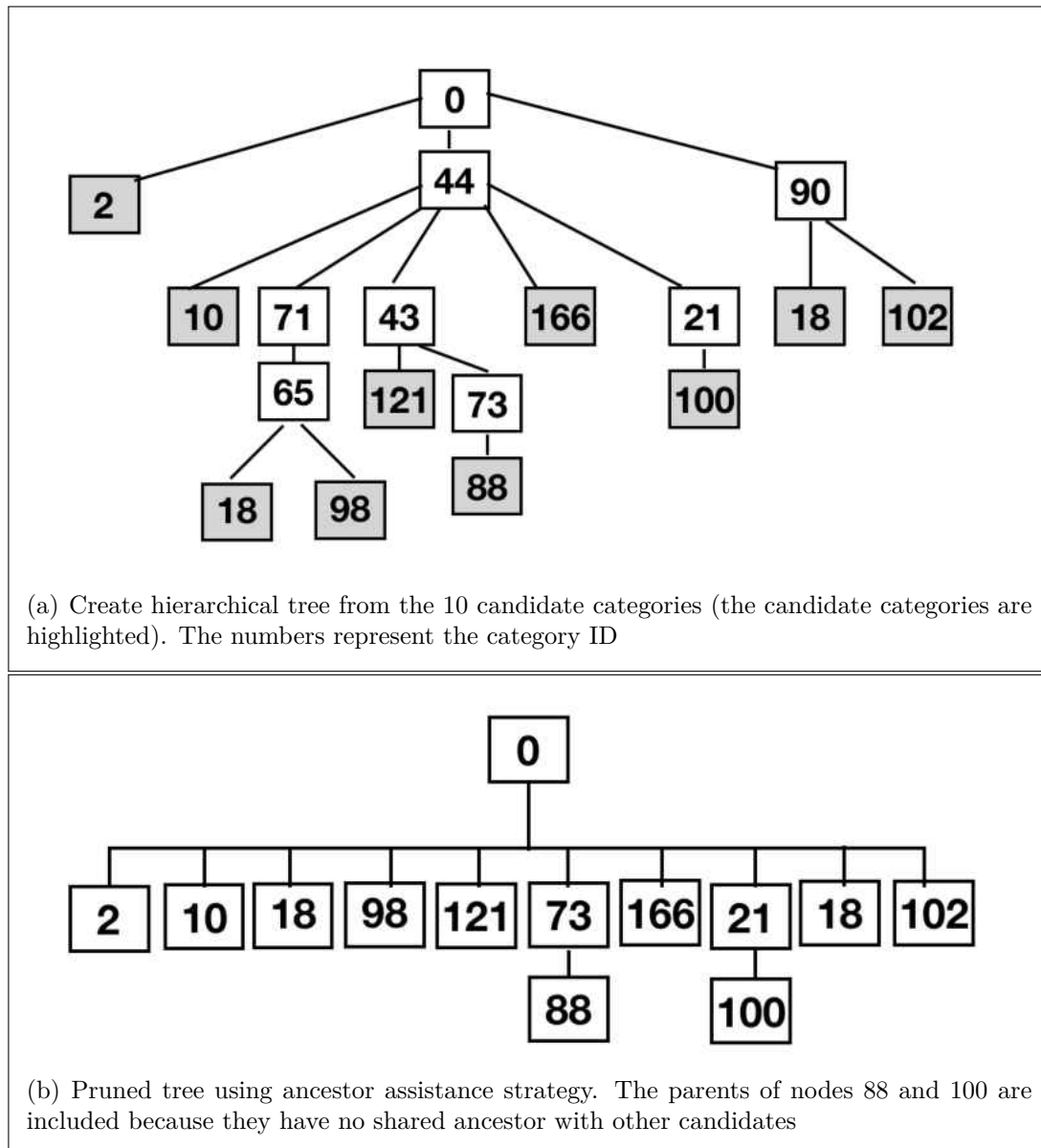
Fig. 35 The process of pruning a hierarchical tree using ancestor assistance strategy [6]

5.4 GATHERING CANDIDATES

After performing deep classification the result will be the category that the URIs belongs to. Based on the result, we gather all the URIs that belongs to this category. Those URIs are the candidates for the requested URI. In the case that the number of URIs is less than K we gather the candidates of the parent category. For example, if

we set $K = 3$, and we categorized a URI to the deep category `Computers/Algorith ms/Complexity/Conferences`, we find that there is only one URI in this category, `ht tp://cs.utep.edu/longpre/complexity.html`. In order to collect more candidates we gather the URIs from the parent category, which is in this case `Computers/ Algorithms/Complexity`. The number of URIs is 5:

- `http://haegar.informatik.uni-wuerzburg.de/person/mitarbeiter/voll mer/tcs-bookmarks.html`

- `http://link.springer-ny.com/link/service/journals/00037/`

- `http://top.cis.syr.edu/people/royer/talks.html`

- `http://claymath.org/prize_problems/p_vs_np.htm`

- `http://cs.washington.edu/homes/kautz/blackbox/`

In this example, our candidates will be these five URIs and in addition to the previous single URI.

5.5 FILTER CANDIDATES

In this section, we address RQ3. We want to determine the filtering features that we need to eliminate from the candidate results, so we do not consider them in later analysis. Also, we will test and evaluate our method. We take the candidates from Step 2 (Section 5.4) and remove any that are not archived. We use TimeMaps from MemGator [155] to determine this.

In addition, when selecting a memento of an archived candidate, we have to consider some cases, such as:

- Archived candidate URI is not found (404). For example, when selecting the memento for `http://web.archive.org/web/20110909111700/coatscrafts. co.uk/crafts/crochet/projects/crochet+project+-+crochet+bag.htm` it returns a 404 response. In this case, we filter out the URIs that are 404s and find other mementos that resolve to 200.

- Archived candidate URIs redirect to other URIs. For example, when selecting the memento for `http://norfolkline.com/ferry` it redirects to `https://we b.archive.org/web/20170613205542/https://dfdsseaways.co.uk`. This is

a challenge that we have to consider and decide if we want to re-evaluate the candidate once it is redirected. This could mean that the URI hostname has been updated or the topic of the URI has shifted. In this case, we score both the redirected memento and find an other memento that resolve to a 200 and score that memento as well.

5.6 RANK CANDIDATES

The final research question we address is RQ4. In this section, we want to determine the features we want to consider to rank the features, set the weights for each feature, apply the weights on the candidates, rank the candidates, and finally test and evaluate the results.

5.6.1 SET FEATURES

In this work, we use different features to rank the candidates. We rank and recommend the remaining candidates based on temporal similarity (t), webpage popularity (p), URI similarity (s), and archival quality (q). Our final list of recommended webpages will be ranked based on Equation 6, where $w_t + w_p + w_s + w_q = 1.0$ and specify the weights given to each of the features.

$$score = w_t t + w_p p + w_s s + w_q q \qquad (6)$$

Temporal similarity

Temporal similarity refers to how close the available candidate webpage's Memento-Datetime is to the requested URI (Section 3.5.3). If the difference between the two Memento-Datetimes are large then it is not a good candidate. This means we have to make sure that the recommended webpage is in the same time frame as the requested time frame. For example, if the user requests a "Super Bowl" in year 2017 we would not recommend results of "Super Bowl" in year 2000. This is shown in Equation 7, where r_d is the request datetime, c_d is the candidate datetime, u_d is the current datetime, and e_d is the earliest datetime. The earliest datetime is considered 1996, because that is when archiving the web started [114].

$$t = \frac{|r_d - c_d|}{u_d - e_d} \qquad (7)$$

Webpage popularity

We use how often the webpage has been archived and the domain's popularity as determined by Alexa[1] as an approximation for the webpage's popularity (Section 3.5.2). Our popularity measure p is given in Equation 8, where a is the Alexa Global Ranking of the requested domain, x is the lowest ranked domain in Alexa, n is the number of times the URI has been archived, and m is the number of times Alexa's top-ranked website has been archived.

$$p = \frac{(|\frac{loga}{logx} - 1| + \frac{\log n}{\log m})}{2} \tag{8}$$

We set x to 30,000,000 as it is the current lowest ranking in Alexa, and we set m to 538,300, the number of times that `http://google.com`, the top-ranked Alexa webpage, has been archived, as of November 2018.

URI similarity

We measure the similarity of requested URI tokens and candidate URI tokens using Jaccard similarity coefficient (Equation 9).

$$s = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{9}$$

Archival quality

Archival quality refers to how well the page is archived. It measures the damage rate of the webpages (Section 3.5.1), meaning missing resources such as images, videos, text, or any other resource. We use Memento-Damage [134] to calculate the impact of missing resources in the webpage. We calculate archival quality in Equation 10, where d is the damage score calculated from Memento-Damage.

$$q = 1 - d \tag{10}$$

5.6.2 SET FEATURES WEIGHTS

Here we want to set weights for each feature. The weights could be preset and

---

[1]`https://alexa.com`

could be altered by the user if the user wants to focus on one feature more than the other. For example, we might assign the weights to the following features in descending order: webpage popularity and in the cold spot, archival quality, content similarity, and temporal similarity. Then the user could alter the weights of these features based on his needs.

Apply weights to candidates

Using the feature weights, we apply them to the candidate webpages. Then each candidate webpage will have a score associated with it. Using these score, we can rank the candidates in descending order. Then we will present the user with K candidates. If there are less than K candidates, then we collect more candidates by gathering the URIs in the parent category, then score and rank the candidates (Section 5.4).

5.6.3 EXAMPLE

Here we present an example of a request and the resulting recommendations. We request `http://odu.edu/compsci` with the date of March 1, 2014. This URI is not classified in DMOZ or in Wikipedia, so we use machine learning and classify it to `Computers/Computer_Science/Academic_Departments/North_America/` `United_States/Virginia`. Then we collect all the candidates from DMOZ:

- `http://cs.gmu.edu`

- `http://cs.odu.edu`

- `http://cs.virginia.edu`

- `http://cs.vt.edu`

- `http://wm.edu/as/computerscience/?svr=web`

- `http://radford.edu/content/csat/home/itec.html`

- `http://cs.jmu.edu`

- `https://php.radford.edu/~itec`

- `http://mathcs.richmond.edu`

- `http://hollins.edu/academics/computersci`

Using equal weights for our ranking equation, the top three ranked candidates are:

1. `https://web.archive.org/web/20140226090846/http://cs.odu.edu:80/`, score $= 0.87$

2. `https://web.archive.org/web/20140208043915/http://cs.virginia.edu/`, score $= 0.75$

3. `https://web.archive.org/web/20140223213510/http://cs.jmu.edu/`, score $= 0.73$

5.6.4 TESTING AND EVALUATING

For testing and evaluating the entire model we will use a sample of requests from the Wayback access log and try to categorize and recommend similar webpages. We will evaluate our categorization and the ranking of the recommendations using human evaluation. This will be discussed in Section 6.2.

5.7 SUMMARY

In this chapter, we presented the baseline of our work and showed the main steps starting from requesting a URI to recommending a list of recommendations. We reviewed each research question and presented a detailed overview of how to answer them. The main steps include detecting URI semantics, mapping tokens to an ontology, filtering candidates, and ranking candidates. We described each step and detected some solutions to challenges that may occur. Finally, for each step we determined the methods of testing and evaluating.

CHAPTER 6

EVALUATION AND RESULTS

In this chapter, we present the evaluation of the deep level classification and the effect of different features in the URI such as depth, presence of dictionary words, long strings, and the category that it belongs to. We also evaluate the entire model and present human evaluators with three different datasets: requests that are found in an ontology, requests that are not found in an ontology and have depth zero, and requests that are not found in an ontology and have a path (i.e., depth is higher than 1). We show the overall scores and the results of each dataset.

6.1 TESTING DEEP LEVEL CLASSIFICATION

We evaluate how well our deep classification method (Section 5.3.2) works. To test this step we use 10% of the DMOZ dataset for testing and the rest for training. We assume that level one categorization is already predicted in Step 1. We evaluate the performance by determining if we classified each level correctly. For example, if a URI is in the category `c1/c2/c3`, then for level two evaluation, we check if we predicted `c1/c2`. For each level we calculate the Micro-average F_1 (Mi-F_1) score. In Figure 36, we show the Mi-F_1 score of each level using 3-gram cosine similarity. Compared to the related work [6], we found that our scores are low. The highest level in our results was 0.2 compared to 0.8 in Xue et al. [6], but that is due to using only the requested URI as the testing data and the URI and title and category as training, where Xue et al. had the entire document available for categorization. This shows that using only the tokens from the URI is not enough for deep classification. Because of limited information, we also show the result of testing the same method using all-gram cosine similarity. We found that the results are better with the highest score of 0.3, however this is still considered low compared to related work. This shows that deep level classification is a difficult problem with only the URI tokens available.
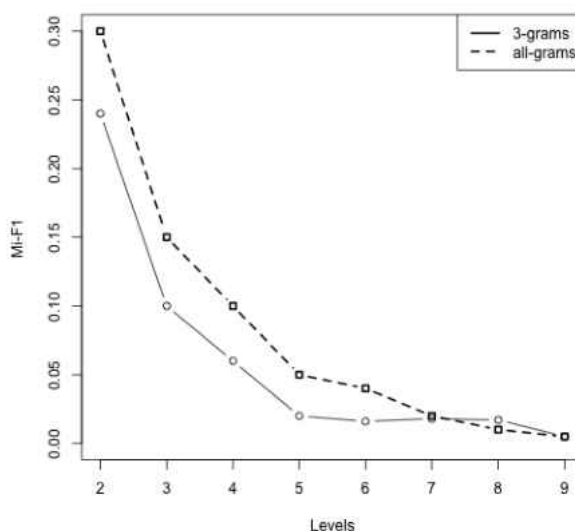
Fig. 36 Performance on classifying to different levels using 3-gram and all-gram

Features affecting Deep Level Classification Performance

Some features could affect the URI classification, such as the depth of the URI, URI words appearing in a dictionary, URI containing long strings, and the category that the URI belongs to. In this section, we will discuss each case and how it affects the classification.

Depth of the URI and classification

We investigated the relationship between the depth of the URI and classification. Table 29 shows the URI depth and the percentage of the correctly classified URIs. We only considered URIs to be correctly classified if they were correct to the deepest level. We found that 63.45% of the correctly classified URIs are of depth 0. Next, we want to determine the depth of the requested URIs. Shown in Table 22 we found that 86.83% of the URIs in the Wayback Machine access log are depth 0, essentially top-level webpages. It means that users often request URIs of depth 0 from the archive. Since 63.45% of the correctly classified URIs are of depth 0, having 86.83% at depth 0 could enhance the classification results. However, this is greater than the percentage of DMOZ URIs with depth 0 (50.57%).

Table 29 URI depth and percentage of correctly classified URIs

Depth	Percent
0	63.45%
1	16.96%
2	13.48%
3	3.77%
4	1.47%
5+	0.86%

Dictionary words in the URI and classification

We check if the words in the URIs are in a dictionary (after removing the TLD). We use the Enchant English dictionary[1] and wordninja[2] to split compound words. For example, the URI `http://mickeymantlebaseballcards.net` is split to (mickey, mantle, baseball, and cards). We found that 36.92% of the correctly classified URIs contain only words from a dictionary, and 44.89% of the correctly classified URIs contain at least one word from a dictionary.

We saw that having terms found in a dictionary affects classification. We found that 5.39% of the Wayback Machine access log URIs contain only words from a dictionary, and 26.74% contain at least one word from a dictionary. These percentages are low and affect the ability for the requested URI to be correctly classified.

Long strings in the URI and classification

An ideal structure of the URI for classification is that it contains long strings that may potentially have more semantics. We are trying to identify URIs with a "slug", which is the part of a URI that contains keywords or the webpage title. An example of a slug is the path in `https://cnn.com/2017/07/31/health/climate-change-two-degrees-studies/index.html`. The slug in the URI is readable, and we can identify what the webpage is about. We evaluate the existence of long strings in the correctly classified URIs. We assume that the average length of an English word is five letters [143, 144] and anything greater is considered a long string.

Overall, we found that 41.58% of the sampled URIs contain long strings in the domain, for example, `http://timesonline.co.uk/tol/sport/cricket/`. Also, we

---

[1]`https://pypi.org/project/pyenchant/`
[2]`https://pypi.org/project/wordninja/`

found that 89.47% of the sampled URIs contain long strings in the path, for example, `http://medlineplus.nlm.nih.gov/medlineplus/parkinsonsdisease.html`. When analyzing the correctly classified URIs, we found that 50.07% of the correctly classified URIs contain long strings in the domain. Also, we found that 13.45% of the correctly classified URIs contain long strings in the path. Words can be separated by delimiters in the domain or path. We found that 9.91% of the correctly classified URIs contain words separated by delimiters in the domain, for example, `http://vintage-poster-art.com/`. We also found that 6.97% of the correctly classified URIs contain separated words by delimiters in the path, for example, `http://seaworldparks.com/en/buschgardens-williamsburg/`.

In our DMOZ evaluation, we found that long strings in the domain helped with classification. When analyzing the Wayback Machine access logs request patterns in Table 23, we found that 40.10% contain long strings in the domain. We also found that only 2.32% contain long strings in the path. In addition, we found that 11.21% contain words separated by delimiters in the domain and only 1.08% contain slugs in the path. This also reflects the large percentage of URIs from the access logs with depth 0 (no path). For classifying most of these requests, we will have to largely rely on domain information.

Table 30 Percentage from the correctly classified URIs for each category

Category	count	Percent
Society	459	15.32%
Arts	401	13.38%
Shopping	355	11.85%
Recreation	331	11.05%
Sports	291	9.71%
Home	288	9.61%
Reference	238	7.94%
Computers	228	7.61%
Health	190	6.34%
Science	130	4.34%
Games	50	1.67%
Business	35	1.17%
News	0	0%
Total	2,996	100%

Category and classification

Here we wanted to investigate the effect of the category on correct classification. As shown in Table 30, we found that 15.32% of the correctly classified URIs were in the Society first level category. We also found that none of the correctly classified URIs were in News. We found that in the News category in DMOZ, there is a level two subcategory Online_Archive that contains 95% of the News URIs and repeats several subcategories inside News. This caused errors in our classification.

6.2 TESTING AND EVALUATING THE MODEL

For evaluating the entire model we use a sample of the Wayback Machine access log requests (described in Section 4.3). We classify the requests, gather candidates, filter candidates, score candidates, rank candidates, and recommend the top three scored candidates. We tested our overall method on a sample of 75 requests, as shown in Table 31. From requests that had exact matches in DMOZ or Wikipedia, which we call "direct match", 25 requests were chosen randomly. From requests that are not found in an ontology and are of depth zero, which we call "depth zero", 25 requests were chosen randomly. From requests that are not found in an ontology and are of depth greater than zero, which we call "deep", 25 requests were chosen randomly. We included in each of the three sets a single intentionally wrongfully categorized request, called a bogus request [156], and a wrong set of recommendations. Finally, we randomly shuffle the sample requests.

Table 31 Distribution of our dataset

	Number of requested URIs	Number of bogus requested URIs	Total
Direct match	24	1	25
Depth zero	24	1	25
Deep	24	1	25
Total	72	3	75

For the direct match dataset, we found a match in an ontology, thus it is correctly classified. We determined the category it belongs to, gathered candidates, filtered, scored, ranked, and selected the top three candidates. Table 32 shows the number of candidates collected for the direct match dataset and the total number of mementos for all candidates. For the other two sets, depth zero and deep, we do not have a

match in an ontology, thus we need to classify each requested URI. We classified the request using first level classification, then using deep classification, gathered candidates, filtered, scored, ranked, and selected the top three candidates. Table 33 and Table 34 show the number of candidates collected for the depth zero dataset and the deep dataset, the total number of mementos for all candidates, and the percentage of mementos that resolved with a status code `200 OK` (Section 2.2.4). In all three datasets, we ranked the candidates based on the overall score of temporal similarity, URI similarity, popularity, and webpage quality (Section 5.6). After that, we selected the top three candidates ordered by total score and recommended them to the reviewers. In most cases, we have many more than three candidates available, so our filtering and ranking algorithm helps to determine the best candidates to recommend.

We use human evaluation to determine if a request is correctly classified in its appropriate category and if we recommended the top three related candidates. We asked five reviewers all with a masters degree in computers science, with some knowledge on web archive, two were males and three were females. The reviewers were provided with some instructions of how to answer the survey.

The reviewers were asked to determine if we classified the requested URI to the corresponding category by moving the slider to the appropriate depth. Then we show the top three recommended candidates and ask if the reviewer agrees, disagrees, or does not know, regarding the ranking of these candidates as shown in Figure 37.

Fig. 37 A sample question to be evaluated by the evaluators

Table 32 The number of candidates, the number of archived candidates, and the total number of mementos in the direct match dataset

Request	Number of candidates	Number of archived candidates	Total mementos for all candidates
1	110	88	63,711
2	12	7	903
3	141	96	45,542
4	36	25	5,278
5	53	29	7,608
6	214	145	26,350
7	339	263	49,218
8	81	69	26,477
9	64	44	20,770
10	69	44	56,730
11	82	55	85,630
12	8	6	1,339
13	28	21	6,797
14	117	57	86,618
15	172	97	43,749
16	102	78	240,505
17	64	53	155,627
18	12	9	3,208
19	35	27	24,999
20	22	17	8,152
21	69	44	56,724
22	10	7	3,358
23	145	109	82,359
24	112	78	13,005
Total	2097	1,468	1,114,657

Table 33 The number of candidates, the number of archived candidates, and the total number of mementos in the depth zero dataset

Request	Number of candidates	Number of archived candidates	Total mementos for all candidates
1	7	7	2,300
2	7	5	321
3	37	28	10,162
4	3	3	196
5	17	14	15,023
6	8	8	402
7	6	6	2,733
8	252	229	10,626
9	19	11	5,361
10	55	42	4,986
11	6	6	2,213
12	9	6	1,925
13	3	3	2,291
14	8	5	9,563
15	103	82	11,124
16	17	14	15,027
17	12	10	1,266
18	32	24	1,363
19	9	5	675
20	3	3	702
21	63	40	10,156
22	252	173	15,610
23	12	9	2,916
24	8	4	9,787
Total	948	737	136,728

Table 34 The number of candidates, the number of archived candidates, and the total number of mementos in the deep dataset

Request	Number of candidates	Number of archived candidates	Total mementos for all candidates
1	58	40	1,103
2	22	17	4,505
3	9	6	1,002
4	12	9	13,375
5	6	4	569
6	45	29	12,497
7	7	6	214
8	17	10	5,166
9	13	7	915
10	5	3	329,033
11	6	4	332,047
12	42	29	3,294
13	14	8	1,555
14	44	33	2,458
15	19	10	4,739
16	63	41	9,878
17	34	15	1,736
18	5	3	332,038
19	5	3	419
20	4	3	1,691
21	19	10	4,739
22	5	5	2,101
23	26	21	2,384
24	34	22	3,147
Total	514	338	1,070,605

In each question, we show the requested URI, for example `apartments.com`, with a screen shot that is linked to the requested webpage from the archive if available, the datetime (shown as only the year for simplicity), for example `2000`, the category found shown by a slider to represent the depth of the hierarchy (the slider starts with "none" if all the categories are considered wrong), for example `None,Regional/Nort hAmerica/UnitedStates/Businessandeconomy/Realestate/Rentals`, and the top three candidates for each request and their datetime ordered by the calculated score, for example `rent.com` (2000) with a total score of 0.79, `apartmentguid.com` (2000) with a total score of 0.79, `vacancynet.com` (2001) with a total score of 0.78.

In Figure 37, we also show screen shots of the recommendations that are linked to the recommended archived webpage. The total score is shown to the reviewer to show that the recommendations are ranked based on the temporal similarity, URI similarity, popularity, and webpage quality.

Measuring the agreement between reviewers

To measure the agreement between the reviewers we use Fleiss' kappa (κ) [157, 158], a statistical measure to determine the reliability of agreement between multiple reviewers when classifying items. Equation 11 shows how we calculate κ:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e} \tag{11}$$

Equations 12-15 show how each part of Equation 11 is calculated:

$$\bar{P} = \frac{1}{N} \sum_{i=1}^{N} p_i \tag{12}$$

$$P_i = \frac{1}{n(n-1)} \left(\sum_{j=1}^{K} n_i^2 j - n \right) \tag{13}$$

$$\bar{P}_e = \sum_{j=1}^{N} p_j^2 \tag{14}$$

$$P_j = \frac{1}{N(n)} \sum_{i=1}^{N} n_{ij} \tag{15}$$

where n is the number of raters which is 5 in our case, N is the number of items being rated which is 75 in our case, and K is the number of categories which varies

depending on the part of question we are evaluating. When evaluating the category depth, part one in each question (Figure 37), the deepest category is 9. Also, when evaluating the recommendation ranking agreement, part two in each question (Figure 37), the number of selections are 3.

If $\kappa = 1.0$, this means that there is complete agreement between reviewers. Table 35 shows the significance interpretation of κ.

Table 35 Significance of κ [3]

κ	Interpretation
<0	Poor agreement
0.01 - 0.20	Slight agreement
0.21 - 0.40	Fair agreement
0.41 - 0.60	Moderate agreement
0.61 - 0.80	Substantial agreement
0.81 - 1.00	Almost perfect agreement

6.2.1 EVALUATING THE CATEGORIZATION

In this section, we want to evaluate the performance of categorizing the requested URIs, part one of each question (Figure 37). The evaluators select the depth of the category, or selects "none" if none of the category depth is considered correct.

First, we measure the agreement rate between the reviewers on the categories, as shown in Table 36. We find that for the overall datasets $\kappa = 0.5293$ which is considered "moderate agreement". We also measure κ for each of the three sets and found all of them are considered "moderate agreement".

Table 36 Agreement rate in each dataset and the overall agreement rate of the categorization results

Dataset	κ	Interpretation
Direct match	0.6040	moderate agreement
Depth zero	0.5080	moderate agreement
Deep	0.4760	moderate agreement
Overall	0.5293	moderate agreement

For the case of bogus categorization questions for all three cases (Table 37), all the reviewers answered it correctly. This helps us determine if the reviewers understood the goal of the model and read each question carefully.

Table 37 Bogus categorization questions

Dataset	Requested URI	Category
Direct match	kidshealth.org	regional/Europe/United Kingdom/ business and economy/ construction and maintenance /do it yourself
Depth zero	thewomenshome.org	regional/oceania/Australia/ New South Wales/localities/st. leonards/ business and economy /computers and internet
Deep	kidswb.warnerbros.com/kids/shows-tomandjerry	computers/internet/ web design and development/promotion/ search engine optimization firms

Next, we measure the number of questions where the reviewer selected the full level, meaning the reviewer agreed on the full category that our model predicted. In Table 38, we found on average 65.83% of the questions were selected as fully correctly classified in direct match dataset, 26.6% for depth zero dataset, and 33.33% for deep dataset. This means that the reviewers agreed on our performance in classifying direct match dataset, then the deep dataset, and finally the depth zero dataset. We note that for the direct match category, we have provided the full category where the URI was found in DMOZ, so we expected this percentage to be high.

Table 38 Reviewers selection of full categorization level

Dataset	Percent
Direct match	65.8%
Depth zero	26.6%
Deep	33.3%

After that, we want to measure the performance of each level of categorization for the dataset. Table 39 shows the performance of classifying to different levels based on reviewers selection of level for the three sets of samples. We found that the best performance is level 1, where in all three sets the reviewers highest selection is the first level. In general, we found that based on the first three levels the direct match has the highest performance, followed by depth zero and deep dataset. As expected, the higher the level, the lower the agreement.

6.2.2 EVALUATING THE RECOMMENDATIONS

In this section, we want to evaluate the performance of recommending URIs,

Table 39 Performance of each classification level in each dataset

Classification	Direct match			Depth zero			Deep		
Level	Number of URIs	Average selection	Percent	Number of URIs	Average selection	Percent	Number of URIs	Average selection	Percent
Level 1	24	23.4	97.5%	24	20.6	85.8%	24	17.2	71.7%
Level 2	24	22.2	92.5%	24	16.6	69.1%	24	13.0	54.2%
Level 3	22	18.2	82.7%	22	11.2	50.9%	18	9.0	50.0%
Level 4	17	13.0	76.5%	15	4.4	29.3%	14	5.6	40.0%
Level 5	11	8.8	80.0%	8	2.4	30.0%	8	2.0	25.0%
Level 6	9	6.0	66.7%	5	1.4	28.0%	3	1.0	33.3%
Level 7	4	1.8	45.0%	3	0.8	26.7%	0	-	-
Level 8	4	1.6	40.0%	1	0.2	20.0%	0	-	-
Level 9	0	-	-	1	0.2	20.0%	0	-	-

part two of each question (Figure 37). The evaluators select their agreement to the selected recommendations by choosing: "agree", "do not agree", or "I do not know" if it is not clear to the evaluators if the recommendations are related to the original URI.

First, we measure the agreement rate between the reviewers for the ranked recommendations, as shown in Table 40. We find that for the overall datasets $\kappa = 0.5486$, which is considered "moderate agreement". We also measure κ for each of the three sets and found that the direct match dataset $\kappa = 0.7240$, which is considered "substantial agreement". For depth zero dataset $\kappa = 0.4583$, which is considered "moderate agreement". Finally, for the deep dataset $\kappa = 0.4600$, which is considered "moderate agreement".

Table 40 Agreement rate in each dataset and the overall agreement rate of the recommendation results

Dataset	κ	Interpretation
Direct match	0.7240	substantial agreement
Depth zero	0.4583	moderate agreement
Deep	0.4600	moderate agreement
Overall	0.5486	moderate agreement

For the case of bogus recommendations for all three cases (Table 41), all the reviewers answered it correctly by selecting "Do not agree". This helps us determine if the reviewers understood the goal of the model and read each question carefully.

Table 41 Bogus recommendation questions

Dataset	Requested URI	Ranked recommendations
Direct match	kidshealth.org	diyswimmingpools.co.uk buildingadvice.co.uk dustup.co.uk
Depth zero	thewomenshome.org	zonecomputing.com.au ewebdesign.com.au ico.com.au
Deep	kidswb.warnerbros.com/kids/shows-tomandjerry	mwi.com metamend.com mastermynde.com

After that, we want to measure how well the reviewers agreed with the recommendations for each dataset. The results are shown in Table 42. For each dataset and answer pair, we show the average number of recommendations chosen and the percentage for that dataset. We found that the direct match dataset performed the best, where 76.7% of the recommendations were marked as "agree". This was followed by depth zero dataset, where 45.0% of the recommendations were marked as "agree". The dataset with the least agreement is the deep dataset, where 40.0% of the recommendations were marked as "agree". Also, we found that the highest percentage of unclear recommendations was in the deep dataset where on average 4.4 recommendations per reviewer were selected as "I do not know".

Table 42 Performance of each recommendation agreement selection

	Agree		Do not agree		I do not know	
	Average Selections	Percent	Average Selections	Percent	Average Selections	Percent
Direct match	18.4	76.7%	3.2	13.3%	2.4	10.0%
Depth zero	10.8	45.0%	9.4	39.2%	3.8	15.8%
Deep	9.6	40.0%	10	41.7%	4.4	18.3%

Table 43 shows that for the direct match dataset, all recommendations were marked as "agree" by at least one reviewer. For the depth zero and deep datasets, that was reduced to 83.33%. We also found that reviewers chose "do not agree" more often for the depth zero and deep datasets, with 83.33% of recommendations receiving a "do not agree" rating from at least one reviewer. The percentage for the direct match set was much lower at 33.33%.

Table 43 Percentage of at least one marked recommendation as "agree", "do not agree", and "I do not know"

Dataset	At least one agree	At least one do not agree	At least one I do not know
Direct match	100%	33.33%	33.33%
Depth zero	83.33%	83.33%	58.33%
Deep	83.33%	83.33%	62.50%

6.2.3 EVALUATING BOTH THE CATEGORIZATION AND THE RECOMMENDATIONS

In Table 44 we show the percentages of different cases of the categorization and the recommendations in the model. When selecting the full categorization, reviewers agreed with 80.3% of the the recommendations, which is much higher than "do not agree" and "I do not know". This indicates the reviewer is more likely to agree on the recommendations when selecting the full categorization. When selecting at least level one categorization, reviewers agreed with 62.1% of the recommendations, which is higher than "do not agree" and "I do not know". This indicates the reviewer is more likely to agree on the recommendations when the selecting at least level one of the categorization. But when selecting the first level only, reviewers did not agree with 57.5% of the recommendations, which is higher than "agree" with 25.5% and "I do not know" with 17.0%. This indicates that performing deep level categorization is worth the effort because it increases the agreement with the recommendations greatly. Finally, when selecting "none" as the categorization, reviewers did not agree with 77.8% of the recommendations, which is higher than "agree" and "I do not know". This indicates that when the reviewer does not agree with the categorization, it is highly likely that they will not agree with the recommendation.

6.3 SUMMARY

In this chapter, we evaluated both the deep classification and the entire model. For the deep level classification we measure the accuracy for each classification level. We found that 63.45% of the URIs are correctly classified in the first level. We also found that 44.89% of the correctly classified URIs contain a word that exists in a dictionary. Also, 50.07% of the correctly classified URIs contain long strings in the

Table 44 Evaluating different cases for both the categorization and the recommendations

Categorization	Recommendation	Total	Percent
Selecting the full categorization	**Agree**	122	**80.3%**
	Do not agree	11	7.2%
	I do not know	19	12.5%
At least level one	**Agree**	190	**62.1%**
	Do not agree	71	23.2%
	I do not know	45	14.7%
First level only	**Agree**	12	25.5%
	Do not agree	27	**57.5%**
	I do not know	8	17.0%
Selecting none	**Agree**	4	7.4%
	Do not agree	42	**77.8%**
	I do not know	8	14.8%

domain.

We tested our overall method on a sample of 72 requests from the Wayback Machine access log, each with its corresponding category and with the top three candidates. The requests were from three different datasets: requests that are found in an ontology ("direct match"), requests that are not found in an ontology and have depth zero ("depth zero"), and requests that are not found in an ontology and have a path ("deep"). For the direct match we determined its classification, gathered candidates, scored, ranked, and selected the top three recommendation. For the other two sets we used first level classification and deep classification, gathered candidates, scored, ranked, and selected the top three recommendation. Overall, when selecting the full categorization, reviewers agreed with 80.3% of the recommendations, which is much higher than "do not agree" and "I do not know". This indicates the reviewer is more likely to agree on the recommendations when selecting the full categorization. But when selecting the first level only, reviewers did not agree with 57.5% of the recommendations, which is higher than "agree" with 25.5% and "I do not know" with 17.0%. This indicates that having deep level categorization improves the performance of finding recommendations.

CHAPTER 7

CONTRIBUTIONS, CONCLUSIONS, AND FUTURE

WORK

In this chapter, we revisit the research questions and mention the methods performed to solve them, the contributions, the future work, and the conclusions.

7.1 RESEARCH QUESTIONS REVISITED

Our main research question is "**Given a URI, how can we build a model for selecting and ranking possible recommendation webpages at a Web archive?**"

Our goal is to allow Web archives to provide recommendations for user queries, potentially surfacing high-quality, but underutilized webpages. We start with a user requesting a URI to finally constructing a ranked list of K archived recommendations that are similar to the requested URI. After the user requests a URI, we detect the URI semantics, map tokens to an ontology, gather candidates, filter candidates, score candidates, and ranking candidates. Each of the steps addresses a research question. Finally, the user will be provided with a ranked list of K recommendations.

Our research questions are:

RQ1: Given a URI, how much information can we learn about the referenced webpage without examining the content of the page?

Here, we determine the tokenizing method. In addition, since DMOZ contains a list categorized URI, we notice that some hostname are categorized in multiple main categories and other hostnames appear in a single category multiple time. For this reason, we check if the requested URI is in the black list, which is the list of hostnames that are hard to classify due to existing in multiple categories. In addition, we check if the URI is in the white list, which is the list of hostnames that we know what main category it belongs to.

RQ2: Given a set of terms that describes a webpage, how do we map the terms to an ontology of webpages?

Here we select the ontology that we are going to use to classify the URI, for both

first-level classification and deep level classification. The first-level classification is to determine the main category that the requested URI belongs to. Deep level classification is to determine the deep-level category that the requested URI belongs to. At the end of this section we will have a list of candidates.

RQ3: Given a set of webpages discovered through the ontology, what metrics do we use to filter the best candidates for recommendation?
After categorizing the requested URI we will end-up with a list of candidates that belong to the categorized URI. Here we removed any candidates that are not archived, since we want to recommend archived webpages only.

RQ4: Given a set of candidate webpages for recommendation, how do we rank the candidates?
Here we score and rank the candidates based on several features. The features are temporal similarity, webpage popularity, URI similarity, and archival quality. Finally, we select the top K ranked recommendations. Note that if the number of candidates are less then K, then we would collect, filter, and score more candidates. Note that we do not have a static weight for each feature, but the administrator can set the weights based on users' needs.

7.2 CONTRIBUTIONS

This dissertation contributes to the fields of creating recommendations of archived webpages as follows:

- We proposed steps to find recommendations to a requested archived URI. The steps include determining URI semantics, mapping tokens to an ontology, filtering candidates, and ranking candidates (Section 5.1).

- We proposed a solution for empty URIs, where the tokens appear only once in the dataset, by using Wikipedia to find an article that contains the requested URI as an official website. This method resolved 70% of the URIs that contains empty tokens (Section 5.2).

- We tested different tokenization methods, such as tokens, all-grams from tokens, all-grams from URI. For each tokenization method we include four cases: where all URI tokens are included, URI tokens without top level domain (TLD), URI tokens without TLD and numbers, and URI tokens without TLD and stopwords. We found the highest F_1 is for all-grams from URI without TLD and

numbers (Section 5.2.1).

- We created a black list of hostnames, which are hostnames that are hard to classify because they appear in multiple categories (Section 5.2.2). Our black list includes 954 hostnames, `https://github.com/oduwsdl/Recommending-Archived-Webpages`.

- We created a white list of hostnames, which are hostnames that are easily classified because they appear in a single category (Section 5.2.3). Our white list includes 9,146 hostnames, `https://github.com/oduwsdl/Recommending-Archived-Webpages`.

- We used both DMOZ and Wikipedia as ontologies to classify URIs. We found that DMOZ hostname diversity is 0.55 and Wikipedia's is 0.03. In addition, we found that DMOZ contains 50.57% depth zero URIs and Wikipedia contains 3.40% depth zero URIs (Section 4.5).

- We used a deep classification method and found that 63.45% of the correctly classified URIs in DMOZ are of depth zero (Section 6.1).

- We used a deep classification method and found that 44.89% of the correctly classified URIs in DMOZ contain at least one word from a dictionary (Section 6.1).

- We used a deep classification method and found that 50.07% of the correctly classified URIs in DMOZ contain long strings in the domain (Section 6.1).

- We analyzed the properties of a sample of URIs requested to the Wayback Machine access log and found that the large majority were of depth 0, meaning that our classification will rely largely on domain information (Section 4.3.4).

- We tested the model using human evaluation and found that when selecting the full categorization, reviewers agreed with 80.3% of the the recommendations, which is much higher than "do not agree" and "I do not know". This indicates the reviewer is more likely to agree on the recommendations when selecting the full categorization. But when selecting the first level only, reviewers did not agree with 57.5% of the recommendations, which is higher than "agree" with 25.5% and "I do not know" with 17.0%. This indicates that having deep level

categorization improves the performance of finding recommendations (Section 6.2.3).

7.3 FUTURE WORK

Future work includes adding other languages, which could be accomplished by including ontologies that include other languages. For example, in DMOZ the category World could be included. In addition, finding additional URIs in archived documents can be retrieved from the archive efficiently, such as by using a CDX file which includes an indexed list of the archived webpages and other information such as the timestamp and the MIME type.

In the filtering step, in addition to filtering the dataset based on if it is archived or not, we can include filtering spam webpages from the candidates.

In ranking the candidates, multiple features could be added such as ranking based on how long the webpage has been not live. The closeness of the creation date of the request and the candidate could be added as a feature to score and rank candidates.

Finally, when evaluating the model we can increase the number evaluators and show why they do not agree with the recommendations provided.

7.4 CONCLUSIONS

In this work, we wanted to expand the usage of the archive by recommending archived webpages. Our work proposes a method to enhance the current response from web archives when a URI cannot be found. We used both DMOZ and Wikipedia to classify the request and find candidates. First, we check if the requested URI is classified in DMOZ or Wikipedia. If the requested URI is pre-classified, then we gather candidates, filter, score, and recommend the top K results. Next, we determine if the URI is in the black list, which are hostnames that are hard to classify due to existing in multiple categories. After that, we determine if the URI is in the white list, which are hostnames that are easily classified due to existing in a single category. In this case we perform deep classification. If the requested URI is not pre-classified we classify the URI using first level classification and then deep classification. This step results in determining the category of the requested URI, thus we gather the URIs that belong to the category as candidates. Then we filter the candidates based on if the webpage is archived. Next, we score and rank the

candidates based on archival quality, webpage popularity, temporal similarity, and URI similarity.

We found that the best method to classify the first level is using all-grams from the URI while filtering the TLD and numbers from the URI. Using a Naïve Bayes classifier resulted in a $F_1 = 0.61$. For the second level classification we measure the accuracy for each classification level. We found that 63.45% of the URIs are correctly classified in the first level. We also found that 44.89% of the correctly classified URIs contain a word that exists in a dictionary. Also, 50.07% of the correctly classified URIs contain long strings in the domain.

We evaluated the model by randomly selecting 72 requests from the Wayback access log, where 24 requests are direct matches to an ontology, 24 requests are not found in an ontology and has depth zero, and 24 requests are not found in an ontology and contains a path. The reviewers were asked to determine if the requested URIs are classified correctly by allowing them to select the depth of the category, and select the agreement on the top three candidates. Overall, when selecting the full categorization, reviewers agreed with 80.3% of the the recommendations, which is much higher than "do not agree" and "I do not know". This indicates the reviewer is more likely to agree on the recommendations when selecting the full categorization. But when selecting the first level only, reviewers did not agree with 57.5% of the recommendations, which is higher than "agree" with 25.5% and "I do not know" with 17.0%. This indicates that having deep level categorization improves the performance of finding recommendations.

REFERENCES

[1] E. Baykan, M. Henzinger, L. Marian, and I. Weber, "A comprehensive study of features and algorithms for URL-based topic classification," *ACM Transactions on the Web (TWEB)*, vol. 5, no. 3, 2011.

[2] E. Baykan, M. Henzinger, and I. Weber, "A comprehensive study of techniques for URL-based web page language classification," *ACM Transactions on the Web (TWEB)*, vol. 7, no. 1, 2013.

[3] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *biometrics*, pp. 159–174, 1977.

[4] I. Jacobs and N. Walsh, "Architecture of the World Wide Web, volume one." `https://w3.org/TR/webarch/`, 2004.

[5] X. Qi and B. D. Davison, "Knowing a web page by the company it keeps," in *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CKIM)*, pp. 228–237, 2006.

[6] G.-R. Xue, D. Xing, Q. Yang, and Y. Yu, "Deep classification in large-scale text hierarchies," in *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 619–626, 2008.

[7] H. C. Huurdeman, A. Ben-David, J. Kamps, T. Samar, and A. P. de Vries, "Finding pages on the unarchived web," in *Proceedings of the 14th IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, pp. 331–340, 2014.

[8] H. M. SalahEldeen and M. L. Nelson, "Carbon dating the web: Estimating the age of web resources," in *Proceedings of the Temporal Web Analytics Workshop (TempWeb)*, pp. 1075–1082, 2013.

[9] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifiers (URI): Generic syntax, Internet RFC 2396." `https://tools.ietf.org/html/rfc2396`, 1998.

[10] J. F. Brunelle, M. Kelly, H. SalahEldeen, M. C. Weigle, and M. L. Nelson, "Not all mementos are created equal: Measuring the impact of missing resources," *International Journal on Digital Libraries (IJDL)*, vol. 16, no. 3-4, pp. 283–301, 2015.

[11] J. Niu, "An overview of web archiving," *D-Lib Magazine*, vol. 18, no. 3/4, 2012.

[12] M. Klein and M. L. Nelson, "Moved but not gone: An evaluation of real-time methods for discovering replacement web pages," *International Journal on Digital Libraries (IJDL)*, vol. 14, no. 1-2, pp. 17–38, 2014.

[13] P. Opitz, "Internet Archive." `http://crawler.archive.org/index.html`, 2012.

[14] M. Kelly and M. C. Weigle, "WARCreate: Create Wayback-Consumable WARC files from any webpage," in *Proceedings of the 12th IEEE-CS/ACM Joint Conference on Digital Libraries (JCDL)*, pp. 437–438, 2012.

[15] X. Roche, "HTTrack: Website copier." `https://httrack.com`.

[16] D. Gomes, J. Miranda, and M. Costa, "A survey on web archiving initiatives," in *Proceedings of the International Conference on Theory and Practice of Digital Libraries (TPDL): Research and Advanced Technology for Digital Libraries*, pp. 408–420, 2011.

[17] J. Bailey, A. Grotke, K. Hanna, C. Hartman, E. McCain, C. Moffatt, and N. Taylor, "Web Archiving in the United States: A 2013 Survey - An NDSA Report." `https://blogs.loc.gov/thesignal/2014/10/results-from-the-2013-ndsa-u-s-web-archiving-survey/`, 2014.

[18] J. Bailey, A. Grotke, E. McCain, C. Moffatt, and N. Taylor, "Web Archiving in the United States: A 2016 Survey - An NDSA Report." `http://ndsa.org/documents/WebArchivingintheUnitedStates_A2016Survey.pdf`, 2017.

[19] M. Kelly, M. C. Weigle, and M. L. Nelson, "Archiving Your Facebook Pages Using Archive Facebook." NDIIPP/NDSA Partners Meeting Special Interest Session, July 2011. `http://www.cs.odu.edu/~mkelly/presentations/2011_ndiipp_archivefacebook.pptx`.

[20] J. Bailey, A. Grotke, K. Hanna, C. Hartman, E. McCain, C. Moffatt, and N. Taylor, "Web Archiving in the United States: A 2013 Survey." `http://ndsa.org/documents/NDSA_USWebArchivingSurvey_2013.pdf`.

[21] J. Bailey, A. Grotke, E. McCain, C. Moffatt, and N. Taylor, "Web Archiving in the United States: A 2016 Survey." `http://ndsa.org/documents/WebArchivingintheUnitedStates_A2016Survey.pdf`.

[22] B. Kahle, "Preserving The Internet," *Scientific American*, vol. 276, no. 3, pp. 82–83, 1997.

[23] H. Van de Sompel, M. L. Nelson, and R. Sanderson, "HTTP framework for time-based access to resource states – Memento, Internet RFC 7089." `http://tools.ietf.org/html/rfc7089`, 2013.

[24] International Organization for Standardization (ISO), "28500: 2017 Information and Documentation-WARC File Format," 2017.

[25] Internet archive, "CDX file format." `http://archive.org/web/researcher/cdx_file_format.php`, 2003.

[26] V. Goel, "Web archive metadata file specifications." `https://webarchive.jira.com/wiki/spaces/Iresearch/pages/13467719/Web+Archive+Metadata+File+Specification`, 2011.

[27] N. Freed, M. Baker, and B. Hoehrmann, "Media types," tech. rep., Tech. Rep., 2014, 2014.

[28] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol–HTTP/1.1, Internet RFC 2616." `https://w3.org/Protocols/rfc2616/rfc2616.html`, 1999.

[29] H. Van de Sompel, "Memento guide - introduction to Memento." `http://mementoweb.org/guide/quick-intro/`, 2015.

[30] M. L. Nelson, "Memento-datetime is not last-modified." `http://ws-dl.blogspot.com/2010/11/2010-11-05-memento-datetime-is-not-last.html`, 2010.

[31] V. Goel, "Beta Wayback Machine - now with site search!." `https://blog.arc hive.org/2016/10/24/beta-wayback-machine-now-with-site-search/`, 2016.

[32] Y. AlNoamany, A. AlSum, M. C. Weigle, and M. L. Nelson, "Who and what links to the Internet Archive," *International Journal on Digital Libraries (IJDL)*, vol. 14, no. 3-4, pp. 101–115, 2014.

[33] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform resource identifier (uri): Generic syntax," tech. rep., 2004.

[34] T. Berners-Lee, "Cool URIs don't change." `https://w3.org/Provider/Styl e/URI`, 1998.

[35] P. Opitz, "Clean URLs for better search engine ranking." `http:// contentwithstyle.co.uk/content/clean-urls-for-a-better-search- engine-ranking/`, 2006.

[36] A. M. Cohen, R. Mariani, S. K. Rajan, and B. Tabbara, "URL mapping methods and systems," in *Google Patents, US Patent 6,654,741*, 2003.

[37] X. Qi and B. D. Davison, "Web page classification: Features and algorithms," *ACM Computing Surveys (CSUR)*, vol. 41, no. 2, p. 12, 2009.

[38] Ying, Cao and Run-Ying, Duan, "Novel top-down methods for hierarchical text classification," *Procedia Engineering*, vol. 24, pp. 329–334, 2011.

[39] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, "Supervised machine learning: A review of classification techniques," *Proceedings of the Emerging Artificial Intelligence Applications in Computer Engineering*, vol. 160, pp. 3–24, 2007.

[40] K. M. Leung, "Naive Bayesian classifier." `http://cis.poly.edu/~mleung/FR E7851/f07/naiveBayesianClassifier.pdf`, 2007.

[41] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proceedings of the European Conference on Machine Learning*, pp. 137–142, 1998.

[42] V. Vapnik, *The nature of statistical learning theory.* Springer science and business media, 2013.

[43] K. Nigam, J. Lafferty, and A. McCallum, "Using Maximum Entropy for text classification," in *Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering*, vol. 1, pp. 61–67, 1999.

[44] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra, "A Maximum Entropy approach to natural language processing," *Computational Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[45] J. Brank, D. Mladenic, and M. Grobelnik, "Gold standard based ontology evaluation using instance assignment," in *Proceedings of the Workshop on Evaluation of Ontologies for the Web (EON)*, 2006.

[46] C. Fellbaum, "WordNet." `https://wordnet.princeton.edu`, 1998.

[47] C. Sherman, "Humans do it better: Inside the open directory project.," *Online*, vol. 24, no. 4, 2000.

[48] S. G. Ainsworth, A. Alsum, H. M. SalahEldeen, M. C. Weigle, and M. L. Nelson, "How much of the web is archived?," in *Proceedings of the 11th IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, pp. 133–136, 2011.

[49] A. AlSum, *Web Archive Services Framework for Tighter Integration Between the Past and Present Web*. PhD thesis, Old Dominion University, 2014.

[50] L. M. Alkwai, M. L. Nelson, and M. C. Weigle, "How well are Arabic websites archived?," in *Proceedings of the 15th IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, pp. 223–232, 2015.

[51] L. M. Alkwai, M. L. Nelson, and M. C. Weigle, "Comparing the archival rate of Arabic, English, Danish, and Korean language web pages," *ACM Transactions on Information Systems (TOIS)*, vol. 36, no. 1, pp. 1:1–1:34, 2017.

[52] G. A. Monroe, J. C. French, and A. L. Powell, "Obtaining language models of web collections using query-based sampling techniques," in *Proceedings of the 35th annual Hawaii International Conference on System Sciences (HICSS)*, pp. 1241–1247, 2002.

[53] A. Gulli and A. Signorini, "The indexable web is more than 11.5 billion pages," in *Proceedings of the Special Interest Tracks and Posters of the 14th International Conference on World Wide Web (WWW)*, pp. 902–903, 2005.

[54] E. Baykan, M. Henzinger, L. Marian, and I. Weber, "Purely URL-based topic classification," in *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pp. 1109–1110, 2009.

[55] L. M. Alkwai, "Using Wikipedia to build a corpus, classify text, and more." `https://ws-dl.blogspot.com/2018/12/2018-12-03-using-wikipe dia-to-build.html`, 2018.

[56] R. Glott, P. Schmidt, and R. Ghosh, "Wikipedia survey–overview of results," *United Nations University: Collaborative Creativity Group*, pp. 1158–1178, 2010.

[57] A. Gattani, D. S. Lamba, N. Garera, M. Tiwari, X. Chai, S. Das, S. Subramaniam, A. Rajaraman, V. Harinarayan, and A. Doan, "Entity extraction, linking, classification, and tagging for social media: a Wikipedia-based approach," *Proceedings of the VLDB Endowment*, vol. 6, no. 11, pp. 1126–1137, 2013.

[58] P. Melville and V. Sindhwani, "Recommender systems," in *Encyclopedia of Machine Learning*, pp. 829–838, 2011.

[59] N. Rubens, D. Kaplan, and M. Sugiyama, "Active learning in recommender systems," in *Recommender Systems Handbook*, 2011.

[60] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[61] E. Rich, "User modeling via stereotypes," *Cognitive Science*, vol. 3, no. 4, pp. 329–354, 1979.

[62] J. Beel, "Towards effective research-paper recommender systems and user modeling based on mind maps," Tech. Rep. arXiv:1703.09109, 2017.

[63] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*, pp. 325–341, 2007.

[64] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[65] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*, pp. 191–198, 2016.

[66] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. Van Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston, *et al.*, "The YouTube video recommendation system," in *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys)*, pp. 293–296, 2010.

[67] H. Small, "Co-citation in the scientific literature: A new measure of the relationship between two documents," *Journal of the Association for Information Science*, vol. 24, no. 4, pp. 265–269, 1973.

[68] D. C. Liu, S. Rogers, R. Shiau, D. Kislyuk, K. C. Ma, Z. Zhong, J. Liu, and Y. Jing, "Related pins at Pinterest: The evolution of a real-world recommender system," in *Proceedings of the 26th International Conference on World Wide Web Companion*, pp. 583–592, 2017.

[69] R. Burke, "Hybrid recommender systems: Survey and experiments," *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[70] C. Chi, C. Ding, and A. Lim, "Word segmentation and recognition for web document framework," in *Proceedings of the 8th International Conference on Information and Knowledge Management (CKIM)*, pp. 458–465, 1999.

[71] P. Koehn and K. Knight, "Empirical methods for compound splitting," in *Proceedings of the 10th Conference on European Chapter of the Association for Computational Linguistics-Volume 1*, pp. 187–193, 2003.

[72] T. Segaran and J. Hammerbacher, *Beautiful data: the stories behind elegant data solutions.* O'Reilly Media, 2009.

[73] S. Khaitan, A. Das, S. Gain, and A. Sampath, "Data-driven compound splitting method for English compounds in domain names," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CKIM)*, pp. 207–214, 2009.

[74] E. Baykan, M. Henzinger, and I. Weber, "Web page language identification based on URLs," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, pp. 176–187, 2008.

[75] S. Raju and R. Udupa, "Extracting advertising keywords from URL strings," in *Proceedings of the 21st International Conference on World Wide Web (WWW)*, pp. 587–588, 2012.

[76] M.-S. Lin, C.-Y. Chiu, Y.-J. Lee, and H.-K. Pao, "Malicious URL filtering? a big data application," in *Proceedings of the IEEE International Conference on Big Data*, pp. 589–596, 2013.

[77] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research (JMLR)*, vol. 7, pp. 551–585, 2006.

[78] M. Dredze, K. Crammer, and F. Pereira, "Confidence-weighted linear classification," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 264–271, 2008.

[79] G. Berardi, A. Esuli, D. Marcheggiani, and F. Sebastiani, "ISTI@ TREC Microblog Track 2011: Exploring the use of hashtag segmentation and text quality ranking," in *Proceedings of the Text Retrieval Conference (TREC)*, 2011.

[80] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[81] A. Cui, M. Zhang, Y. Liu, S. Ma, and K. Zhang, "Discover breaking events with popular hashtags in Twitter," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CKIM)*, pp. 1794–1798, 2012.

[82] A. Beale, "12dicts word lists." `http://wordlist.aspell.net/12dicts/`, 2007.

[83] R. Zhang, W. Li, D. Gao, and Y. Ouyang, "Automatic Twitter topic summarization with speech acts," *IEEE transactions on Audio, Speech, and Language Processing*, vol. 21, no. 3, pp. 649–658, 2013.

[84] Z. Wu and G. Tseng, "Chinese text segmentation for text retrieval: Achievements and problems," *Journal of the American Society for Information Science*, vol. 44, no. 9, p. 532, 1993.

[85] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang, "Clustering user queries of a search engine," in *Proceedings of the 10th International Conference on World Wide Web (WWW)*, pp. 162–168, 2001.

[86] R. Baeza-Yates, C. Hurtado, and M. Mendoza, "Query recommendation using query logs in search engines," in *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pp. 588–596, 2004.

[87] D. Koller and M. Sahami, "Hierarchically classifying documents using very few words," *Proceedings of the 14th International Conference on Machine Learning (ICML)*, pp. 170–178, 1997.

[88] G. Attardi, A. Gulli, and F. Sebastiani, "Automatic web page categorization by link and context analysis," *Proceedings of European Symposium on Telematics, Hypermedia and Artificial Intelligence (THAI)*, vol. 99, no. 99, pp. 105–119, 1999.

[89] M.-Y. Kan, "Web page classification without the web page," in *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers and Posters*, pp. 262–263, 2004.

[90] J. Cho and H. Garcia-Molina, "WebBase and the Stanford Interlib Project," in *Proceedings of the International Conference on Digital Libraries: Research and Practice*, pp. 180–181, 2000.

[91] M.-Y. Kan and H. O. N. Thi, "Fast webpage classification using URL features," in *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CKIM)*, pp. 325–326, 2005.

[92] Carnegie Mellon University, "Universities WebKB data." `http://cs.cmu.edu/~webkb/`, 1997.

[93] J. Kustanowitz, B. Shneiderman, and B. Kules, "Categorizing web search results into meaningful and stable categories using fast-feature techniques," in

Proceedings of the 6th IEEE-CS/ACM Joint Conference on Digital Libraries (JCDL), pp. 210–219, 2006.

[94] M. I. Devi and K. Selvakuberan, "Fast web page categorization without the web page," in *Proceedings of the International Conference on Semantic Web and Digital Libraries (ICSD)*, 2007.

[95] R. Rajalakshmi and C. Aravindan, "Web page classification using N-gram based URL features," in *Proceedings of the 5th International Conference on Advanced Computing (ICoAC)*, pp. 15–21, 2013.

[96] M. I. Devi, R. Rajaram, and K. Selvakuberan, "Machine learning techniques for automated web page classification using url features," in *Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA)*, vol. 2, pp. 116–120, 2007.

[97] R. Rajalakshmi and C. Aravindan, "Naive bayes approach for website classification," in *Proceedings of the Information Technology and Mobile Communication. Communications in Computer and Information Science*, vol. 147, 2011.

[98] T. A. Abdallah, *A New Method For URL-based Web Page Classification Using N-gram Language Models.* PhD thesis, The University of East Anglia, 2013.

[99] T. A. Abdallah and B. de La Iglesia, "URL-based web page classification: With N-gram language models," in *Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management (IC3K)*, pp. 19–33, 2014.

[100] J. Lin, M. Gholami, and J. Rao, "Infrastructure for supporting exploration and discovery in web archives," in *Proceedings of the 23rd International Conference on World Wide Web (WWW)*, pp. 851–856, 2014.

[101] J. Lin, I. Milligan, J. Wiebe, and A. Zhou, "Warcbase: Scalable analytics infrastructure for exploring web archives," *Journal on Computing and Cultural Heritage (JOCCH)*, vol. 10, no. 4, p. 22, 2017.

[102] H. Holzmann, V. Goel, and A. Anand, "ArchiveSpark: Efficient web archive access, extraction and derivation," in *Proceedings of the 16th IEEE-CS/ACM Joint Conference on Digital Libraries (JCDL)*, pp. 83–92, 2016.

[103] H. Holzmann, V. Goel, and E. N. Gustainis, "Universal distant reading through metadata proxies with ArchiveSpark," in *Proceedings of the IEEE International Conference on Big Data*, pp. 459–464, 2017.

[104] H. Holzmann, *Concepts and tools for the effective and efficient use of web archives.* PhD thesis, 2019.

[105] Y. AlNoamany, *Using Web Archives to Enrich the Live Web Experience Through Storytelling.* PhD thesis, Old Dominion University, 2016.

[106] S. M. Jones, "Storify will be gone soon, so how do we preserve the stories?." `https://ws-dl.blogspot.com/2017/12/2017-12-14-storify-will-be-gone-soon-so.html`, 2017.

[107] S. Lawrence, D. M. Pennock, G. W. Flake, R. Krovetz, F. Coetzee, E. Glover, F. Nielsen, A. Kruger, and C. L. Giles, "Persistence of web references in scientific research," *IEEE Computer*, vol. 34, no. 2, pp. 26–31, 2001.

[108] W. Koehler, "Web page change and persistence? a four-year longitudinal study," *Journal of the American Society for Information Science and Technology*, vol. 53, no. 2, pp. 162–171, 2002.

[109] M. L. Nelson and B. D. Allen, "Object persistence and availability in digital libraries," *D-Lib Magazine*, vol. 8, no. 1, 2002.

[110] D. Spinellis, "The decay and failures of web references," *Communications of the ACM*, vol. 46, no. 1, pp. 71–77, 2003.

[111] M. Klein, H. Van de Sompel, R. Sanderson, H. Shankar, L. Balakireva, K. Zhou, and R. Tobin, "Scholarly context not found: one in five articles suffers from reference rot," *Public Library of Science (PLOS ONE)*, vol. 9, no. 12, pp. 1–39, 2014.

[112] S. M. Jones, H. Van de Sompel, H. Shankar, M. Klein, R. Tobin, and C. Grover, "Scholarly context adrift: Three out of four URI references lead to changed content," *Public Library of Science (PLOS ONE)*, vol. 11, no. 12, pp. 1–32, 2016.

[113] M. Costa and M. J. Silva, "Understanding the information needs of web archive users," in *Proceedings of the 10th International Web Archiving Workshop (IWAW10)*, 2010.

[114] M. Thelwall and L. Vaughan, "A fair history of the web? examining country balance in the Internet Archive," *Library and Information Science Research*, vol. 26, no. 2, pp. 162–176, 2004.

[115] F. McCown and M. L. Nelson, "Characterization of search engine caches," in *Proceedings of the Archiving Conference*, pp. 48–52, 2007.

[116] N. Kanhabua, P. Kemkes, W. Nejdl, T. N. Nguyen, F. Reis, and N. K. Tran, "How to search the Internet Archive without indexing it," in *Proceedings of the International Conference on Theory and Practice of Digital Libraries (TPDL)*, pp. 147–160, 2016.

[117] L. Francisco-Revilla, F. Shipman, R. Furuta, U. Karadkar, and A. Arora, "Managing change on the web," in *Proceedings of the 1st IEEE-CS/ACM Joint Conference on Digital libraries (JCDL)*, pp. 67–76, 2001.

[118] J. Martinez-Romo and L. Araujo, "Recommendation system for automatic recovery of broken web links," in *Proceedings of the Ibero-American Conference on Artificial Intelligence*, pp. 302–311, 2008.

[119] J. Martinez-Romo and L. Araujo, "Retrieving broken web links using an approach based on contextual information," in *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*, pp. 351–352, 2009.

[120] J. Martinez-Romo and L. Araujo, "Analyzing information retrieval methods to recover broken web links," in *Proceedings of the European Conference on Information Retrieval (ECIR)*, pp. 26–37, 2010.

[121] T. L. Harrison and M. L. Nelson, "Just-in-time recovery of missing web pages," in *Proceedings of the 17th Conference on Hypertext and Hypermedia*, pp. 145–156, 2006.

[122] T. L. Harrison, "Opal: In vivo based preservation framework for locating lost web pages," Master's thesis, 2005.

[123] M. Klein and M. L. Nelson, "Revisiting lexical signatures to (re-) discover web pages," in *Proceedings of the International Conference on Theory and Practice of Digital Libraries (TPDL)*, pp. 371–382, 2008.

[124] B. Haslhofer and N. Popitsch, "DSNotify-detecting and fixing broken links in linked data sets," in *Proceedings of the 20th International Workshop on Database and Expert Systems Application (DEXA'09)*, pp. 89–93, 2009.

[125] N. P. Popitsch and B. Haslhofer, "DSNotify: handling broken links in the web of data," in *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pp. 761–770, 2010.

[126] N. Popitsch and B. Haslhofer, "DSNotify–a solution for event detection and link maintenance in dynamic datasets," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 9, no. 3, pp. 266–283, 2011.

[127] H. C. Huurdeman, J. Kamps, T. Samar, A. P. de Vries, A. Ben-David, and R. A. Rogers, "Lost but not forgotten: Finding pages on the unarchived web," *International Journal on Digital Libraries (IJDL)*, vol. 16, no. 3-4, pp. 247–265, 2015.

[128] L. M. Alkwai, "Summary of Finding Pages on the Unarchived Web." http://ws-dl.blogspot.com/2016/10/2016-10-03-summary-of-finding-pages-on.html, 2016.

[129] Y. AlNoamany, M. C. Weigle, and M. L. Nelson, "Detecting off-topic pages within timemaps in web archives," *International Journal on Digital Libraries (IJDL)*, vol. 17, no. 3, pp. 203–221, 2016.

[130] S. M. Jones, M. C. Weigle, and M. L. Nelson, "The off-topic memento toolkit," tech. rep., 2018.

[131] Z. Bar-Yossef, A. Z. Broder, R. Kumar, and A. Tomkins, "Sic transit gloria telae: Towards an understanding of the web's decay," in *Proceedings of the 13th International Conference on World Wide Web (WWW)*, pp. 328–337, 2004.

[132] T. Lee, J. Kim, J. W. Kim, S.-R. Kim, and K. Park, "Detecting soft errors by redirection classification," in *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pp. 1119–1120, 2009.

[133] L. Meneses, R. Furuta, and F. Shipman, "Identifying "soft 404" error pages: analyzing the lexical signatures of documents in distributed collections," in *Proceedings of the International Conference on Theory and Practice of Digital Libraries (TPDL)*, pp. 197–208, 2012.

[134] E. Siregar, "Deploying the memento-damage service." `https://ws-dl.blogs pot.com/2017/11/2017-11-22-deploying-memento-damage.html`, 2017.

[135] L. Azzopardi and V. Vinay, "Retrievability: An evaluation measure for higher order information access tasks," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management (CKIM)*, pp. 561–570, 2008.

[136] M. C. Traub, T. Samar, J. van Ossenbruggen, J. He, A. de Vries, and L. Hardman, "Querylog-based assessment of retrievability bias in a large newspaper corpus," in *Proceedings of the 16th IEEE-CS/ACM Joint Conference on Digital Libraries (JCDL)*, pp. 7–16, 2016.

[137] G. Garvy, "Inequality of income: Causes and measurement," in *Studies in Income and Wealth, Volume 15*, pp. 25–47, 1952.

[138] J. Russell, "Topsy, the popular social analytics service bought by Apple, closes down." `https://techcrunch.com/2015/12/15/rip-topsy/`, 2015.

[139] G. Atkins, "Carbon Dating the Web, version 4.0." `http://ws-dl.blogspot.com/2017/09/2017-09-19-carbon-dating-web-version-40.html`, 2017.

[140] R. Geva, "Article's publication date extractor - an overview." `http://blog.webhose.io/2015/12/13/articles-publication-date-extractor-an-overview/`, 2015.

[141] E. Miller, "An introduction to the resource description framework," *Bulletin of the American Society for Information Science and Technology*, vol. 25, no. 1, pp. 15–19, 1998.

[142] A. Nwala, "An exploration of URL diversity measures." `https://ws-dl.blogspot.com/2018/05/2018-05-04-exploration-of-url-diversity.html`, 2018.

[143] J. R. Pierce, *An introduction to information theory: symbols, signals and noise.* Courier Corporation, 2012.

[144] D. D. Palmer, "A trainable rule-based algorithm for word segmentation," in *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 321–328, Association for Computational Linguistics, 1997.

[145] J. Goldsmith, "A Pythonic wrapper for the Wikipedia API ." `https://github.com/goldsmith/Wikipedia`, 2016.

[146] M. Majlis, "Python wrapper for Wikipedia." `https://github.com/martin-majlis/Wikipedia-API`, 2019.

[147] Y. AlNoamany, M. C. Weigle, and M. L. Nelson, "Access patterns for robots and humans in web archives," in *Proceedings of the 13th IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, pp. 339–348, 2013.

[148] I. D. Melamed, R. Green, and J. P. Turian, "Precision and recall of machine translation," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pp. 61–63, 2003.

[149] D. M. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[150] S. Bird, E. Klein, E. Loper, and J. Baldridge, "Multidisciplinary instruction with the natural language toolkit," in *Proceedings of the Third Workshop on Issues in Teaching Computational Linguistics*, pp. 62–70, Association for Computational Linguistics, 2008.

[151] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[152] J. Perkins, *Python text processing with NLTK 2.0 cookbook.* No. 1, Packt Publishing Ltd, 2010.

[153] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for idf," *Journal of documentation*, vol. 60, no. 5, pp. 503–520, 2004.

[154] A. Sun and E.-P. Lim, "Hierarchical text classification and evaluation," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pp. 521–528, IEEE, 2001.

[155] S. Alam and M. L. Nelson, "MemGator-A portable concurrent memento aggregator: Cross-platform CLI and server binaries in Go," in *Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries*, pp. 243–244, ACM, 2016.

[156] P. J. Lavrakas, *Encyclopedia of survey research methods*. Sage Publications, 2008.

[157] M. Banerjee, M. Capozzoli, L. McSweeney, and D. Sinha, "Beyond kappa: A review of interrater agreement measures," *Canadian journal of statistics*, vol. 27, no. 1, pp. 3–23, 1999.

[158] T. R. Nichols, P. M. Wisner, G. Cripe, and L. Gulabchand, "Putting the kappa statistic to use," *The Quality Assurance Journal*, vol. 13, no. 3-4, pp. 57–61, 2010.

APPENDIX A

WEB ARCHIVE FILE FORMATS

Listing 11-13, show the CDX, WARC, and WAT files that can be generated from crawling the URL `http://cs.odu.edu/~lalkwai/publication.html`. These are referenced in Section 2.1.

Listing 11 CDX file resulting from crawling the URL
`http://cs.odu.edu/~lalkwai/publication.html`

```
{
 CDX N b a m s k r V g
cs.odu.edu/~lalkwai/css/style-print.css 20180413034055
    http://www.cs.odu.edu/~lalkwai/css/style-print.css
    text/css 200 TJI7NJEGJVDUHII2N35YDGSOM36IVESU -
    7665403 20180413034055402.warc
cs.odu.edu/~lalkwai/css/style.css 20180413034055 http
    ://www.cs.odu.edu/~lalkwai/css/style.css text/css
    200 TJI7NJEGJVDUHII2N35YDGSOM36IVESU - 7664033
    20180413034055402.warc
cs.odu.edu/~lalkwai/odu.jpg 20180413034055 http://www.
    cs.odu.edu/~lalkwai/ODU.jpg image/jpeg 200
    NXQLI2FPP2G2EVULIHUXN4TFSW374QIA - 7676142
    20180413034055402.warc
cs.odu.edu/~lalkwai/publication.html 20180413034055
    http://www.cs.odu.edu/~lalkwai/publication.html text
    /html 200 LA4CFY57P7OIWQT47T7IMYNDPSDGY2IO - 2399
    20180413034055402.warc
}
```

Listing 12 WARC file resulting from crawling the URL
`http://cs.odu.edu/~lalkwai/publication.html`

```
{
WARC/1.0
WARC-Type: warcinfo
WARC-Date: 2018-04-13T03:40:55Z
WARC-Filename: 20180413034055402.warc
```

```
WARC-Record-ID: <urn:uuid:ef7bbdc3-beba-dc4e-c8f6-
    ade122dcf075>
Content-Type: application/warc-fields
Content-Length: 447

software: WARCreate/0.2017.6.6 http://warcreate.com
format: WARC File Format 1.0
conformsTo: http://bibnum.bnf.fr/WARC/
    WARC_ISO_28500_version1_latestdraft.pdf
isPartOf:  basic
description:  Crawl initiated from the WARCreate Google
     Chrome extension
robots: ignore
http-header-user-agent:  Mozilla/5.0 (X11; Linux x86_64
    ) AppleWebKit/537.36 (KHTML, like Gecko) Chrome
    /63.0.3239.132 Safari/537.36
http-header-from: warcreate@matkelly.com

WARC/1.0
WARC-Type: request
WARC-Target-URI: http://www.cs.odu.edu/~lalkwai/
    publication.html
WARC-Date: 2018-04-13T03:40:55Z
WARC-Concurrent-To: <urn:uuid:a9be70a0-a480-bab1-4676-
    b8dceb3c3365>
WARC-Record-ID: <urn:uuid:1cdb4b73-32d3-0191-1bb4-1
    fc40aa7a6f1>
Content-Type: application/http; msgtype=request
Content-Length: 512

GET /~lalkwai/publication.html HTTP/1.1
Accept: text/plain, */*; q=0.01
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit
    /537.36 (KHTML, like Gecko) Chrome/63.0.3239.132
    Safari/537.36
Referer: http://www.cs.odu.edu/~lalkwai/publication.
    html
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

```
Cookie: __utma
    =256360345.2034739502.1446824378.1473178021.1487193943.13;
     _ga=GA1.2.863692975.1446056072; _gid=GA1
    .2.1651099024.1523590838; _gat_UA-2088428-1=1

WARC/1.0
WARC-Type: metadata
WARC-Target-URI: http://www.cs.odu.edu/~lalkwai/
    publication.html
WARC-Date: 2018-04-13T03:40:55Z
WARC-Concurrent-To: <urn:uuid:dddc4ba2-c1e1-459b-8d0d-
    a98a20b87e96>
WARC-Record-ID: <urn:uuid:6fef2a49-a9ba-4b40-9f4a-5
    ca5db1fd5c6>
Content-Type: application/warc-fields
Content-Length: 537

outlink: http://www.cs.odu.edu/~lalkwai/publication.
    htmlODU.jpg E =EMBED_MISC
outlink: http://www.cs.odu.edu/~lalkwai/publication.
    htmlnull E link/@href
outlink: http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl
    -2015-arabic-sites.pdf L a/@href
outlink: https://dx.doi.org/10.1145/3041656 L a/@href
outlink: https://arxiv.org/abs/1703.03302 L a/@href
outlink: http://www.cs.odu.edu/~mkelly/papers/2017
    _jcdl_countingMementos.pdf L a/@href
outlink: http://www.cs.odu.edu/~lalkwai/ L a/@href
outlink: https://ws-dl.cs.odu.edu/ L a/@href

WARC/1.0
WARC-Type: response
WARC-Target-URI: http://www.cs.odu.edu/~lalkwai/
    publication.html
WARC-Date: 2018-04-13T03:40:55Z
WARC-Record-ID: <urn:uuid:34a0a293-80b9-ffdc-4f77-3
    d027a08bfbd>
Content-Type: application/http; msgtype=response
Content-Length: 2743

HTTP/1.1 200 OK
```

```
Server: nginx
Date: Fri, 13 Apr 2018 03:40:34 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding

<!DOCTYPE html><html><head>
    <title>Lulwah Alkwais' Publication list</title>
    <meta charset="utf-8">
    <meta http-equiv="Content-type" content="text/html;
        charset=utf-8">
    <meta name="viewport" content="width=device-width,
        initial-scale=1">
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family:  sans-serif;
        font-size: 14px;
    }
    div {
        width: 800px;
        margin: 0em auto;
        padding: 25px;
        background-color: #fff;
        border-radius: 1em;
    }
    a:link, a:visited {
        color: #38488f;
    }
    @media (max-width: 700px) {
        body {
            background-color: #fff;
        }
    }
    </style>
</head>
<body><left><img alt="Old Dominion University" height="
    150" src="ODU.jpg" width="150"></left>
```

```html
<div>
    <h1>Lulwah Alkwais' Publication list</h1>

  <ul>
      <li><a href="http://www.cs.odu.edu/~mln/pubs/jcdl
          -2015/jcdl-2015-arabic-sites.pdf">Lulwah M.
          Alkwai, Michael L. Nelson and Michele C.
          Weigle, "How Well Are Arabic Websites Archived
          ?", In Proceedings of the ACM/IEEE Joint
          Conference on Digital Libraries (JCDL).
          Knoxville, TN, June 2015, </a>Best Student
          Paper Award.</li><br>

        <li><a href="https://dx.doi.org/10.1145/3041656
            ">Lulwah M. Alkwai, Michael L. Nelson and
            Michele C. Weigle, "Comparing the Archival
            Rate of Arabic, English, Danish, and Korean
            Language Web Pages", ACM Transactions on
            Information Systems (TOIS), Vol. 36, No. 1,
            July 2017, pp. 1:1-1:34.</a></li><br>

        <li><a href="https://arxiv.org/abs/1703.03302">
            Mat Kelly, Lulwah M. Alkwai, Michael L.
            Nelson, Michele C. Weigle and Herbert Van de
             Sompel, "Impact of URI Canonicalization on
            Memento Count", Technical report arXiv
            :1703.03302, March 2017.</a></li><br>

  <li><a href="http://www.cs.odu.edu/~mkelly/papers
      /2017_jcdl_countingMementos.pdf">Mat Kelly, Lulwah
       M. Alkwai, Sawood Alam, Michael L. Nelson,
      Michele C. Weigle and Herbert Van de Sompel, "
      Impact of URI Canonicalization on Memento Count",
      In Proceedings of the ACM/IEEE Joint Conference on
       Digital Libraries (JCDL). Toronto, Ontario,
      Canada, June 2017, pp. 303-304,</a> Best Poster
      Award.</li>
  </ul>
</div>
<br>
```

```
<center><a href="http://www.cs.odu.edu/~lalkwai/">Back
    to Lulwah Alkwais' Old Dominion University Computer
    Science homepage</a><center><br>
<center><a href="https://ws-dl.cs.odu.edu/">Web Science
     and Digital Libraries Research Group in the
    Department of Computer Science at Old Dominion
    University</a></center>

</center></center></body></html>


WARC/1.0
WARC-Type: request
WARC-Target-URI: http://www.cs.odu.edu/~lalkwai/ODU.jpg
WARC-Date: 2018-04-13T03:40:55Z
WARC-Concurrent-To: <urn:uuid:a9be70a0-a480-bab1-4676-
    b8dceb3c3365>
WARC-Record-ID: <urn:uuid:3c5d51d3-11e6-5161-f800-71463
    fa4a8cc>
Content-Type: application/http; msgtype=request
Content-Length: 449


GET /~lalkwai/ODU.jpg HTTP/1.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit
    /537.36 (KHTML, like Gecko) Chrome/63.0.3239.132
    Safari/537.36
Accept: */*
Referer: http://www.cs.odu.edu/~lalkwai/publication.
    html
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: __utma
    =256360345.2034739502.1446824378.1473178021.1487193943.13;
     _ga=GA1.2.863692975.1446056072; _gid=GA1
    .2.1651099024.1523590838; _gat_UA-2088428-1=1


WARC/1.0
WARC-Type: request
WARC-Target-URI: http://www.cs.odu.edu/favicon.ico
WARC-Date: 2018-04-13T03:40:55Z
WARC-Concurrent-To: <urn:uuid:a9be70a0-a480-bab1-4676-
    b8dceb3c3365>
```

```
WARC -Record -ID: <urn:uuid:cb7d3c7b -787c -9e3c -75b1 -1
    f293c0b7e19 >
Content -Type: application/http; msgtype=request
Content -Length: 480

GET /favicon.ico HTTP/1.1
User -Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit
    /537.36 (KHTML, like Gecko) Chrome/63.0.3239.132
    Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://www.cs.odu.edu/~lalkwai/publication.
    html
Accept -Encoding: gzip, deflate
Accept -Language: en-US,en;q=0.9
Cookie: __utma
    =256360345.2034739502.1446824378.1473178021.1487193943.13;
     _ga=GA1.2.863692975.1446056072; _gid=GA1
    .2.1651099024.1523590838; _gat_UA -2088428 -1=1
}
```

Listing 13 WAT file resulting from crawling the URL
`http://cs.odu.edu/~lalkwai/publication.html`

```
{
WARC/1.0
WARC -Type: warcinfo
WARC -Date: 2018 -04 -13T03:43:33Z
WARC -Filename: 20180413034055402.warc
WARC -Record -ID: <urn:uuid:eb04fc6b -1f11 -4675 -8915 -
    a8df5bad2130 >
Content -Type: application/warc -fields
Content -Length: 120

Software -Info: archive -commons -0.0.1 - SNAPSHOT
    -2011 -04 -30 10:23:13
Extracted -Date: Fri, 13 Apr 2018 03:43:33 GMT

WARC/1.0
WARC -Type: metadata
WARC -Target -URI: 20180413034055402.warc
WARC -Date: 2018 -04 -13T03:40:55Z
```

```
WARC-Record-ID: <urn:uuid:3ee95b5b-a7e8-4259-b447-4
    a8b833d995f>
WARC-Refers-To: <urn:uuid:ef7bbdc3-beba-dc4e-c8f6-
    ade122dcf075>
Content-Type: application/json
Content-Length: 1087

{"Envelope":
{
""Format":"WARC",
"WARC-Header-Length":"230",
"Block-Digest":"sha1:FJP3JGJB6UU3QXDNUWLLC2MEZ7TNZ5PU",
"Actual-Content-Length":"447",
"WARC-Header-Metadata":
{
""WARC-Type":"warcinfo",
"WARC-Filename":"20180413034055402.warc",
"WARC-Date":"2018-04-13T03:40:55Z",
"Content-Length":"447",
"WARC-Record-ID":"<urn:uuid:ef7bbdc3-beba-dc4e-c8f6-
    ade122dcf075>",
"Content-Type":"application/warc-fields"},
"Payload-Metadata":
{
""Trailing-Slop-Length":"0",
"Actual-Content-Type":"application/warc-fields",
"Actual-Content-Length":"447",
"WARC-Info-Metadata":
{
""robots":"ignore",
"software":"WARCreate/0.2017.6.6 http://warcreate.com",
"http-header-from":"warcreate@matkelly.com",
"http-header-user-agent":"Mozilla/5.0 (X11; Linux
    x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/63.0.3239.132 Safari/537.36",
"conformsTo":"http://bibnum.bnf.fr/WARC/
    WARC_ISO_28500_version1_latestdraft.pdf",
"description":"Crawl initiated from the WARCreate
    Google Chrome extension",
"isPartOf":"basic",
"format":"WARC File Format 1.0"}}},
```

```
"Container":
{
""Compressed":false,
"Offset":"0",
"Filename":"20180413034055402.warc"}}

WARC/1.0
WARC-Type: metadata
WARC-Target-URI: http://www.cs.odu.edu/~lalkwai/
    publication.html
WARC-Date: 2018-04-13T03:40:55Z
WARC-Record-ID: <urn:uuid:cc1b9313-db6f-427b-ac52-038
    bfe3473f4>
WARC-Refers-To: <urn:uuid:1cdb4b73-32d3-0191-1bb4-1
    fc40aa7a6f1>
Content-Type: application/json
Content-Length: 1441

{"Envelope":
{
""Format":"WARC",
"WARC-Header-Length":"335",
"Block-Digest":"sha1:2HEW37YGOZ62YRCRTCJHD5ODWHBWZBMI",
"Actual-Content-Length":"512",
"WARC-Header-Metadata":
{
""WARC-Type":"request",
"WARC-Date":"2018-04-13T03:40:55Z",
"Content-Length":"512",
"WARC-Record-ID":"<urn:uuid:1cdb4b73-32d3-0191-1bb4-1
    fc40aa7a6f1>",
"WARC-Target-URI":"http://www.cs.odu.edu/~lalkwai/
    publication.html",
"WARC-Concurrent-To":"<urn:uuid:a9be70a0-a480-bab1
    -4676-b8dceb3c3365>",
"Content-Type":"application/http; msgtype=request"},
"Payload-Metadata":
{
""Trailing-Slop-Length":"4",
"HTTP-Request-Metadata":
```

```
{"Headers":
{
""Accept-Language":"en-US,en;q=0.9",
"Cookie":"__utma
    =256360345.2034739502.1446824378.1473178021.1487193943.13;
      _ga=GA1.2.863692975.1446056072; _gid=GA1
    .2.1651099024.1523590838; _gat_UA-2088428-1=1",
"X-Requested-With":"XMLHttpRequest",
"Accept-Encoding":"gzip, deflate",
"Referer":"http://www.cs.odu.edu/~lalkwai/publication.
    html",
"User-Agent":"Mozilla/5.0 (X11; Linux x86_64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome
    /63.0.3239.132 Safari/537.36",
"Accept":"text/plain, */*; q=0.01"},
"Headers-Length":"510",
"Entity-Length":"0",
"Entity-Trailing-Slop-Bytes":"0",
"Request-Message":
{
""Method":"GET",
"Version":"HTTP/1.1",
"Path":"/~lalkwai/publication.html"},
"Entity-Digest":"sha1:3I42H3S6NNFQ2MSVX7XZKYAYSCX5QBYJ"
    },
"Actual-Content-Type":"application/http; msgtype=
    request"}},

"Container":
{
""Compressed":false,
"Offset":"681",
"Filename":"20180413034055402.warc"}}

WARC/1.0
WARC-Type: metadata
WARC-Target-URI: http://www.cs.odu.edu/~lalkwai/
    publication.html
WARC-Date: 2018-04-13T03:40:55Z
WARC-Record-ID: <urn:uuid:6a476834-3c77-4dcc-8862-44
    e42fe72280>
```

```
WARC-Refers-To: <urn:uuid:6fef2a49-a9ba-4b40-9f4a-5
    ca5db1fd5c6>
Content-Type: application/json
Content-Length: 1433

{"Envelope":
{
""Format":"WARC",
"WARC-Header-Length":"326",
"Block-Digest":"sha1:XXDSCAMB3J5JKR6M72PCITGAZGO73CXW",
"Actual-Content-Length":"537",
"WARC-Header-Metadata":
{
""WARC-Type":"metadata",
"WARC-Date":"2018-04-13T03:40:55Z",
"Content-Length":"537",
"WARC-Record-ID":"<urn:uuid:6fef2a49-a9ba-4b40-9f4a-5
    ca5db1fd5c6>",
"WARC-Target-URI":"http://www.cs.odu.edu/~lalkwai/
    publication.html",
"WARC-Concurrent-To":"<urn:uuid:dddc4ba2-c1e1-459b-8d0d
    -a98a20b87e96>",
"Content-Type":"application/warc-fields"},
"Payload-Metadata":
{
""Trailing-Slop-Length":"4",
"WARC-Metadata-Metadata":
{
""FIELDS_CORRUPT":true,
"Trailing-Slop-Length":"0",
"Metadata-Records":[{"Name":"outlink",
"Value":"http://www.cs.odu.edu/~lalkwai/publication.
    htmlODU.jpg E =EMBED_MISC"},{"Name":"outlink",
"Value":"http://www.cs.odu.edu/~lalkwai/publication.
    htmlnull E link/@href"},{"Name":"outlink",
"Value":"http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl
    -2015-arabic-sites.pdf L a/@href"},{"Name":"outlink"
    ,
"Value":"https://dx.doi.org/10.1145/3041656 L a/@href"
    },{"Name":"outlink",
```

```
"Value":"https://arxiv.org/abs/1703.03302 L a/@href"},{
    "Name":"outlink",
"Value":"http://www.cs.odu.edu/~mkelly/papers/2017
    _jcdl_countingMementos.pdf L a/@href"},{"Name":"
    outlink",
"Value":"http://www.cs.odu.edu/~lalkwai/ L a/@href"}],
"Actual-Content-Length":"537"},
"Actual-Content-Type":"application/metadata-fields"}},

"Container":
{
""Compressed":false,
"Offset":"1532",
"Filename":"20180413034055402.warc"}}

WARC/1.0
WARC-Type: metadata
WARC-Target-URI: http://www.cs.odu.edu/~lalkwai/
    publication.html
WARC-Date: 2018-04-13T03:40:55Z
WARC-Record-ID: <urn:uuid:67c5bc7f-044c-4d35-b679-
    b1845e0c4561>
WARC-Refers-To: <urn:uuid:34a0a293-80b9-ffdc-4f77-3
    d027a08bfbd>
Content-Type: application/json
Content-Length: 2395

{"Envelope":
{
""Format":"WARC",
"WARC-Header-Length":"269",
"Block-Digest":"sha1:JLGNN3SMFT7Z2PW7KINL47CIXG35OMOZ",
"Actual-Content-Length":"2743",
"WARC-Header-Metadata":
{
""WARC-Type":"response",
"WARC-Date":"2018-04-13T03:40:55Z",
"Content-Length":"2743",
"WARC-Record-ID":"<urn:uuid:34a0a293-80b9-ffdc-4f77-3
    d027a08bfbd>",
```

```
"WARC-Target-URI":"http://www.cs.odu.edu/~lalkwai/
    publication.html",
"Content-Type":"application/http; msgtype=response"},
"Payload-Metadata":
{
""Trailing-Slop-Length":"4",
"Actual-Content-Type":"application/http; msgtype=
    response",
"HTTP-Response-Metadata":
{
""Headers":
{
""Vary":"Accept-Encoding",
"Transfer-Encoding":"chunked",
"Date":"Fri, 13 Apr 2018 03:40:34 GMT",
"Connection":"keep-alive",
"Content-Type":"text/html",
"Server":"nginx"},
"Headers-Length":"171",
"Entity-Length":"2572",
"Entity-Trailing-Slop-Bytes":"0",
"Response-Message":
{
""Status":"200",
"Version":"HTTP/1.1",
"Reason":"OK"},
"HTML-Metadata":
{
""Links":[{"alt":"Old Dominion University",
"path":"IMG@/src",
"url":"ODU.jpg"},
{"text":"Lulwah M. Alkwai, Michael L. Nelson and
    Michele C. Weigle, \"How Well Are Arabic Websites
    Archived?\",",
"path":"A@/href",
"url":"http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl
    -2015-arabic-sites.pdf"},{"text":"Lulwah M. Alkwai,
    Michael L. Nelson and Michele C. Weigle, \"Comparing
     the Archival Rate of Arabic, E",
"path":"A@/href",
```

```
    "url":"https://dx.doi.org/10.1145/3041656"},{"text":"
       Mat Kelly, Lulwah M. Alkwai, Michael L. Nelson,
       Michele C. Weigle and Herbert Van de Sompel, \"
       Impact",
    "path":"A@/href",
    "url":"https://arxiv.org/abs/1703.03302"},{"text":"Mat
       Kelly, Lulwah M. Alkwai, Sawood Alam, Michael L.
       Nelson, Michele C. Weigle and Herbert Van de So",
    "path":"A@/href",
    "url":"http://www.cs.odu.edu/~mkelly/papers/2017
       _jcdl_countingMementos.pdf"},{"text":"Back to Lulwah
        Alkwais' Old Dominion University Computer Science
       homepage",
    "path":"A@/href",
    "url":"http://www.cs.odu.edu/~lalkwai/"},{"text":"Web
       Science and Digital Libraries Research Group in the
       Department of Computer Science at Old Domini",
    "path":"A@/href",
    "url":"https://ws-dl.cs.odu.edu/"}],
    "Head":
    {
    ""Metas":[{"content":"text/html; charset=utf-8",
    "http-equiv":"Content-type"},{"content":"width=device-
       width, initial-scale=1",
    "name":"viewport"}],
    "Title":"Lulwah Alkwais' Publication list"}},

    "Entity-Digest":"sha1:LA4CFY57P7OIWQT47T7IMYNDPSDGY2IO"
       }}},

    "Container":
    {
    ""Compressed":false,
    "Offset":"2399",
    "Filename":"20180413034055402.warc"}}

    WARC/1.0
    WARC-Type: metadata
    WARC-Target-URI: http://www.cs.odu.edu/~lalkwai/ODU.jpg
    WARC-Date: 2018-04-13T03:40:55Z
```

```
WARC-Record-ID: <urn:uuid:ad998f15-5176-44dd-b9f6-
    bb8310ee4a99>
WARC-Refers-To: <urn:uuid:3c5d51d3-11e6-5161-f800-71463
    fa4a8cc>
Content-Type: application/json
Content-Length: 1368

{"Envelope":
{
""Format":"WARC",
"WARC-Header-Length":"326",
"Block-Digest":"sha1:UGBCYT3ALX627ST4JRN2DIOXVLL6NRJZ",
"Actual-Content-Length":"449",
"WARC-Header-Metadata":
{
""WARC-Type":"request",
"WARC-Date":"2018-04-13T03:40:55Z",
"Content-Length":"449",
"WARC-Record-ID":"<urn:uuid:3c5d51d3-11e6-5161-f800
    -71463fa4a8cc>",
"WARC-Target-URI":"http://www.cs.odu.edu/~lalkwai/ODU.
    jpg",
"WARC-Concurrent-To":"<urn:uuid:a9be70a0-a480-bab1
    -4676-b8dceb3c3365>",
"Content-Type":"application/http; msgtype=request"},
"Payload-Metadata":
{
""Trailing-Slop-Length":"4",
"HTTP-Request-Metadata":
{
""Headers":
{
""Accept-Language":"en-US,en;q=0.9",
"Cookie":"__utma
    =256360345.2034739502.1446824378.1473178021.1487193943.13;
     _ga=GA1.2.863692975.1446056072; _gid=GA1
    .2.1651099024.1523590838; _gat_UA-2088428-1=1",
"Accept-Encoding":"gzip, deflate",
"Referer":"http://www.cs.odu.edu/~lalkwai/publication.
    html",
```

```
"User-Agent":"Mozilla/5.0 (X11; Linux x86_64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome
    /63.0.3239.132 Safari/537.36",
"Accept":"*/*"},
"Headers-Length":"447",
"Entity-Length":"0",
"Entity-Trailing-Slop-Bytes":"0",
"Request-Message":
{
""Method":"GET",
"Version":"HTTP/1.1",
"Path":"/~lalkwai/ODU.jpg"},
"Entity-Digest":"sha1:3I42H3S6NNFQ2MSVX7XZKYAYSCX5QBYJ"
    },
"Actual-Content-Type":"application/http; msgtype=
    request"}},

"Container":
{
""Compressed":false,
"Offset":"5415",
"Filename":"20180413034055402.warc"}}

WARC/1.0
WARC-Type: metadata
WARC-Target-URI: http://www.cs.odu.edu/favicon.ico
WARC-Date: 2018-04-13T03:40:55Z
WARC-Record-ID: <urn:uuid:98f0d8e4-18a0-46f8-bdc6-813
    c807805b4>
WARC-Refers-To: <urn:uuid:cb7d3c7b-787c-9e3c-75b1-1
    f293c0b7e19>
Content-Type: application/json
Content-Length: 1394
}
```

APPENDIX B

LIST OF ABBREVIATIONS

AT	Aboutness Time
AUT	Archives Unleashed Toolkit
CW	Confidence Weighted
CD	Creation datetime
DT	Decision Trees
DMOZ	Directory.mozilla.org
FN	False Negatives
FP	False Positives
FST	Finite state transducer
κ	Fleiss' kappa
FAQ	Frequently Asked Queries
gTLD	generic top-level domain
HTTP	HyperText Transfer Protocol
IC	Information content
LM	Last modified
LS	Lexical signature
ML	Machine learning
ME	Maximum Entropy
URI-M	Memento
MD	Memento datetime
Mi-F1	Micro-average F_1
MIME	Multipurpose Internet Mail Extensions
NB	Naïve Bayes
NLTK	Natural Language Toolkit
OTMT	Off-Topic Memento Toolkit
ODP	Open directory project
URI-R	Original Resource
ccTLD	Country code top-level domain
PA	Passive-Aggressive
RDF	Resource description framework
STM	Science Technology and Medicine
SER	Search Engine Results
SURT	Sort-friendly URI Reordering Transform
SVM	Support Vector Machines

TF	Term frequency
TF-IDF	Term frequency-inverse document frequency
URI-G	TimeGate
URI-T	TimeMap
URL	Uniform Resource Locator
TLD	Top level domain
TN	True Negatives
TP	True Positives
URI	Uniform Resource Identifier
WARC	Web ARChive
WAT	Web Archive Metadata

VITA

Lulwah M. Alkwai

Department of Computer Science

Old Dominion University

Norfolk, VA 23529

EDUCATION

Ph.D.   Computer Science, Old Dominion University, 2019

M.S.   Computer Science, Old Dominion University, 2013

B.S.   Computer Science, King Saud University, Saudi Arabia, 2007

EMPLOYMENT

2007 - 2009   Teacher Assistant at University of Hail, Saudi Arabia

PUBLICATIONS

A complete list is available at `https://scholar.google.com/citations?user=EnQF73AAAAAJ&hl=en&oi=ao`

Typeset using LaTeX.