

Spring 2015

Avoiding Spoilers on Mediawiki Fan Sites Using Memento

Shawn M. Jones

Old Dominion University, jones.shawn.m@gmail.com

Follow this and additional works at: https://digitalcommons.odu.edu/computerscience_etds



Part of the [Computer Sciences Commons](#)

Recommended Citation

Jones, Shawn M.. "Avoiding Spoilers on Mediawiki Fan Sites Using Memento" (2015). Master of Science (MS), thesis, Computer Science, Old Dominion University, DOI: 10.25777/d8hw-b984

https://digitalcommons.odu.edu/computerscience_etds/1

This Thesis is brought to you for free and open access by the Computer Science at ODU Digital Commons. It has been accepted for inclusion in Computer Science Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

**AVOIDING SPOILERS ON MEDIAWIKI FAN SITES
USING MEMENTO**

by

Shawn M. Jones
B.S. May 1999, Old Dominion University

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

MASTER OF SCIENCE

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
May 2015

Approved by:

Michael L. Nelson (Director)

Michele C. Weigle (Member)

Irwin Levinstein (Member)

ABSTRACT

AVOIDING SPOILERS ON MEDIAWIKI FAN SITES USING MEMENTO

Shawn M. Jones
Old Dominion University, 2015
Director: Dr. Michael L. Nelson

A variety of fan-based wikis about episodic fiction (e.g., television shows, novels, movies) exist on the World Wide Web. These wikis provide a wealth of information about complex stories, but if readers are behind in their viewing they run the risk of encountering “spoilers” – information that gives away key plot points before the intended time of the show’s writers. Enterprising readers might browse the wiki in a web archive so as to view the page prior to a specific episode date and thereby avoid spoilers. Unfortunately, due to how web archives choose the “best” page, it is still possible to see spoilers (especially in sparse archives).

In this paper we discuss how to use Memento to avoid spoilers. Memento uses TimeGates to determine which best archived page to give back to the user, currently using a minimum distance heuristic. We quantify how this heuristic is inadequate for avoiding spoilers, analyzing data collected from fan wikis and the Internet Archive. We create an algorithm for calculating the probability of encountering a spoiler in a given wiki article. We conduct an experiment with 16 wiki sites for popular television shows. We find that 38% of those pages are unavailable in the Internet Archive. We find that when accessing fan wiki pages in the Internet Archive there is as much as a 66% chance of encountering a spoiler. Using sample access logs from the Internet Archive, we find that 19% of actual requests to the Wayback Machine for `wikia.com` pages ended in spoilers. We suggest the use of a different minimum distance heuristic, `minpast`, for wikis, using the desired datetime as an upper bound.

Finally, we highlight the use of an extension for MediaWiki that utilizes this new heuristic and can be used to avoid spoilers. An unexpected revelation about Memento comes from the development of this extension. It turns out that an optimized two request-response Memento pattern for interacting with TimeGates does not perform well with MediaWiki, leading us to fall back to the original Memento pattern of three request-response pairs. We also conduct performance testing on the extension and show that it has a minimal impact on MediaWiki’s performance.

Copyright, 2015, by Shawn M. Jones, All Rights Reserved.

ACKNOWLEDGEMENTS

This thesis would not have been possible without the assistance and care of my loving wife, Valentina Neblitt-Jones. She tolerated many hours of separation in order to ensure I accomplished this goal. She also endured many hours of discussion whereby I refined the explanation of the concepts within this document. Without her continued input and reinforcement my faith would have certainly faltered and I would not have finished. Our love of fiction and the hours of discussion it produced brought me special insight into this topic. To settle conflicts and promote these discussions, we discovered fan-based wikis such as *Lostpedia*, where I began to notice people updating fan wikis while television show episodes aired.

I would like to thank the hard work of Michael L. Nelson in listening to me come up with ideas, both good and bad, about how best to move this along. He was instrumental in ensuring that I was not only on track, but also helped me develop the visualizations that best describe the data and conclusions from the thesis. If it hadn't been for his hard work putting together a curriculum studying Web Science and Digital Libraries, I would not have found this topic of interest. His course, *Introduction to Digital Libraries*, forced me to further process data from *Lostpedia*, where I started to see the trends in wiki usage that led me to propose hypotheses that led to this thesis.

Very important to this work were both Herbert Van de Sompel and Harihar Shankar at Los Alamos National Laboratory, without whom I would have not had the opportunity to work with Wikipedia and find such an interesting intersection of concerns to research. Both provided support on the technical aspects of wikis as well as providing feedback on the technical report that became Chapter 8 of this thesis.

Without the sponsorship and assistance of Irwin Levinstein, Ravi Mukkamala, Michele Weigle, Mohammad Zubair, and Scott Ainsworth, I would not have been able to get through the process of not only entering the graduate program at Old Dominion University, but also the process of successfully finishing a Master's Degree. Levinstein and Ainsworth wrote my recommendation letters for entrance and encouraged me to enter the program. Mukkamala, Weigle, and Zubair provided the guidance and legwork necessary to ensure that I was able to stay on top of the university paperwork needed to remain a student and also finish this degree. Mukkamala's course on data mining was key to examining the data collected as well as understanding the

techniques used in many of the references I read. Weigle's course on networking was key to discussing the networking concepts in Chapter 8.

This thesis, and my entire Master's Degree, would not have been possible without the assistance of the management from SPAWAR Systems Center Norfolk, who were nice enough to allow me to use my leave and take time off from work to complete my academic work.

The encouragement of my friends Michael Olson, Kara Olson, and Richard Hughes were appreciated as they withstood my hours explanation on these concepts, providing feedback while also ensuring that I did not waver in my commitment to complete this work.

The work would not have been possible without the Old Dominion University Perry Library and its staff. I utilized not only their collection, but also their facilities as an exceptional work area surrounded by resources important to bringing this document to life.

Finally, the input and encouragement of the ODU Computer Science WS-DL team was crucial to my success; including Justin Brunelle, Hany SalahEldeen, Yasmin AlNoamany, Ahmed AlSum, Mat Kelly, and Sawood Alam.

This work would not have been possible without funding by the Andrew Mellon Foundation.

TABLE OF CONTENTS

	Page
LIST OF TABLES	ix
LIST OF FIGURES	xvii
Chapter	
1. MOTIVATION - WARNING, THIS THESIS MAY CONTAIN SPOILERS .	1
2. BACKGROUND	10
2.1 HYPERTEXT MARKUP LANGUAGE (HTML)	10
2.2 THE WORLD WIDE WEB	12
2.3 MEMENTO	20
2.4 WEB ARCHIVING AND THE WAYBACK MACHINE	26
2.5 WIKIS, WEB ARCHIVES, AND MEMENTO	30
2.6 THE NAÏVE SPOILER CONCEPT AND BRINGING IT ALL TO- GETHER	35
3. RELATED WORK	39
3.1 EXISTING STUDIES ON SPOILERS	39
3.2 NOTICES, BLURRING TEXT, AND OTHER TECHNICAL AT- TEMPTS AT SPOILER MANAGEMENT	42
3.3 EXISTING SOFTWARE THAT ATTEMPTS TO HELP USERS AVOID SPOILERS	46
3.4 EXISTING STUDIES OF WIKIS	53
3.5 EXISTING SOFTWARE THAT PROVIDES PAST VERSIONS OF MEDIAWIKI PAGES	54
3.6 SUMMARY	57
4. SURVEY OF TIMEGATE HEURISTICS	60
4.1 GENERIC TIMEGATE HEURISTIC CONSIDERATIONS	60
4.2 TIMEGATE HEURISTICS UNDER CONSIDERATION	62
4.3 CONCLUSIONS FOR AVOIDING SPOILERS	72
5. SPOILER AREAS CREATED BY MINDIST	76
5.1 SPOILER AREAS CREATED BY MINDIST HEURISTICS	76
5.2 CONSIDERATIONS FOR MULTIPLE EVENTS AND AGGRE- GATING SPOILER AREAS	87
5.3 SUMMARY	88

6. MEASURING SPOILER PROBABILITY IN POPULAR WIKIS.....	90
6.1 STRUCTURE OF THE EXPERIMENT	90
6.2 RESULTS	91
6.3 CONCLUSIONS	105
7. MEASURING ACTUAL SPOILERS IN WAYBACK MACHINE LOGS	107
7.1 OUR METHOD FOR ANALYZING THE LOGS	107
7.2 SUMMARY	111
8. PREVENTING SPOILERS WITH THE MEMENTO MEDIAWIKI EXTEN- SION	112
8.1 DESIGN AND ARCHITECTURE	112
8.2 PERFORMANCE IMPACT ON MEDIAWIKI INSTALLATIONS . .	129
8.3 ATTEMPTS AT TEMPORAL COHERENCE	133
8.4 SUMMARY	141
9. FUTURE WORK	143
10. CONCLUSIONS	145
REFERENCES	147
APPENDICES	
A. SPOILER AREA VISUALIZATIONS	158
B. SPOILER PROBABILITY HISTOGRAMS	167
C. SPOILER PROBABILITY CUMULATIVE DISTRIBUTION FUNCTION .	183
VITA.....	199

LIST OF TABLES

Table	Page
1	Some example HTTP request methods 15
2	Some example HTTP response status codes 16
3	Dimensions of content negotiation 19
4	Memento Resource Types 20
5	Some examples of wikitext compared to HTML 30
6	Notation used in this thesis 38
7	Summary of TimeGate Heuristics 74
8	Conditions for relationships between episodes denoted by e , revisions denoted by r , mementos denoted by m , and a midpoint between mementos denoted by h . Mementos m_1 and m_n denote first and last mementos, respectively. 89
9	Fan wikis used in the spoiler areas experiment 91
10	Information required to determine if spoilers can be encountered if mindist is used 92
11	Spoiler probabilities for most popular pages within each fan wiki 100
12	Statistics for each fan wiki 103
13	Specifications of the Test Machine Used to Process the Wayback Machine Logs 111
14	Version 2.0 Memento MediaWiki Extension MementoResource Class Family Members Mapped To Their Memento Resource Type 118
15	Examples of TimeMap URIs From the Memento MediaWiki Extension 119
16	Examples of Memento Resources From the Memento MediaWiki Extension 120
17	Specifications of the Test Machine Used to Compare Pattern 1.1 vs. Pattern 2.1 URI-G Performance 122

18	Statistics on Pattern 1.1 vs. Pattern 2.1 TimeGate testing results	127
19	Status of full temporal coherence among MediaWiki Entities	133

LIST OF FIGURES

Figure	Page
1 Kumar hesitates to use the wiki for <i>Once Upon A Time</i> because he has not seen the latest episode	2
2 Maurice finds that he cannot avoid information on <i>A Dance With Dragons</i>	3
3 Kim finds out that character James Novak from <i>Scandal</i> is dead	4
4 Spoiler notice for <i>Downton Abbey Wiki</i>	4
5 Demonstration of Spoiler Shield blocking Facebook posts about the TV series <i>Game of Thrones</i>	5
6 Example of a wiki history page for the article on <i>Abraham Lincoln</i>	6
7 A screen shot of a specific memento in the Wayback Machine of http://lostpedia.wikia.com from February 14, 2014	6
8 Rendering of HTML from Listing 2.1	11
9 Relationship between URIs, Resources, and Representations	13
10 Example of HTTP request-response process	13
11 An example of a web browser, the most common user agent type for the World Wide Web	14
12 HTTP request-response examples	17
13 Visualization of mementos captured for a given resource at times t_1 , t_2 , t_3 , and t_4	23
14 General Memento pattern	25
15 Screenshot of the Memento Time Travel Chrome Extension, a Memento client	26
16 Architecture for a simple web crawler	27
17 Wayback Machine Screenshots	29
18 Example Wiki Page	31

19	Example Edit Page for a Wiki Article	32
20	Example History Page for a Wiki Article	32
21	Example of viewing an earlier revision of a Wiki Article	33
22	Example of a revision notice, present at the top of old revisions in MediaWiki	33
23	Example of a wiki page viewed from the Wayback Machine	33
24	Example Timeline Showing Captured Mementos of Wiki Edits	34
25	Each event can inspire a new wiki revision which may be captured as a memento by a web archive	36
26	Representation of a Naïve Spoiler Concept	37
27	Results of Leavitt and Christenfeld’s spoilers research, indicating a slight preference for spoiled stories over unspoiled stories. (Error bars represent standard errors)	40
28	Examples of Spoiler Notices on the Web	43
29	Guidance for wiki editors for the site <i>A Wiki of Ice and Fire</i> , indicating that they should not include plot details for an upcoming book, avoiding the addition of spoilers to existing pages	44
30	<i>TV Tropes</i> web site examples of spoiler text shown as white text on white background for the television show <i>The Office</i>	44
31	Demonstration web page for the Spoiler Alert JavaScript library, showing blurred text and images instead of spoiler information	45
32	Examples of Tumblr Savior	47
33	Examples of configuration screens for social media filter programs that can be used to block spoilers entirely	48
34	Spoiler Shield For Chrome posts about <i>Game of Thrones</i> on Facebook	49
35	Screenshots of Spoiler Shield configuration screens	50
36	Screenshots of the TweetDeck application	51
37	Screenshots of two Spoiler Foiler web applications created by Netflix	52
38	High level process for the use of the Memento Wikipedia Proxy	55

39	The operations screen for the MediaWiki Time Machine extension	56
40	Conception Diagram of the Parsoid MediaWiki application (image created by J. D. Forrester, Gabriel Wicke, and Trevor Parscal)	57
41	The use and products of the MediaWiki <i>Collection</i> extension	59
42	Demonstration of the <i>mindist</i> heuristic, in this case $m_2@t_7$ is chosen because it is closest to t_a	63
43	Demonstration of the <i>mindist</i> heuristic; in this case $m_3@t_{10}$ is chosen because it is closest to t_a	63
44	Demonstration of the <i>minpast</i> heuristic, in this case $m_2@t_7$ is chosen because it is closest, but not greater than, t_a	65
45	Demonstration of the <i>minpast</i> heuristic, in this case $m_2@t_7$ is still chosen because it is closest, but not greater than, t_a , even through $m_3@t_{10}$ has the minimum distance	65
46	Demonstration of the <i>minfutur</i> heuristic, in this case $m_3@t_{10}$ is chosen because it is closest, but not less than, t_a	67
47	Demonstration of the <i>minfutur</i> heuristic, in this case $m_3@t_{10}$ is still chosen because it is closest, but not less than, t_a , even through $m_3@t_7$ has the minimum distance	67
48	Example of <i>pre-archive spoiler areas</i> (shown in light red) created using the <i>mindist</i> heuristic; the overlap of the spoiler areas for episodes e_3 and e_2 is shown in darker red.	77
49	Example of a <i>archive-extant spoiler area</i> (shown in light red) created by using the <i>mindist</i> heuristic, h is the midpoint between m_{k-1} and m_k	79
50	Example of the condition: <i>Archive-Extant Safe HRE</i> for event e_i	81
51	Example of the condition: <i>Archive-Extant Safe RHE</i> for episode e_i	81
52	Example of the condition: <i>Archive-Extant Safe EHR</i> for event e_i	82
53	Example of the condition: <i>Archive-Extant Safe ERH</i> for event e_i	82
54	Example of the condition: <i>Archive-Extant Safe REH</i> for event e_i	84
55	Example of the condition: <i>Pre-Archive Safe</i> for event e_3 ; spoiler area exists for event e_2 , but not e_3	84

56	Example of the condition: <i>Post-Archive Safe ER</i> for event e_i	85
57	Example of the condition: <i>Post-Archive Safe RE</i> for event e_i	85
58	Example of a potential spoiler zone, stretching from t_{e_1} to t_{e_n}	86
59	Example of a spoiler area (light red area) for episode e_i inside potential spoiler zone (dotted red rectangle), stretching from t_{e_1} to t_{e_n}	86
60	Example export page for a MediaWiki installation ¹	92
61	Timelines for the wiki sites used in this experiment: top timeline represents the length of the episode run, middle timeline represents the life of the wiki, bottom timeline represents the span of time the Internet Archive has operated on the site	94
62	Spoiler areas for the most popular page in <i>Lostpedia</i> (3,531 revisions) ² . . .	98
63	Spoiler areas for the most popular page in the <i>Game of Thrones Wiki</i> (768 revisions) ³	98
64	Spoiler areas for the most popular page in the <i>House of Cards Wiki</i> (380 revisions) ⁴	99
65	Histogram of spoiler probabilities for <i>Lostpedia</i>	101
66	Histogram of spoiler probabilities for <i>Game of Thrones Wiki</i>	102
67	Histogram of spoiler probabilities for <i>House of Cards Wiki</i>	102
68	Histogram of spoiler probabilities for all pages in study	104
69	Graph of the cumulative distribution function of spoiler probabilities for all 16 wiki sites	104
70	Plot of missed updates for 16 wiki sites over time, lighter colors indicate few to no missed updates, darker colors indicate many missed updates . . .	106
71	Plot of redundant mementos for 16 wiki sites over time, lighter colors indicate few to no redundant mementos, darker colors indicate many redundant mementos	106
72	URI-M pattern for the Wayback Machine and Internet Archive	108
73	Memento MediaWiki Extension Class Hierarchy Diagram	113

74	Memento Pattern 2.1 Overview with Only Salient Headers, Methods, and Responses	115
75	Memento Pattern 1.1 Overview with Only Salient Headers, Methods, and Responses	121
76	Differences in URI-G performance between Pattern 1.1 and 2.1	123
77	Histogram showing Pattern 1.1 values	124
78	Histogram showing Pattern 2.1 values	125
79	Plot showing the difference in times for URI-Rs between a MediaWiki installation containing our extension vs one without it installed	130
80	Plot showing the difference in times for URI-Ms between a MediaWiki installation containing our extension vs one without it installed	131
81	Plot showing the difference in size between MediaWiki history pages and TimeMaps for the same article	132
82	Example Wikipedia page ⁵ with an embedded image that has been changed as the page content changes	134
83	June 5, 2013 version of the example MediaWiki page ⁶ with an embedded image that is changed as the page content changes (note that the map is the same as in Figure 82, which does not match the article text)	135
84	June 5, 2013 version of the example MediaWiki page should show this map ⁷ instead if it is to be consistent with the article content.....	136
85	MediaWiki Page ⁸ showing the map's file history	137
86	Example of CSS history ⁹ in MediaWiki	140
87	Example of JavaScript history in MediaWiki ¹⁰	141
88	Example of the current CSS not agreeing with an previous revision of a MediaWiki page	142
89	Spoiler areas for the most popular page in <i>Lostpedia</i> (3,531 revisions) ¹¹	158
90	Spoiler areas for the page in <i>the Big Bang Theory Wiki</i> that contains the most revisions ¹²	159

91	Spoiler areas for the page in the <i>Boardwalk Empire Wiki</i> that contains the most revisions ¹³	159
92	Spoiler areas for the page in the <i>Breaking Bad Wiki</i> that contains the most revisions ¹⁴	160
93	Spoiler areas for the page in the <i>Continuum Wiki</i> that contains the most revisions ¹⁵	160
94	Spoiler areas for the page in the <i>Downton Abbey Wiki</i> that contains the most revisions ¹⁶	161
95	Spoiler areas for the most popular page in the <i>Game of Thrones Wiki</i> (768 revisions) ¹⁷	161
96	Spoiler areas for the page in the <i>Grimm Wiki</i> that contains the most revisions ¹⁸	162
97	Spoiler areas for the most popular page in the <i>House of Cards Wiki</i> (380 revisions) ¹⁹	162
98	Spoiler areas for the page in the <i>How I Met Your Mother Wiki</i> that contains the most revisions ²⁰	163
99	Spoiler areas for the page in the <i>Mad Men Wiki</i> that contains the most revisions ²¹	163
100	Spoiler areas for the page in the <i>NCIS Database</i> that contains the most revisions ²²	164
101	Spoiler areas for the page in the <i>Once Upon A Time Wiki</i> that contains the most revisions ²³	164
102	Spoiler areas for the page in the <i>Scandal Wiki</i> that contains the most revisions ²⁴	165
103	Spoiler areas for the page in the <i>True Blood Wiki</i> that contains the most revisions ²⁵	165
104	Spoiler areas for the page in the <i>White Collar Wiki</i> that contains the most revisions ²⁶	166
105	Big Bang Theory Wiki	167
106	Boardwalk Empire Wiki	168
107	Breaking Bad Wiki	169

108	Continuum Wiki	170
109	Downton Abbey Wiki	171
110	Game of Thrones Wiki	172
111	Grimm Wiki	173
112	House of Cards Wiki	174
113	How I Met Your Mother Wiki	175
114	Lostpedia	176
115	Mad Men Wiki	177
116	NCIS Database.....	178
117	Once Upon A Time Wiki	179
118	Scandal Wiki	180
119	True Blood Wiki	181
120	White Collar Wiki	182
121	Big Bang Theory	183
122	Boardwalk Empire	184
123	Breaking Bad	185
124	Continuum	186
125	Downton Abbey	187
126	Game of Thrones	188
127	Grimm	189
128	House of Cards	190
129	How I Met Your Mother	191
130	Lostpedia	192
131	Mad Men.....	193

132	NCIS	194
133	Once Upon A Time	195
134	Scandal	196
135	True Blood	197
136	White Collar	198

CHAPTER 1

MOTIVATION - WARNING, THIS THESIS MAY CONTAIN SPOILERS

Introducing Pramiti, a young woman who enjoys a variety of fiction. Pramiti does not merely watch or read her fiction. She also engages more directly in review and discussion with her friends and colleagues. One day, she is discussing information about *Once Upon a Time*, a popular television show, with her coworkers, and they wish to settle a dispute about a character using the fan-created *Once Upon a Time Wiki* [108] resource on the World Wide Web.

Unfortunately, her coworker Kumar has not yet watched the most recent episode. Kumar wants to access the information about the character in order to continue the conversation, using a resource as shown in Figure 1, but does not wish to encounter any information that will ruin the episode he has not seen.

How can Kumar avoid these *spoilers* while still acquiring useful information for the conversation from this resource?

Consider the case of Maurice, a long time fan of the book series *A Song of Ice and Fire*. He finished the fourth book years ago, and is about to read the fifth, *A Dance with Dragons*, released in 2011. Maurice wants to use the fan wiki site *A Wiki of Ice and Fire* [30] to review information about some of the characters so he can prepare himself for the next book. Unfortunately, it will be difficult to access this resource without encountering information from the fifth book, as shown in Figure 2.

How does Maurice improve his knowledge of this fictional world without encountering the *spoilers* on the current version of this wiki site?

Kim watches the television show *Scandal*. Her mother has been ill, so she has not had a chance to watch the full third season of the show. She uses her tablet to visit the fan wiki [109] for the television show *Scandal*, looking up information about the character *James Novak*. Without even reading the article, she sees in the information box to the right that this character has died. Now her experience with the show is ruined. She cannot unlearn the information she sees in Figure 3.

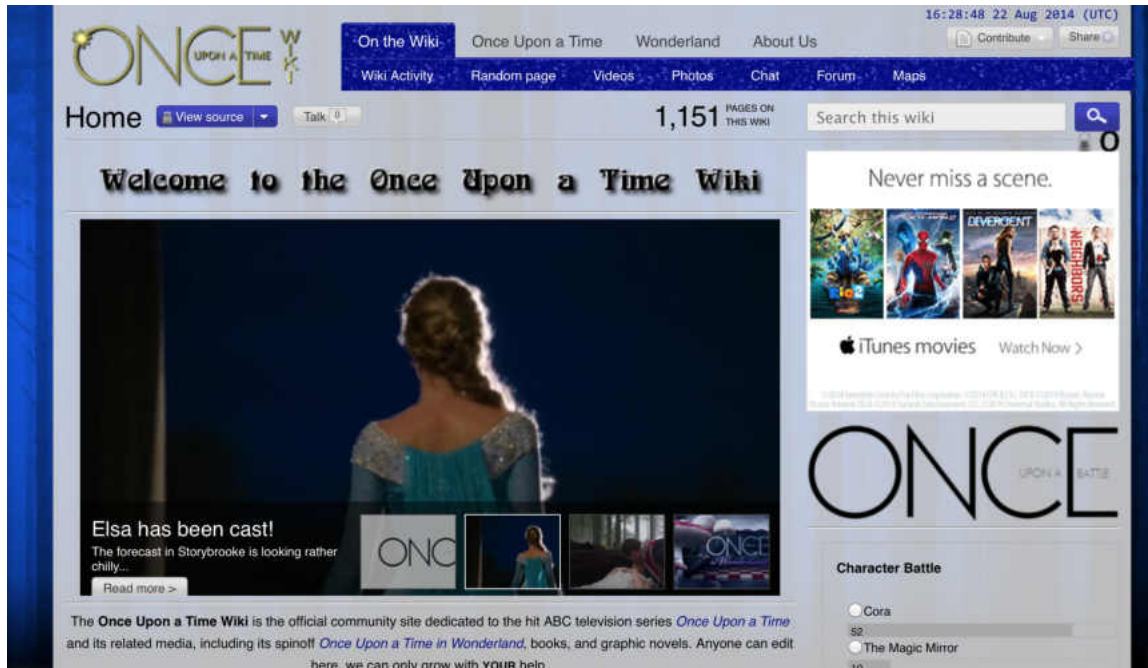


FIG. 1: Kumar hesitates to use the wiki for *Once Upon A Time* because he has not seen the latest episode

Finally, David lives in the United States and enjoys the show *Downton Abbey*. This show premieres first in the United Kingdom and is later exported to the United States. David would like to use the Downton Abbey Wiki [107] to settle a dispute among his friends. Unfortunately for David, he cannot use this wiki in its current form, because the current season of Downton Abbey is showing in the United Kingdom, and fans in that country are updating it with current information from the show. David is warned about using the web site by the large notice shown in Figure 4.

All four of these individuals want to avoid *spoilers* on the Web. Spoilers are defined as pieces of information that user wants to control the time and place of their consumption, preferring to consume them in the order that the author (or director) intended. If these pieces of information are delivered in the wrong order, enjoyment about a movie or television program is destroyed [42].

This issue is not something merely affecting a small segment of the population. *CNN* recently reported the growing issue of spoilers in social media [33]. The *New York Times* reported that Wikipedia has given up trying to protect its visitors from

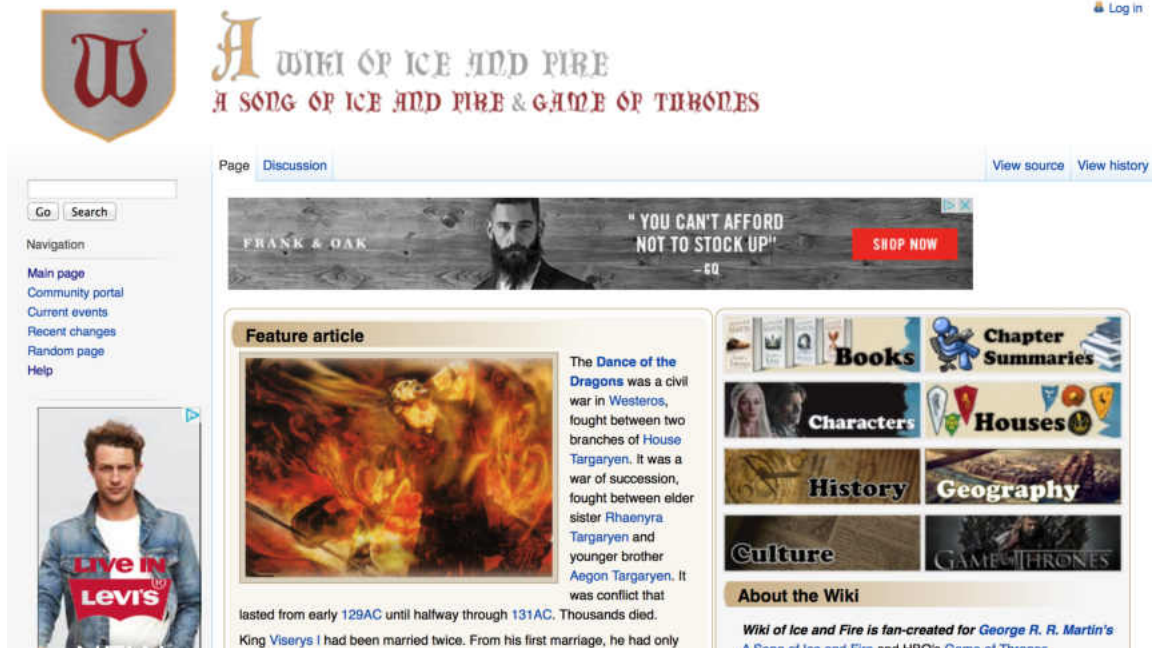


FIG. 2: Maurice finds that he cannot avoid information on
A Dance With Dragons

spoilers, despite public outcry [15].

Using examples from *Lost* and *Battlestar Galactica*, *Wired* discusses the *War of the Spoilers*, emphasizing how fans of many television prefer to view their content as the creators intended, rather than having it ruined by parts of the Web [34].

Spoilers are also controversial, as *Wired* again discusses in a poll attempting to determine the definition of spoilers as well as the etiquette surrounding them. The late Roger Ebert implored fellow movie critics to not share spoilers in their reviews, seeing as many moviegoers read reviews to determine if a movie is worth watching [19].

An academic study undertaken by the University of California, San Diego determined that there is a perceived, if not actual, harm to the enjoyment of fiction if spoilers are known [54].

Spoilers are such a problem in social media, that Boyd-Graber [14] evaluated machine learning algorithms to detect spoilers so people can avoid social media posts containing them, resulting in improvements over text-only searches. Golbeck [32] attempted to target just Twitter, in an attempt to remove all Tweets on a particular

James Novak was [Cyrus Beene's](#) journalist husband. He officially died in a carjacking but truthfully [B-613 Command Jake Ballard](#) executed James for knowing too much about the death of [Daniel Douglas Langston](#).

History

When James met Cyrus he was a reporter following the Grant campaign. At the time Cyrus wasn't ready to come out of the closet; after getting into the [The White House](#) Cyrus proposed to James promising him a house, a garden and a baby. ([Happy Birthday, Mr. President](#))

Because Cyrus hadn't held up his end of the bargain - giving James a baby - James decided to go back to work. His old boss at *The DC Times* offered him his old job; James became the "Chief White House Correspondent for *The DC Times*. ([All Roads Lead to Fitz](#))

When Cyrus learned that James was getting too close to learning the truth about Defiance with the assistance of [David Rosen](#) he procured a baby girl for James in order to get him to stop working. James gave David the Cytron memory card, knowing it was no longer safe at his place and quit his job. ([One For the Dog](#))

James Novak

Portrayed by [Dan Bucatinsky](#)



Deceased Character

Biographical Information

Status:	Deceased
Episode of Death:	Kiss Kiss Bang Bang
Cause of Death:	Shot in the chest
Residence:	Georgetown, Washington, D.C.
Aliases:	<ul style="list-style-type: none"> • Publius
Current Occupation/s:	<ul style="list-style-type: none"> • WH Press Secretary

FIG. 3: Kim finds out that character James Novak from *Scandal* is dead



On the Wiki
Downton Abbey
Behind the scenes

Wiki Activity
Random page
Videos
Photos

Home
View source
Talk (31)

336 PAGES ON THIS WIKI



TO EVERYONE WHO WILL NOT RECEIVE SERIES 4 OF DOWNTON ABBEY UNTIL JANUARY 2014, THIS IS A **WARNING**: THIS WIKI WILL BE UPDATED AND EDITED AS THE EPISODES BROADCAST IN THE UNITED KINGDOM (SEPTEMBER - DECEMBER 2013)! PROCEED AT YOUR OWN RISK! **THERE WILL BE SPOILERS ABUNDANT ON MOST OF THE PAGES!** YOU HAVE BEEN WARNED!

FIG. 4: Spoiler notice for *Downton Abbey Wiki*

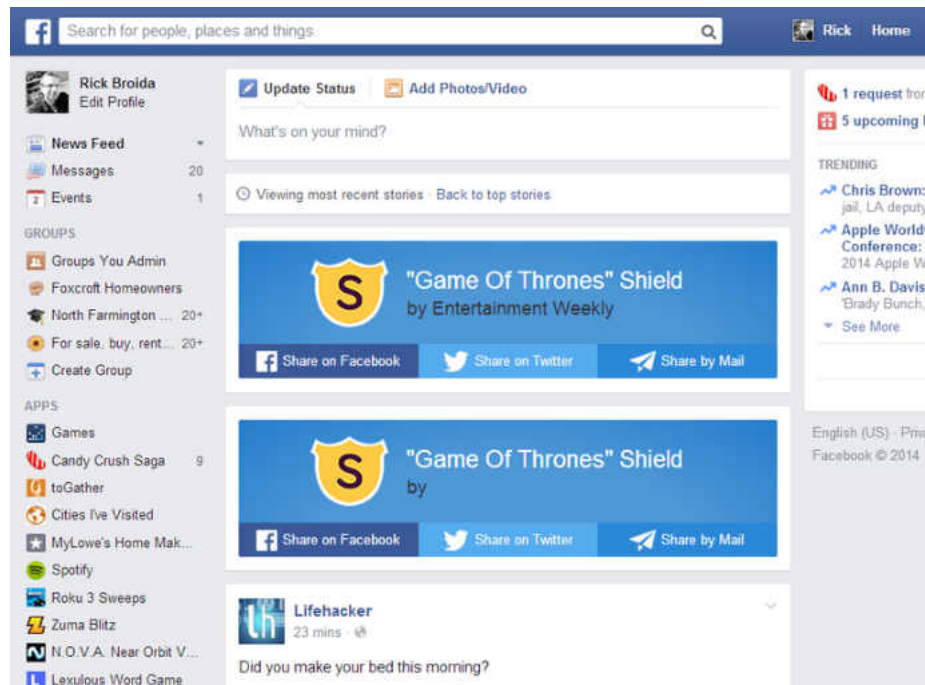


FIG. 5: Demonstration of Spoiler Shield blocking Facebook posts about the TV series *Game of Thrones*

topic, with a rate of false positives that may be unacceptable to users. Commercially, apps now exist on the market to prevent users from viewing spoilers in Twitter feeds, Facebook walls, and Tumblr dashboards [40]. One example is Spoiler Shield, shown in action in Figure 5.

Kumar, Maurice, Kim, and David all used fan-based wikis, which are on the rise on the Web [70]. In addition to Wikipedia, Wikia exists as the largest fan-based wiki site containing over 400,000 communities creating information about a variety of topics, focusing on entertainment [106]. Wikis, as shown in Figure 6, have access to the past revision of every page. We show, that in the case of wikis, spoilers can be avoided altogether, without resorting to simple text-matching algorithms or machine learning.

An academic study by Boyd-Graber, Glasgow, and Zajac indicated that spoilers refer to events “*later* than the viewer’s knowledge of the current work” [14]. If the viewer’s knowledge has not caught up to the present, what if they could view a page at the time of their knowledge. What if Kumar, Maurice, Kim, and David could

Abraham Lincoln: Revision history

[View logs for this page](#)

Browse history

From year (and earlier): From month (and earlier): Tag filter:

For any version listed below, click on its date to view it. For more help, see [Help:Page history](#) and [Help:Edit summary](#).
 External tools: [Revision history statistics](#) · [Revision history search](#) · [Contributors](#) · [Edits by user](#) · [Number of watchers](#) · [Page view statistics](#)

(cur) = difference from current version, (prev) = difference from preceding version,
 m = minor edit, → = section edit, + = automatic edit summary
 (newest | oldest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)

- [\(cur | prev\)](#) 14:09, 10 May 2014 John Paul Parks (talk | contribs) .. (156,487 bytes) (-2) .. (→*Early life: corrected noun-verb error. "Loan" is a noun. "Lend" is a verb.*)
- [\(cur | prev\)](#) 19:00, 7 May 2014 Markiemagne (talk | contribs) m .. (156,489 bytes) (+4) .. (*I added a link to the Wikipedia article on the Trent Affair*)
- [\(cur | prev\)](#) 16:07, 6 May 2014 Lockesdonkey (talk | contribs) .. (156,485 bytes) (+96) .. (→*Slavery and a "House Divided"*)
- [\(cur | prev\)](#) 17:41, 5 May 2014 Jamiri (talk | contribs) m .. (156,389 bytes) (0) .. (→*Marriage and children*)
- [\(cur | prev\)](#) 17:39, 5 May 2014 Jamiri (talk | contribs) .. (156,389 bytes) (-2) .. (→*Marriage and children*)
- [\(cur | prev\)](#) 15:22, 5 May 2014 Stemoc (talk | contribs) m .. (156,391 bytes) (+1) .. (*(Script) File renamed: File:Lincolns Gettysburg Address, Gettysburg.JPG → File:Lincoln's Gettysburg Address, Gettysburg.jpg File renaming criterion #5: Correct obvious errors in file names (e.g. l...)*)
- [\(cur | prev\)](#) 08:36, 4 May 2014 Telpardec (talk | contribs) .. (156,390 bytes) (-325) .. (*trim google bloat from URL - fix quote with ellipsis, middle sentence not included*)

FIG. 6: Example of a wiki history page for the article on *Abraham Lincoln*

INTERNET ARCHIVE
Wayback Machine

http://lostpedia.wikia.com/wiki/Main_Page

203 captures
22 Dec 08 - 23 Jul 14

JAN FEB APR
14
2013 2014 2015

wikia Video Games - Entertainment - Lifestyle - Log in - Sign up

LOSTPEDIA
THE LOST ENCYCLOPEDIA

Search this wiki

On the Wiki Characters Episodes Features Community

Wiki Activity Random page Videos Photos Chat

Home

7,312 PAGES ON THIS WIKI

Welcome to **Lostpedia**
 The Lost Encyclopedia
 A Place That You All Made Together
 Currently 7,312 articles dedicated to ABC's hit TV show *Lost*

[Create an account](#)

Entertainment Video Games Lifestyle

FIG. 7: A screen shot of a specific memento in the Wayback Machine of <http://lostpedia.wikia.com> from February 14, 2014

browse the web (and their chosen fan wiki), at a time prior to the episode containing the spoilers? They would be able to view the information they wanted without the information that could ruin the enjoyment of their fiction.

Memento [101], a protocol used to view past versions of web pages, holds the answer. Memento uses special web resources named TimeGates to provide these past versions of web pages, called **mementos**, an example of which is shown in Figure 7. The Memento protocol is an extension to the Hypertext Transfer Protocol (HTTP) that allows a user to specify a web site and a time and then redirects the user to the best past version of that page based on the time given by the user. It provides seamless access to all places where previous versions of web pages are stored, from web archives to content management systems.

For example, if we, like Maurice above, had not read the book *A Dance With Dragons*, and wanted to avoid spoilers in the web site *A Wiki of Ice and Fire*, then we could use Memento to avoid spoilers on that web site. We could determine the release date of *A Dance With Dragons*. Then we select a time prior to that date and use a Memento-enabled client, such as Memento for Chrome [75] or Mink [49], to browse the web site as it looked on the date prior to the release of the novel, avoiding spoilers.

The Wayback Machine exists as an alternative, allowing users to visit past versions of web pages, but we will show how it is not sufficient to solve the spoiler problem.

An analysis of the Wayback Machine algorithm and its logs reveals an important implication for pop culture enthusiasts: the Wayback Machine and standard Memento TimeGates are insufficient for reliably avoiding spoilers in wikis, and a better solution exists in the wiki's own history function combined with an improved Memento TimeGate algorithm.

In this thesis we discuss a method to avoid spoilers on the Web, specifically focusing on fan-based wikis. We show that Memento can be used to successfully, but not reliably, avoid spoilers in general web pages. We also show that the current heuristic used by the Wayback Machine, which we label *mindist* for *minimum distance*, is insufficient for reliably avoiding spoilers, with evidence that users can and are getting spoilers with this heuristic. We offer a solution for wikis, using a different heuristic, which we call *minpast* because it provides the memento with the minimum distance without going over the datetime specified by the user. Finally, we detail an implementation using this new heuristic in the form of an extension to MediaWiki.

This thesis has 10 chapters. In Chapter 2, we explore the technologies and concepts required to not only understand the spoiler problem, but also solve it for fan-based wikis. We provide an overview of HTML, HTTP, Memento, web archiving, the Wayback Machine, and how we are using these all to avoid spoilers.

In Chapter 3, we discuss what others have done to address the spoiler problem. We will discover what issues have existed and how they can be addressed as well as reaffirming the boundary of our thesis to just focus on wikis. We briefly discuss what other work has been done on wikis and how our contribution is different.

In Chapter 4, we survey heuristics for Memento TimeGates, the resources used to bring us past versions of web pages. We will see how, out of all of the heuristics discussed, the minpast heuristic is best for avoiding spoilers. Unfortunately, mindist is the one used by most TimeGates and web archives. Mindist delivers the past web page that is closest to the datetime we have requested, where minpast will not go over that datetime.

In Chapter 5, we compare wiki revisions to web archive mementos and show just how, even while using mindist, it is probable to get a spoiler even if one chooses a datetime prior to that spoiler.

In Chapter 6, a series of experiments were conducted on actual wikis to determine the probability of acquiring a spoiler if one used Memento with the mindist heuristic.

In Chapter 7, we use logs from the Wayback Machine to show that the mindist heuristic is actually bringing users to pages from dates in the future of what they had originally specified.

In Chapter 8, we outline a solution to the problem in the form of a MediaWiki Extension that uses the minpast heuristic to deliver previous revisions of web pages to users, safely allowing them to avoid spoilers. We also show how this tool has a minimal impact on performance in MediaWiki installations and can also be used to partially solve the problem of **temporal coherence**, where embedded images and other web page resources do not match the content of the web page they are embedded in due to web archiving issues.

In Chapter 9, we discuss future work for this analysis of algorithms, indicating that mindist and minpast can be explored in other ways. We also discuss how some of the other algorithms discussed in Chapter 4 can be used for future research.

In Chapter 10, we conclude by tying all of these concepts together to show that minpast is the best algorithm for avoiding spoilers and that solutions for wikis are

the best utilization of this algorithm to date.

CHAPTER 2

BACKGROUND

In this section, we discuss which tools and concepts may be combined and improved to address the spoiler problem, specifically for wikis on the World Wide Web.

We will discuss Hypertext Markup Language (HTML), the principal document format of the Web. HTML documents link to each other, forming much of the web, including our fan-based spoiler web sites. From here we will transition to the larger World Wide Web, discussing how HTML web pages and other documents refer to one another. During that discussion we will also briefly cover HTTP, the network protocol used to move web pages from web servers to users, which is necessary to understand Memento, a form of web time travel. We will show how Memento can be used to view past versions of web pages, most of which are stored in web archives. Then we move on to discuss wikis and how web archives and Memento currently work with wikis. Finally, we tie it all together by showing that Memento can be used to view the past versions of wiki pages on fan sites, thus helping fans avoid spoilers on wiki web sites.

2.1 HYPERTEXT MARKUP LANGUAGE (HTML)

The documentation format used by the fan wikis we are interested in (as well as most of the web) is **Hypertext Markup Language (HTML)**. A document constructed of HTML is referred to as a **web page**. A collection of web pages (and other supporting files) is called a **web site**.

HTML is important to our research because it is one of the ways the Web is linked together. Listing 2.1 shows an example web page. In this example we see a series of **tags** which start with the symbol `<` and end with `>` (which are typically referred to as **angle brackets**) [8]. Using these angle brackets, the actual text of the page can be separated from the organization of the page. Tags contain text, like the `<p>` tag in our example displays all text between `<p>` and `</p>`. The trailing slash indicates when the paragraph has ended. The same situation exists for `<html>` indicating the start of the web page and `</html>` indicating the end of the web page. All

Listing 2.1: An example Web Page

```

1 <html>
2 <head>
3 <title>Example Web Page</title>
4 </head>
5 <body>
6 <p>
7 Here is an example web page, with a link to <a href="http://en.wikipedia.org/
   wiki">Wikipedia</a>.
8 </p>
9 <p>
10 The symbol for Wikipedia is /
11 </body>
12 </html>

```

Here is an example web page, with a link to [Wikipedia](http://en.wikipedia.org/wiki/).



FIG. 8: Rendering of HTML from Listing 2.1

text contained within `<html>` and `</html>` is included in the web page. Tags can appear inside other tags, like `<p>` appears inside `<html>`. Some tags also do not necessarily need an end tag, such as the `` tag on line 10. Tags can also have **attributes** such as the `src` attribute on line 10. This way the behavior of a tag can be influenced by the attribute. There are many tags for defining the structure and content of a web page. Only a few are of interest to our thesis.

The `<a>` tag, shown in action on line 7, wraps the text *Wikipedia* and makes it blue. This makes the text *Wikipedia* a **hyperlink**, allowing one to follow the text to another document, referred to by the `href` attribute of `<a>`. This is how web pages are able to refer to other web pages. The `` tag, shown on line 10, embeds an

external image resource in the page.

The web sites that we visit to read information on our favorite characters are sent to the user as web pages, but to understand the solution to the spoiler problem further, we must understand the overall World Wide Web.

2.2 THE WORLD WIDE WEB

On the World Wide Web, there exist items of interest which are referred to as **resources** [11]. The concept of resources is intentionally flexible to allow for new technologies, but for the sake of this discussion, we will define a resource as anything that can be identified by a **Uniform Resource Identifier (URI)** [12], such as `http://buffy.wikia.com/wiki/`.

Many are familiar with Uniform Resource Locators (URLs), which are a subset of URIs. URLs specify a location identifying where a resource may be found. URIs are a more generic identifier, not necessarily requiring a location or any infrastructure. Conceivably, URIs may also be used for the identification of objects in the real world (such as “non-information resources” like people, monuments, etc.) [10]. For this discussion, we will just refer to the use of URIs to identify web pages or information resources.

URIs only identify distinct resources. A single URI cannot refer to more than one resource. For example, `http://lostpedia.wikia.com/` refers to the resource, and only the resource, that is an online encyclopedia about the television series *Lost*.

Resources are not the end of the chain. Each resource may have one or more **representations**. Each representation varies in several dimensions. For example, a resource consisting of a document about cats may contain both an English and Telugu representation of the same resource. Additionally, the same resource may have an PDF representation and an HTML representation.

To recap, as Figure 9 shows, **URIs** refer to **resources** which can have one or more **representations**. The act of acquiring a representation from a URI is referred to as **dereferencing**.

The Hypertext Transfer Protocol (HTTP) is the most common protocol used by the World Wide Web to dereference URIs into representations. HTTP is not the only way to acquire representations, alternatives, such as the File Transfer Protocol (FTP), exist, but HTTP is the most widely used and will be the only protocol discussed here. The *http* at the beginning of most URIs, referred to as the **scheme**,

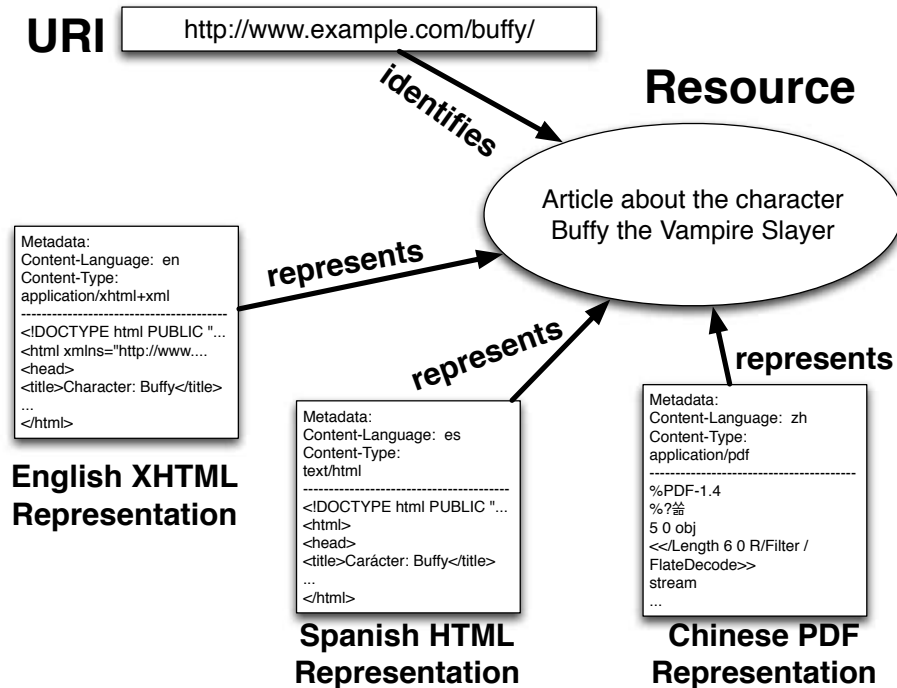


FIG. 9: Relationship between URIs, Resources, and Representations

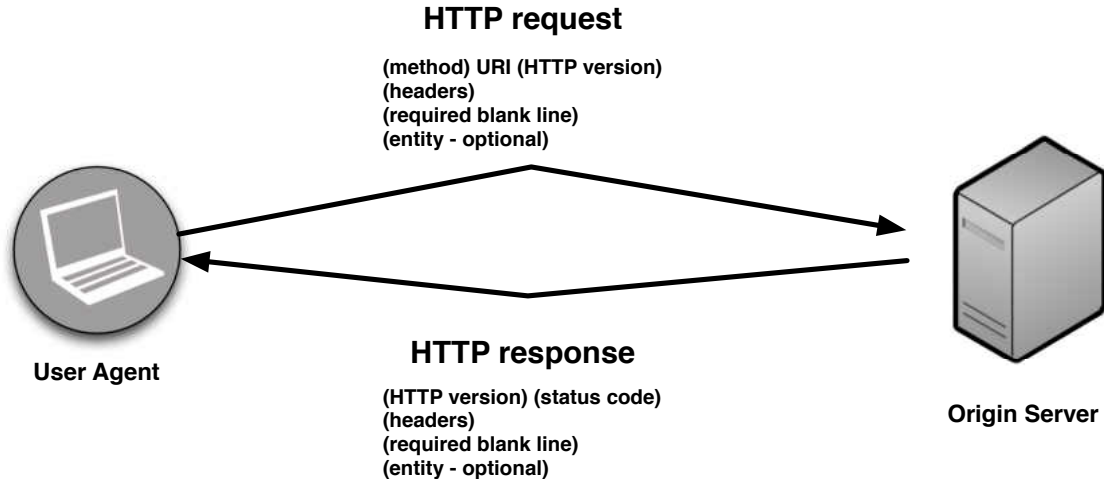


FIG. 10: Example of HTTP request-response process

indicates the use of this protocol. HTTPS [85] is similar to HTTP, but wraps all HTTP traffic in transport layer security providing authentication and encryption. We only discuss HTTP in this thesis, as what can be provided over HTTP can also be provided over HTTPS.



FIG. 11: An example of a web browser, the most common user agent type for the World Wide Web

HTTP acquires representations using a system of **requests** and **responses**, as shown in Figure 10. Requests are issued from a **user agent** and responses are sent back by the **origin server** [22].

Requests are typically initiated by the user, typically from a client user agent tool called a **browser**, an example of which is shown in Figure 11. Requests use one of have several available **methods** and Table 1 shows some examples. Only the GET and HEAD methods are useful for our thesis. The typical form of the start of a request is `method URI HTTP-version`, followed by headers supplying additional information about the request. Each header in the request is separated by a newline. The request is terminated by a blank line only containing a newline.

Responses are initiated by the origin server. Responses use one of several available

TABLE 1: Some example HTTP request methods

Method	Description	Example uses
GET	Dereference a URI to acquire a representation	Acquire the front page of a newspaper web site
HEAD	Dereference a URI to acquire just the response headers for a representation	Acquire the headers associated with a page to retrieve information about a newspaper web site's front page to see if it has changed since last time
POST	Request that the origin server accept the enclosed entity as new information for the URI	Change the contents of a message board or online document
PUT	Request that the origin server create or update the existing entity at this URI with the enclosed entity	Upload a file to the origin server at the given URI
DELETE	Remove the resource identified by the URI	Delete a file on the origin server referenced by the given URI
TRACE	Mirror what has been submitted in the request to the given URI	Diagnosis of web proxy services to ensure that the correct headers are being placed on requests
OPTIONS	Provide information about which methods are available for the given URI	Clients can determine if the server supports PUT, DELETE, or other methods at the given URI prior to initiating additional requests

status codes. Table 2 shows some example response codes used by origin servers. Of these, 200, 302, 400, and 404 are useful for our thesis. The typical form of the start of a response is `HTTP-version status-code message`. Just like the request, the response separates each header by a newline. After the response headers, a blank line containing a newline signifies the end of the headers and start of the **message body**, which is typically the content of the web page the user was looking for to begin with.

TABLE 2: Some example HTTP response status codes

Response Code	Description	Example Uses
200	The request has succeeded	Returning the web page corresponding to the given URI
301	The resource at this URI has a new permanent URI, use the new given URI in the future	A web site or page has been moved to a new server or directory location, but the owner wants old users to use the new URI
302	The resource at this URI can be found temporarily at a different URI, continue to use the URI in the request	A web site or page has been moved temporarily, or the requested resource performs some function to locate URIs for you
400	Something is wrong with the request, such as a bad header or other data; do not reissue this request with the same data, fix the request	A server does not want to accept poorly formed headers to ensure that a web application does not get corrupted
403	The server will not successfully respond to your request, do not reissue the request	A client asked for a file or directory known to, but not accessible to the origin server. Perhaps the file or directory permissions prevent the origin server's software from accessing it.
404	The resource requested cannot be found	A client asks for a web page that does not exist at the origin server
500	Something happened on the server while it tried to respond to the request and the server cannot recover	A web application is broken or has been broken by bad data from the client

Listing 12a shows a capture of the actual headers that are sent across the network to dereference the URI `http://en.wikipedia.org/wiki/The_Hunger_Games` using the Google Chrome browser. Line number 1 shows the GET method requesting the path `/wiki/The_Hunger_Games` using HTTP version 1.1; GET is the method used to request the transfer of a representation of a resource [23]. Line 2 tells the receiving server that we are interested in only acquiring this path from the

```
1 GET /wiki/The_Hunger_Games HTTP/1.1
2 Host: en.wikipedia.org
3 Accept: image/webp,*/*;q=0.8
4 Accept-Encoding: gzip,deflate,sdch
5 Accept-Language: en-US,en;q=0.8
6 Referer: https://plus.google.com/
7 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (
   KHTML, like Gecko) Chrome/36.0.1985.143 Safari/537.36
```

(a) Example HTTP Request

```
1 HTTP/1.1 200 OK
2 Accept-Ranges: bytes
3 Age: 352057
4 Cache-Control: private, s-maxage=0, max-age=0, must-revalidate
5 Connection: keep-alive
6 Content-Encoding: gzip
7 Content-language: en
8 Content-Length: 13769
9 Content-Type: text/html; charset=UTF-8
10 Date: Fri, 15 Aug 2014 13:43:03 GMT
11 Last-Modified: Mon, 11 Aug 2014 11:55:18 GMT
12 Server: Apache
13 Set-Cookie: GeoIP=US:Norfolk:36.9312:-76.2397:v4; Path=/; Domain=.wikipedia.org
14 Vary: Accept-Encoding, Cookie
15 Via: 1.1 varnish, 1.1 varnish
16 X-Cache: cp1052 hit (4), cp1065 frontend hit (906)
17 X-Content-Type-Options: nosniff
18 X-UA-Compatible: IE=Edge
19 X-Varnish: 350285837 350035840, 1753471343 1233627904
20
21 ... entity begins here
```

(b) Example HTTP response

FIG. 12: HTTP request-response examples

host `en.wikipedia.org`. We also see how the browser desires an appropriate representation of this resource. On line 3, the request uses the `Accept` header to indicate the user's desire for the `image/webp` file format, if possible. Line 4, using the `Accept-Encoding` header, indicates which encoding methods the browser prefers, and in what order of preference. Finally, line 5 shows the `Accept-Language` header, indicating which language the user prefers. Using these *Accept* headers, the web client specifies in the web request what representation will best suit the end user. The use of these headers to find the best representation from among multiple representations for a single resource is called **content negotiation**.

Listing 12b shows the response headers that are returned prior to submitting the actual representation. From this response, on line 1, we see the 200 status code, which indicates that the server can successfully return what was requested. Line 11 shows the `Last-Modified` header, indicating when this web page was last modified [24], which may not be present in all cases. Line 6 indicates to the user agent the `Content-Encoding` that the origin server is using for this entity, corresponding to the `Accept-Encoding` header in the request. In this case, the server was able to use `gzip` just like the request specified. Line 7 indicates to the user agent the `Content-Language` of the entity being returned, corresponding to the `Accept-Language` header in the request. In this case, the server was able to again find a representation using the language `en`. Line 9 indicates to the user agent the `Content-Type` of the entity being returned, corresponding to the `Accept` header in the request. This time, the server could not find the requested `image/webp` representation for this resource, instead returning `text/html; charset=UTF-8`. These `Accept*` headers in the request and their corresponding `Content*` headers from the response are what allow content negotiation to happen seamlessly to the end user.

To discuss spoilers, we need to consider that another dimension upon which we can perform content negotiation is *time*.

TABLE 3: Dimensions of content negotiation

Request Header	Response Header	Dimension	Examples	Source
Accept	Content-Type	content-type of the representation	text/html	RFC 7231
			text/plain	RFC 2616
			image/png	
			application/pdf	
Accept-Language	Content-Language	language of the representation	en	RFC 7231
			en-US	RFC 2616
			cz	
			es	
Accept-Encoding	Content-Encoding	medium, typically compression, that the entity has been processed with and also what will need to be done by the user agent to return the entity to its original form	compress	RFC 7231
			gzip	RFC 2616
			deflate	
Accept-Charset	Content-Type	the character set used by the web page	iso-8859-5	RFC 7231
			unicode-1-1	RFC 2616
Accept-Datetime	Memento-Datetime	time of the representation	Fri, 15 Aug 2014 13:43:03 GMT	RFC 7089
			Wed, 30 May 2007 18:47:52 GMT	
			Tue, 20 Mar 2001 20:35:00 GMT	

TABLE 4: Memento Resource Types

Resource	Designation	Request Headers Required	Response Headers Required
Original Resource	URI-R	none	none
Memento Resource	URI-M	none	Memento-Datetime
TimeGate	URI-G	Accept-Datetime	Link Vary
TimeMap	URI-T	none	none

2.3 MEMENTO

Tim Berners-Lee, one of the architects of the World Wide Web, originally defined four dimensions in which a resource could generate different representations. They are *target medium*, *content-type*, *language*, and *time* [9]. The first three of these evolved into four separate dimensions of HTTP content negotiation [37]. Memento finally introduces *time* as a fifth dimension in which a user can request a specific representation of a resource [73]. These dimensions are listed in Table 3.

Memento uses the existing content negotiation concept, allowing a user to specify a **datetime** for a given URI, also called **datetime negotiation** [101]. Combining these concepts together, one can browse the web as it looked on any given date and time.

Memento defines certain types of resources, summarized in Table 4 but detailed below.

First is the **original resource**, denoted as **URI-R**. This is the URI of a given resource as it is on the web at the current time. It is what we normally think of as a URI for a given resource.

Next is the **memento resource**, denoted as **URI-M**. A **memento** is a fixed representation of the original resource at a specific point in time, fulfilling Tim Berners-Lee’s final dimension of content negotiation. This resource is from where the Memento Protocol gets its name. There are one or more mementos for each original resource on the web. Consider the resource identified by `http://www.cnn.com`,

Listing 2.2: Example Response Headers for the URI-M `https://web.archive.org/web/20010601045129/http://www.cnn.com/`

```

HTTP/1.1 200 OK
Server: Tengine/2.0.3
Date: Fri, 22 Aug 2014 17:23:11 GMT
Content-Type: text/html;charset=utf-8
Content-Length: 80824
Connection: keep-alive
set-cookie: wayback_server=6; Domain=archive.org; Path=/; Expires=Sun, 21-Sep
-14 17:23:10 GMT;
Memento-Datetime: Fri, 01 Jun 2001 04:51:29 GMT
Link: <http://www.cnn.com/>; rel="original", <http://web.archive.org/web/
timemap/link/http://www.cnn.com/>; rel="timemap"; type="application/link-
format", <http://web.archive.org/web/http://www.cnn.com/>; rel="timegate",
<http://web.archive.org/web/20000620180259/http://www.cnn.com/>; rel="first
memento"; datetime="Tue, 20 Jun 2000 18:02:59 GMT", <http://web.archive.
org/web/20010601045124/http://www.cnn.com/>; rel="prev memento"; datetime="
Fri, 01 Jun 2001 04:51:24 GMT", <http://web.archive.org/web/20010601045129/
http://www.cnn.com/>; rel="memento"; datetime="Fri, 01 Jun 2001 04:51:29
GMT", <http://web.archive.org/web/20010601050038/http://www.cnn.com/>; rel
="next memento"; datetime="Fri, 01 Jun 2001 05:00:38 GMT", <http://web.
archive.org/web/20140822104304/http://www.cnn.com/>; rel="last memento";
datetime="Fri, 22 Aug 2014 10:43:04 GMT"
X-Archive-Guessed-Charset: UTF-8
X-Archive-Orig-server: Netscape-Enterprise/4.1
X-Archive-Orig-expires: Fri, 01 Jun 2001 04:52:29 GMT
X-Archive-Orig-set-cookie: CNNid=cf3013e6-11244-991371089-9; expires=Wednesday,
30-Dec-2037 16:00:00 GMT; path=/; domain=.cnn.com
X-Archive-Orig-date: Fri, 01 Jun 2001 04:51:29 GMT
X-Archive-Orig-content-type: text/html
X-Archive-Orig-last-modified: Fri, 01 Jun 2001 04:51:29 GMT
X-Archive-Orig-connection: close
X-Archive-Wayback-Perf: [IndexLoad: 677, IndexQueryTotal: 677, RobotsFetchTotal
: 3, RobotsRedis: 3, RobotsTotal: 3, Total: 974, WArcResource: 139]
Set-Cookie: wb_total_perf=974; Expires=Fri, 22-Aug-2014 17:24:11 GMT; Path=/web
/20010601045129/http://www.cnn.com/
X-Archive-Playback: 1
X-Page-Cache: MISS

```

Listing 2.3: Example Request Headers for interacting with a Memento TimeGate

```

GET /web/http://www.cnn.com/ HTTP/1.1
Host: web.archive.org
Accept: image/webp,*/*;q=0.8
Accept-Datetime: Wed, 19 Mar 2003 01:25:35 GMT
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Referer: http://www.cnn.com/
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_4) AppleWebKit/537.36 (
   KHTML, like Gecko) Chrome/36.0.1985.143 Safari/537.36

```

Listing 2.4: Example Response Headers from a Memento TimeGate, corresponding to the request from Listing 2.3

```

HTTP/1.1 302 Moved Temporarily
Connection: keep-alive
Content-Type: text/html
Date: Fri, 22 Aug 2014 17:27:57 GMT
Link: <http://www.cnn.com></http:>; rel="original", <http://web.archive.org/web
    /timemap/link/http://www.cnn.com></http:>; rel="timemap"; type="application
    /link-format", <http://web.archive.org/web/20000620180259/http://www.cnn.
    com></http:>; rel="first memento"; datetime="Tue, 20 Jun 2000 18:02:59 GMT
    ", <http://web.archive.org/web/20030215012445/http://www.cnn.com></http:>;
    rel="prev memento"; datetime="Sat, 15 Feb 2003 01:24:45 GMT", <http://web.
    archive.org/web/20030320060210/http://www.cnn.com></http:>; rel="memento";
    datetime="Thu, 20 Mar 2003 06:02:10 GMT", <http://web.archive.org/web
    /20030321181645/http://www.cnn.com></http:>; rel="next memento"; datetime="
    Fri, 21 Mar 2003 18:16:45 GMT", <http://web.archive.org/web/20140822104304/
    http://www.cnn.com></http:>; rel="last memento"; datetime="Fri, 22 Aug 2014
    10:43:04 GMT"
Location: http://web.archive.org/web/20030320060210/http://www4.cnn.com/
Server: Tengine/2.0.3
Set-Cookie: wb_total_perf=7554; Expires=Fri, 22-Aug-2014 17:28:57 GMT; Path=/
    web/http://www.cnn.com/
Transfer-Encoding: chunked
Vary: accept-datetime
X-Archive-Playback: 0
X-Archive-Wayback-Perf: [IndexLoad: 4184, IndexQueryTotal: 4184,
    RobotsFetchTotal: 1, RobotsRedis: 1, RobotsTotal: 1, Total: 7554]
X-Link-JSON: {"closest":{"wb_url":"http://web.archive.org/web/20030320060210/
    http://www.cnn.com/","timestamp":"20030320060210","status":"200"}}
X-Page-Cache: MISS

```

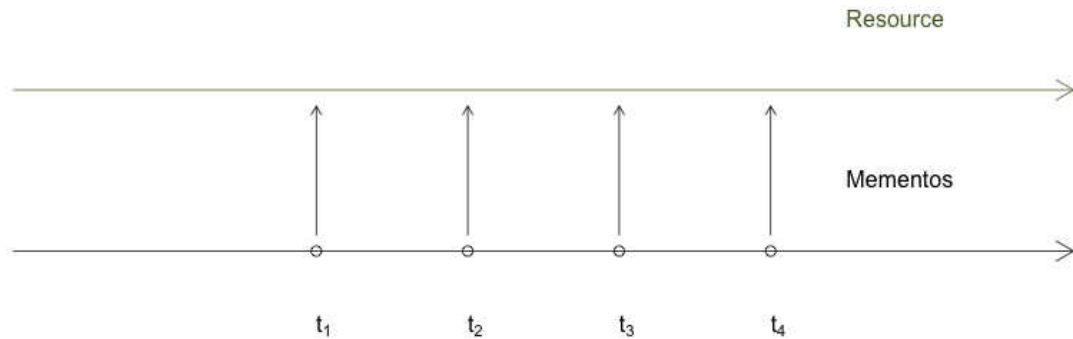



FIG. 13: Visualization of mementos captured for a given resource at times t_1 , t_2 , t_3 , and t_4

the homepage of CNN, which changes more than once per hour. As shown in Figure 13, mementos can be captured at times t_1 through t_4 for each change to a given resource, and then each memento is given a different URI-M for identification. Dereferencing a URI-M returns a typical HTTP response containing an additional `Memento-Date` header indicating the datetime that the memento was captured, as shown in Listing 2.2.

A **TimeGate**, denoted as **URI-G**, accepts a given datetime from a user and a URI-R, and produces the *best* URI-M for that datetime. TimeGate resources typically use the HTTP 302 status code to redirect a user from the TimeGate to the best URI-M. TimeGates do not really have a representation themselves, either responding with a 302 if they can successfully redirect the user, a 404 if the given URI-R is not known to the TimeGate, or 400 if the given `Accept-Date` is in the wrong format or otherwise unusable. The specification states that a TimeGate must be consistent in its decision on the *best* URI-M, but it does not prescribe which heuristics or algorithm should be used. We will explore, in Chapter 4, the options for this algorithm and which ones are not useful for avoiding spoilers. In Chapter

Listing 2.5: Example TimeMap

```

<http://lostpedia.wikia.com/wiki/The_Numbers>; rel="original",
<http://web.archive.org/web/timemap/link/http://lostpedia.wikia.com/wiki/
  The_Numbers>; rel="self"; type="application/link-format"; from="Wed, 31 Dec
  2008 03:44:05 GMT"; until="Tue, 01 Jul 2014 04:48:08 GMT",
<http://web.archive.org/web/http://lostpedia.wikia.com/wiki/The_Numbers>; rel="
  timegate",
<http://web.archive.org/web/20081231034405/http://lostpedia.wikia.com/wiki/
  The_Numbers>; rel="first memento"; datetime="Wed, 31 Dec 2008 03:44:05 GMT",
<http://web.archive.org/web/20090119183329/http://lostpedia.wikia.com/wiki/
  The_Numbers>; rel="memento"; datetime="Mon, 19 Jan 2009 18:33:29 GMT",
<http://web.archive.org/web/20090203143759/http://lostpedia.wikia.com/wiki/
  The_Numbers>; rel="memento"; datetime="Tue, 03 Feb 2009 14:37:59 GMT",
<http://web.archive.org/web/20090204193446/http://lostpedia.wikia.com/wiki/
  The_numbers>; rel="memento"; datetime="Wed, 04 Feb 2009 19:34:46 GMT",
....deletia

```

5, we will show how one of these heuristics can result in spoilers being encountered when users may not expect them.

Finally, a **TimeMap**, denoted as **URI-T**, is a list of mementos for a given URI-R. TimeMaps are ultimately used, in some form, by the TimeGate in its decision making process. For this reason, TimeMaps are meant to be machine-consumable. An example TimeMap is shown in Listing 2.5. It is important to note that a single TimeMap is not the only definitive listing of mementos for a given resource. The resource at a URI-T is only aware of some of the mementos available. Even though Memento is an attempt to aggregate the mementos at different archives, due to the *open-world* [84] nature of the web, there may be archives with mementos that are not represented in a given TimeMap.

To recap, Memento works using a form of content negotiation called datetime negotiation. Figure 14 shows the one of the patterns used by Memento to perform datetime negotiation. In step 1, the Memento client sends a HEAD request to the server containing the original resource (URI-R) to determine if the server response contains an entry in the HTTP Link header for the TimeGate (URI-G) to be used for that specific URI-R. If no TimeGate entry exists in the HTTP Link header, then the client chooses a predefined default URI-G. In step 2, the Memento client contacts the URI-G with the `Accept-Datetime` request header, specifying the datetime desired by the user. The URI-G then uses this information to find the best URI-M

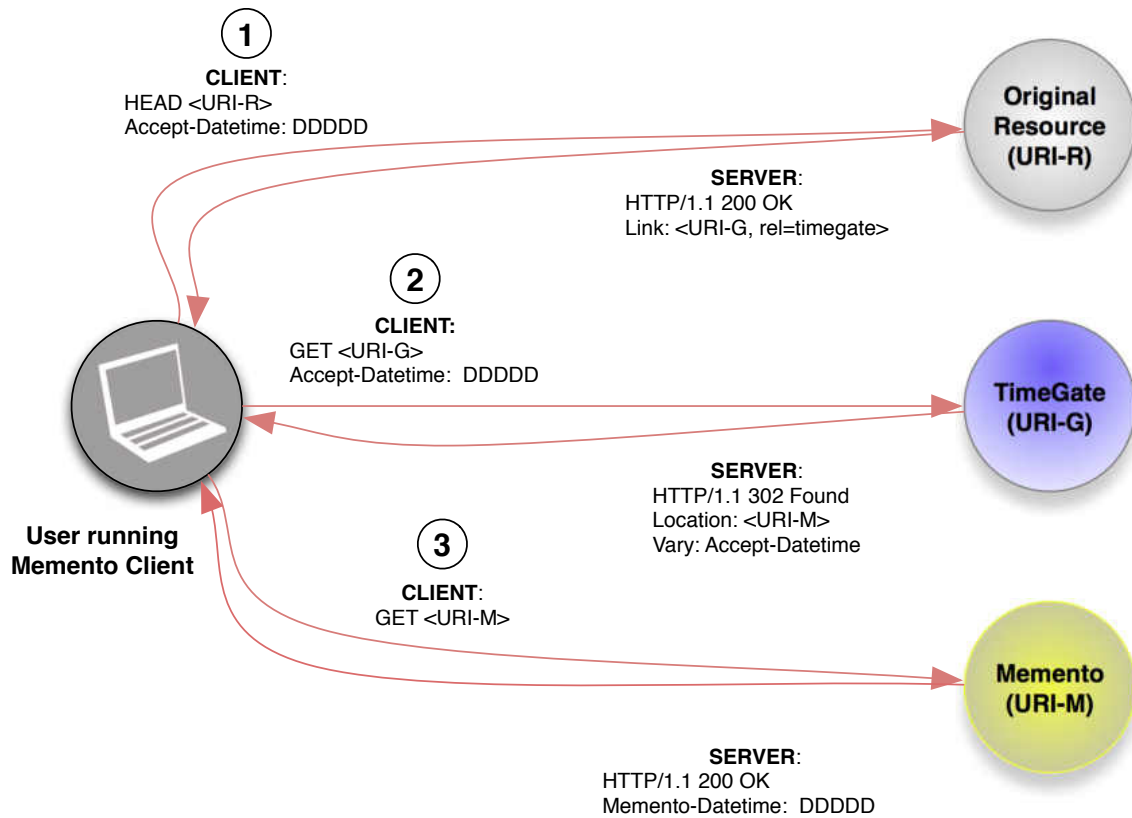


FIG. 14: General Memento pattern

for that datetime. Once a URI-M is chosen, the URI-G includes a `Location` header in the HTTP response, along with the 302 redirect status code to redirect the client to the URI-M. Finally, the Memento client visits the URI-M.

These steps can all be automated for the end user, reducing the experience to a few mouse clicks to choose the desired datetime and visit the appropriate URI-R. Figure 15 shows a screenshot of a user engaging in datetime negotiation with the Memento Time Travel Chrome Extension [93], introduced earlier, referred to hereafter as *Memento For Chrome*.

Web site owners by default support URI-Rs, the Memento project provides infrastructure for URI-Gs, but who maintains the URI-Ms? Web archives have traditionally filled this role.

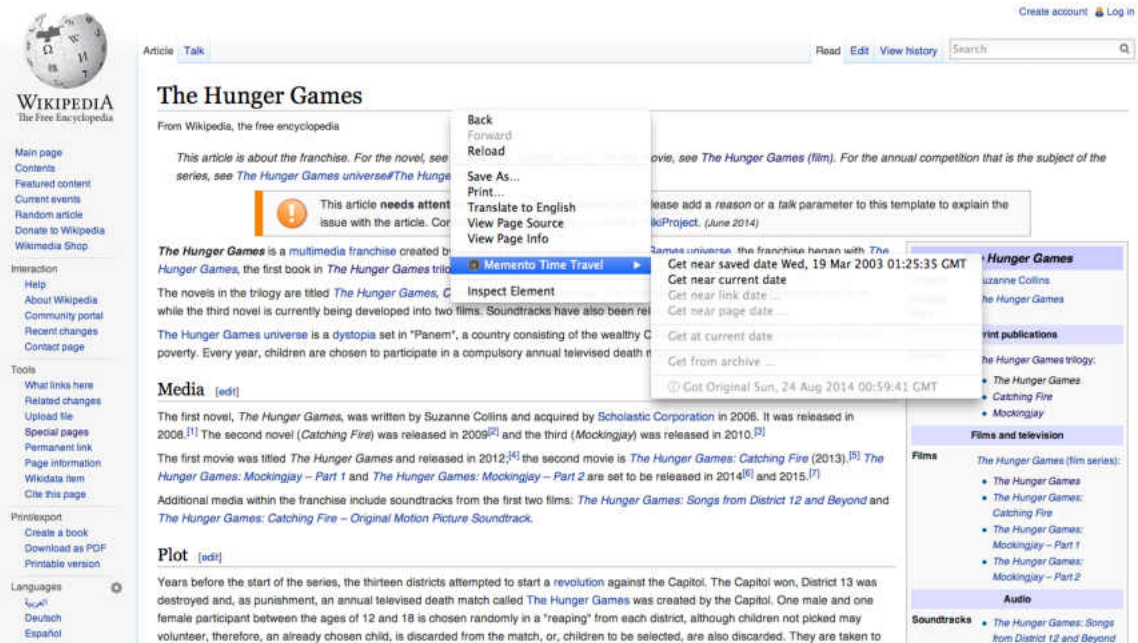


FIG. 15: Screenshot of the Memento Time Travel Chrome Extension, a Memento client

2.4 WEB ARCHIVING AND THE WAYBACK MACHINE

The Internet Archive, founded in 1996 by Brewster Kahle, became the first widely known public web archive [99]. It continued the process of digital preservation into the realm of the web. Thanks to their enterprising work, Tim Berners-Lee's idea of content negotiation in time is now possible. Web archives are where URI-Ms are traditionally stored.

Web archives work by using a special program called a **crawler**, which starts with a set of URIs (referred to as the **link heap** or the **frontier**), dereferences those URIs, and archives the representations [67]. Additionally, a **parser** exists to process the representation and determine if additional URIs exist in the representation (i.e., links to other resources). For example, if a page being archived is HTML and contains a tag ``, then the parser would extract `http://example.com` from this tag for inclusion onto the link heap. If those additional URIs exist, then those URIs are added to the heap so the crawl can continue. This way a web archive can acquire much of the World Wide Web. Figure 16 shows the simple architecture for such a crawler.

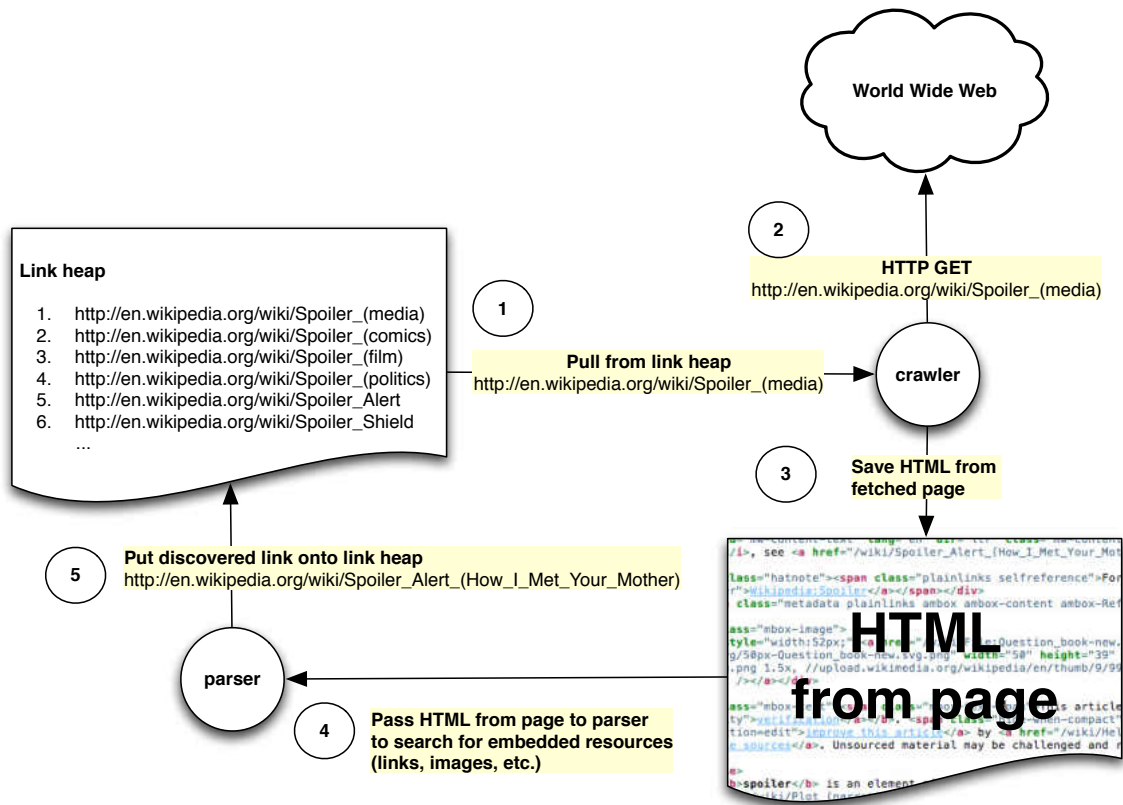


FIG. 16: Architecture for a simple web crawler

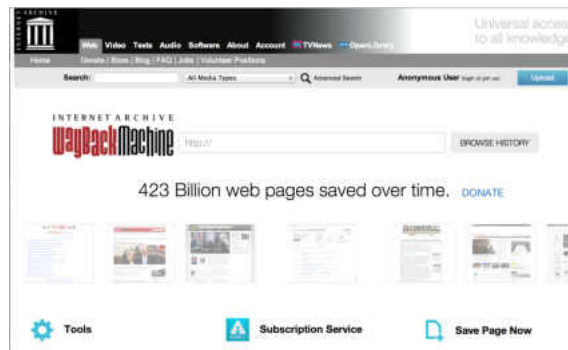
One of the benefits of web archiving is that these crawlers only dereference URIs at specific points in *time*, leading us to view web resources as they change over time. Because web archive crawlers revisit the same resource at different points in time, we have snapshots of what pages looked like at these particular points. Web archives are intended to be permanent records, so these snapshots last far longer than they do in search engine caches [2] and thus are the most reliable way to access the previous versions of general web pages.

We refer to these snapshots as **mementos**. As mentioned above, the Memento project uses the Memento protocol to make these snapshots available easily, without relying on a specific web archive. Because these mementos are only observed at a given time, and not all pages contain a reliable Last-Modified header, Memento uses the Memento-Datetime header to indicate when the archived copy was actually observed.

It is worth discussing the differences between the `Last-Modified` and `Memento-Datetime` headers for a moment. `Last-Modified` was originally intended to indicate when the given web pages was last altered by the author. Unfortunately, in the current web, the content of a given page may stay the same, even though the surrounding data may not. Consider the case where an article on a news website stays the same, but the site itself has changed their company banner. Now that the company banner has been changed, the HTML `` tag in the page must change, thereby resulting in a change to the `Last-Modified` date, even though the actual content of the article has stayed the same. Consider another case whereby you are creating a web archive that crawls other web archives. Your crawl will detect the `Last-Modified` time of the mementos it collects, not the actual time that the memento was archived, thus the `Last-Modified` date applies to the representation, but not the content. For these reasons, the `Memento-Datetime` header was created to indicate when that particular representation was observed [72].

The **Wayback Machine** is a graphical utility for browsing and playing back these mementos in the Internet Archive [27]. Figure 17a shows the front page of the Wayback Machine, allowing a visitor to enter a URI for a given resource. Figure 17b shows the calendar view of the mementos for a given resource, allowing the user to select a date containing a blue circle to see what the memento looks like for that given date [78]. Figure 17c shows the interface for a given memento, as captured. Additionally, if the user clicks links within the memento, then the Wayback Machine tries to follow those links within the archive, arriving at a date closest to the one originally chosen [71]. The Wayback Machine also has Memento support [74].

In Chapters 5, 6, and 7 we will show that while the Wayback Machine and the Internet Archive can be useful to avoiding spoilers, the *mindist* heuristic of using the closest date of the memento to the date provided by the user is not reliable enough to avoid spoilers entirely.



(a) Main Page of the Wayback Machine

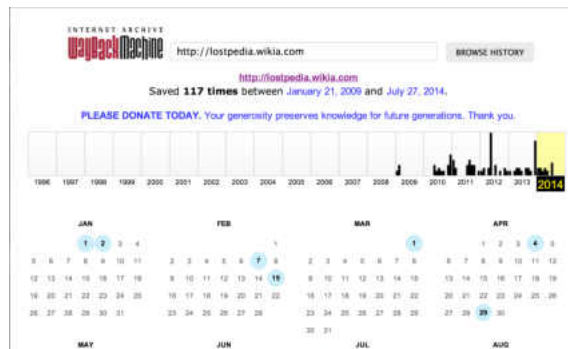
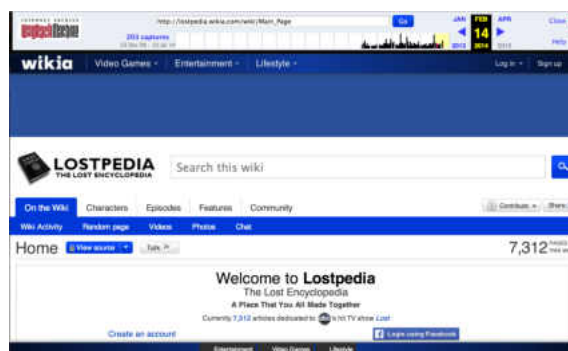
(b) Calendar view of Mementos for the URI <http://lostpedia.wikia.com>(c) Viewing the specific memento of <http://lostpedia.wikia.com> from February 14, 2014**FIG. 17:** Wayback Machine Screenshots

TABLE 5: Some examples of wikitext compared to HTML

MediaWiki Syntax	HTML	Rendered
New paragraph text	<code><p> New paragraph text </p></code>	New paragraph text
<code>'''bold'''</code>	<code>bold</code>	bold
<code>''italics''</code>	<code>italics</code>	<i>italics</i>
<code>=heading=</code>	<code><h1>heading</h1></code>	heading
<code>==heading==</code>	<code><h2>heading</h2></code>	heading
<code>* unordered list item * unordered list item</code>	<code> unordered list item unordered list item </code>	<ul style="list-style-type: none"> • unordered list item • unordered list item
<code># ordered list item # ordered list item</code>	<code> ordered list item ordered list item </code>	<ol style="list-style-type: none"> 1. ordered list item 2. ordered list item
<code>[http://www.example.com Example link]</code>	<code> Example link</code>	Example link

2.5 WIKIS, WEB ARCHIVES, AND MEMENTO

In 1994, Ward Cunningham developed software for his company's web site that consisted of a series of interconnected web pages each providing an easy editing interface while also keeping a history of all edits [55]. He named the software WikiWiki-Web, but the name has since been shortened to just **wiki**. Much like Cunningham's original wiki, each wiki provides the ability to easily add or edit a new page while keeping a record of every previous revision of the given page.

Wikipedia, established in 2001, as an online encyclopedia, quickly became one of the most popular sites on the Internet. By 2005, a study had shown that Wikipedia was as accurate as Encyclopædia Britannica [31]. Wikia, established in 2004, provides fan-based wikis from everything from television shows to books to video games. Wikia has more than 100,000 wikis on various topics [57]. Both of these sites use the popular wiki software package **MediaWiki**.

Wikis are popular due to the use of **wiki syntax**, which requires typing fewer characters than HTML. Table 5 shows some examples of wiki syntax when compared to HTML. Wiki pages, like the example shown in Figure 18 allow the user to quickly

The image shows a screenshot of the Wikipedia page for Abraham Lincoln. The page layout includes a top navigation bar with links like 'Shawnpress', 'Talk', 'Sandbox', 'Preferences', 'Beta', 'Watchlist', 'Contributors', and 'Log out'. Below this is the article title 'Abraham Lincoln' and a sub-header 'From Wikipedia, the free encyclopedia'. The main text begins with a lead sentence: 'This article is about the American president. For other uses, see Abraham Lincoln (disambiguation).' The text continues with a detailed biography of Lincoln, mentioning his birth in 1809, his role as the 16th president from 1861 to 1865, and his impact on the American Civil War and the abolition of slavery. A portrait of Lincoln is shown on the right side of the page. Below the portrait is a summary table with the following information:

Abraham Lincoln
 <div>An 1863 daguerrotype of Lincoln, at the age of 54.</div>
16th President of the United States
In office
March 4, 1861 – April 15, 1865
Vice President
Herbert H. (1861–1865) Andrew Johnson (1865)
Preceded by
James Buchanan
Succeeded by
Andrew Johnson
Member of the U.S. House of Representatives
from Illinois's 7th district
In office
March 4, 1847 – March 4, 1849
Preceded by
John Henry
Succeeded by
Thomas Harris
Member of the Illinois House of Representatives
In office
December 1, 1834 – 1842
Personal details

FIG. 18: Example Wiki Page

create or edit a page using this wiki syntax, as shown in Figure 19.

For avoiding spoilers, however, we are interested in the wiki's ability to display previous versions of pages. A history page, shown in Figure 20 shows the previous revisions for a given page, allowing editors to revert edits and view previous content, as we see in Figure 21. Previous revisions of MediaWiki articles contain notices, like the one shown in Figure 22.

Web archives do archive wiki pages, and previous mementos of wiki pages can be viewed in Memento and the Wayback Machine as can be seen in Figure 23.

Wikis, and other Content Management Systems, store the actual time each revision of a page was created, making them of special interest to those trying to view past versions of pages on the web. Typical web sites, such as news sites, do change frequently, but have no public way of viewing previous revisions. Wikis allow users access to all previous revisions. This makes it possible to measure the effectiveness of the capture rate of a web archive against wikis.

Wikis are also effectively their own archives. This is important, because it means that the Last-Modified datetime of each wiki revision *is* the datetime that it was archived into the wiki. This means that the Last-Modified datetime of a wiki revision is the same as its Memento-Datetime, meaning that **each wiki**

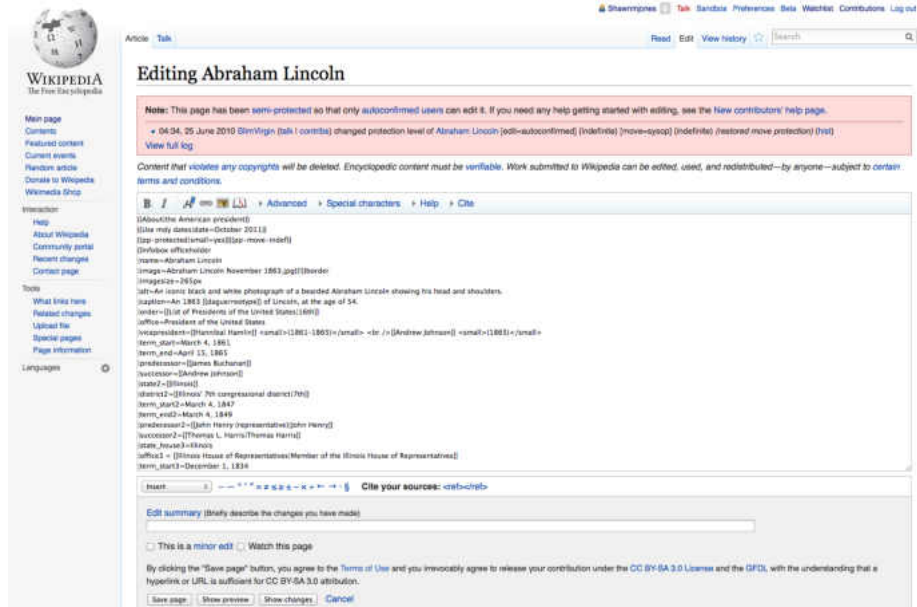


FIG. 19: Example Edit Page for a Wiki Article

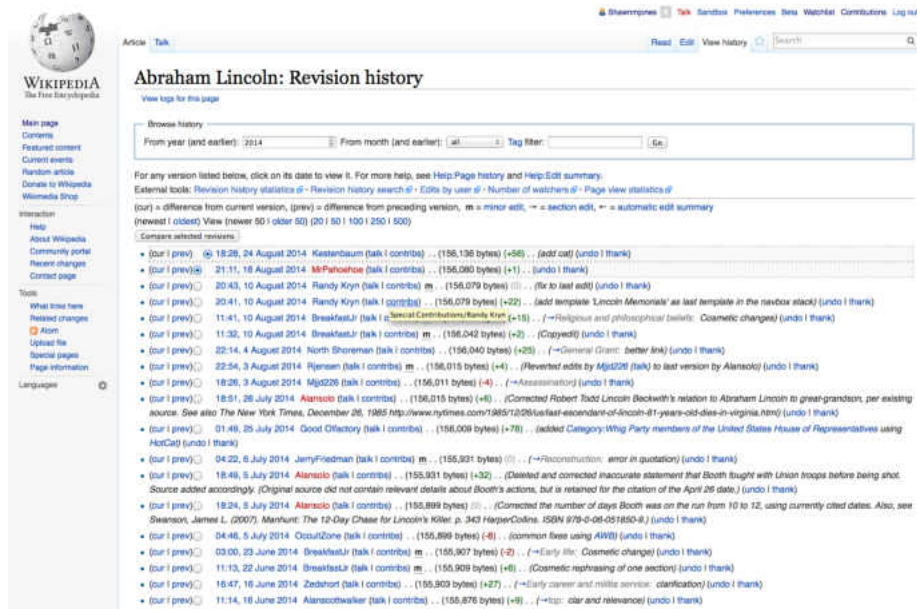


FIG. 20: Example History Page for a Wiki Article

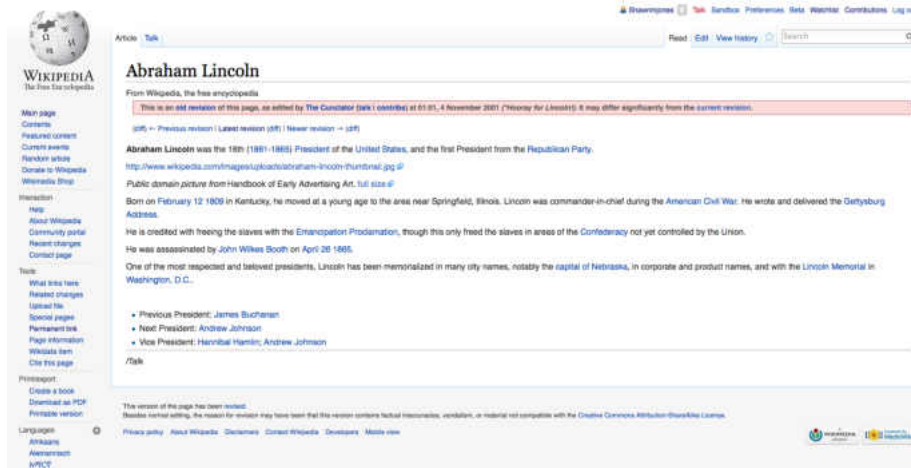


FIG. 21: Example of viewing an earlier revision of a Wiki Article

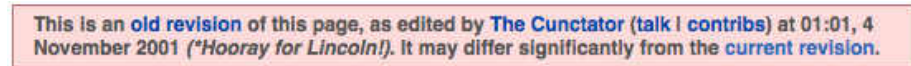


FIG. 22: Example of a revision notice, present at the top of old revisions in MediaWiki



FIG. 23: Example of a wiki page viewed from the Wayback Machine

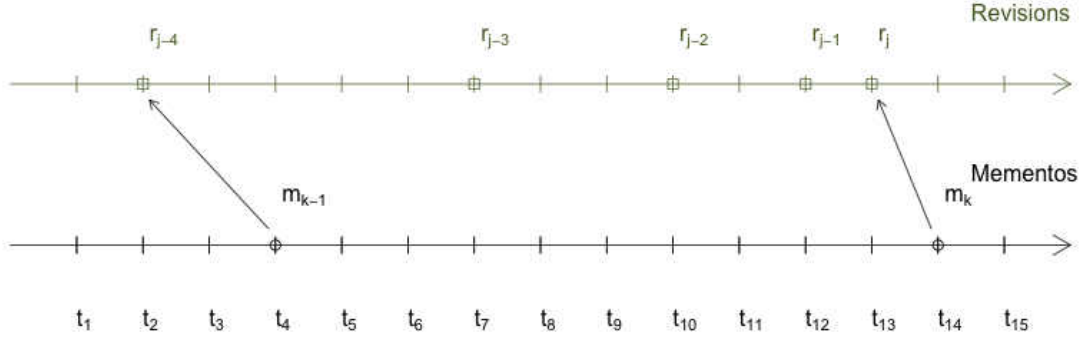


FIG. 24: Example Timeline Showing Captured Mementos of Wiki Edits

revision is a memento. When we consider an external archive to a wiki, we can then compare the Memento-Datetime of the external archive’s mementos to the Memento-Datetime of each wiki revision.

To differentiate between the two in this thesis, we will refer to the mementos for a given wiki page that are stored and referenced by the wiki as **revisions** and the mementos in a web archive that captures these revisions as **mementos**.

Figure 24 shows two timelines. The top, in green, denotes a wiki revision timeline for a single wiki page. The bottom, in black, denotes a web archive timeline, containing mementos captured at certain times. The diagonal arrows show the capture relationship between each memento and its associated wiki revision. The times t_1 through t_{15} at the bottom show the memento-datetimes for each revision and/or memento.

From this figure, we see that memento m_k was archived by a web archive at datetime t_{14} . We denote this as $m_k@t_{14}$. Likewise, $r_{j-4}@t_2$ denotes that revision r_{j-4} was archived by the wiki at time t_2 .

Also from this figure, we see that memento m_k is a capture of r_j . We denote this as $m_k \equiv r_j$. Likewise, using this same figure, we see that memento m_{k-1} is a capture

of r_{j-4} , or $m_{k-1} \equiv r_{j-4}$.

We use the notation \equiv rather than $=$ because web archives can not always capture web pages as faithfully as desired, resulting in pages that are not quite the same as the originals. That problem is referred to as **temporal coherence** [4], and solving it for web archives is outside the scope of this thesis. We mention it here for completeness, and also because our solution in Chapter 8 is able to partially address the problem for wikis.

Using this notation for Figure 24, we see that $m_k @ t_{14} \equiv r_j @ t_{13}$ and $m_{k-1} @ t_4 \equiv r_{j-2} @ t_2$. What about $r_{j-1} @ t_{12}$, $r_{j-2} @ t_{10}$, and $r_{j-3} @ t_7$? Where are those revisions' mementos in the archive? They do not exist altogether in the archive. These **missed updates** are one of the reasons that wikis, rather than web archives, can be used to more reliably avoid spoilers. In essence, web archives do not crawl sites with enough frequency to acquire all revisions, and currently have no way of knowing when wikis update.

Seeing as each wiki page has a URI, like `http://en.wikipedia.org/wiki/Abraham_Lincoln`, and each wiki revision has its own URI, like `https://en.wikipedia.org/w/index.php?title=Abraham_Lincoln&oldid=345783631`, it is possible to use Memento directly on the wiki rather than going through the web archive. In this case the wiki page URI becomes the URI-R and the revision page URI becomes a URI-M. Unfortunately, wikis have no native *TimeGate* to perform datetime negotiation, so not all of the pieces are present to fully support Memento. In Chapter 8 we describe our solution for bringing this functionality to MediaWiki.

2.6 THE NAÏVE SPOILER CONCEPT AND BRINGING IT ALL TOGETHER

Figure 25 formalizes the progression of events for a fan wiki article. Like we saw in Figure 24, we have two timelines on the bottom, representing the mementos and wikis as before. To understand the spoiler problem, we introduce a third timeline, consisting of the times **events** occur. For our purposes, events represent a release at a specific datetime of a single episode in a series of episodic fiction (e.g., a book, movie, or television episode).

So, using Figure 25, an event, such as $e_{i-2} @ t_2$ leads a fan to write revision $r_{j-2} @ t_3$ which is eventually archived as memento $m_{k-2} @ t_4$. The same goes for the other

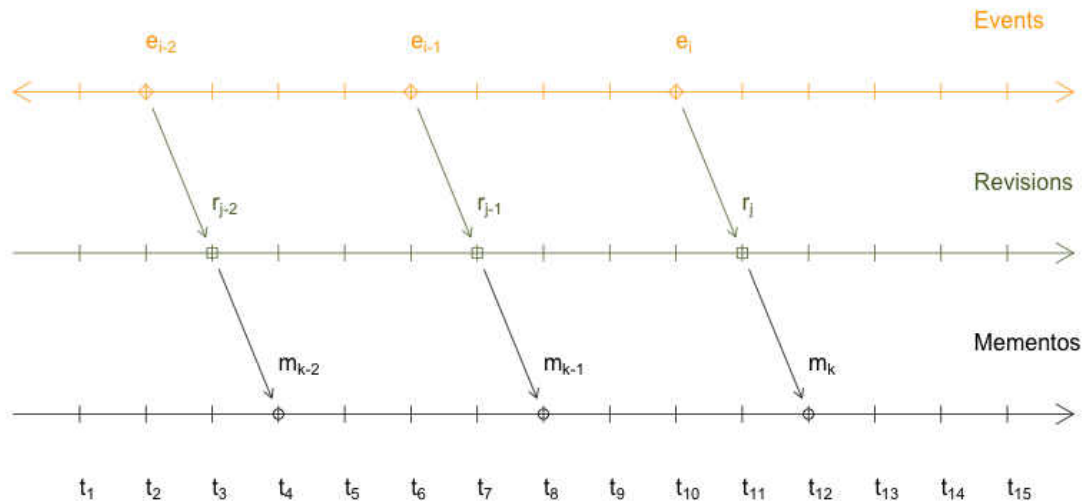


FIG. 25: Each event can inspire a new wiki revision which may be captured as a memento by a web archive

events, revisions, and mementos. This pattern was first noticed by Steiner with Wikipedia related to news [95], where a temporary increase in the number of wiki edits (revisions) would follow world events. For the moment we are ignoring missed updates.

Figure 26 shows a graphical form for the naïve definition of a spoiler. Using this figure, we see that a resource has revisions r_j and r_{j+1} ; if event e_i occurs at time t_8 then revision r_{j+1} of that resource altered at a time greater than t_8 is a **spoiler**. Any revision of resource r altered at a time less than t_8 is considered **safe**.

We call this the naïve spoiler concept because we are only using the memento-datetime of the resource for comparison. We are not analyzing the contents of r_j to determine if the information contained within is not desired. **We are making the assumption that revision r_j , existing after the event e_i intentionally or unintentionally contains information about event e_i , which someone concerned about spoilers is attempting to avoid.**

Using this concept, we can derive the following formal relationships for episode

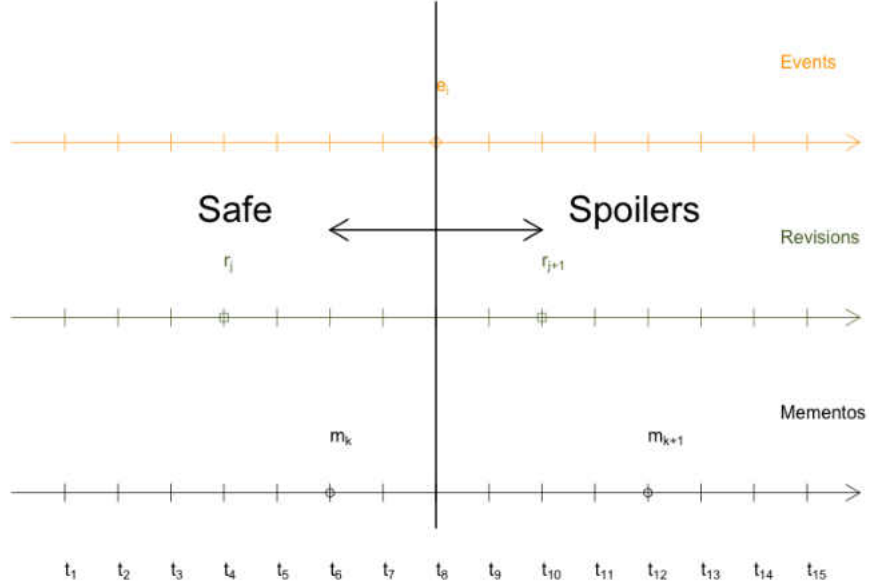


FIG. 26: Representation of a Naïve Spoiler Concept

e_i and wiki revision r_j :

$$t_{r_j} \geq t_{e_i} \implies \textit{spoiler} \quad (1)$$

$$t_{r_j} < t_{e_i} \implies \textit{safe} \quad (2)$$

where t_{r_j} corresponds to the memento-datetime of revision r_j and t_{e_i} corresponds to the time of event e_i . Table 6 summarizes the notation that we will use for revisions, mementos, events, and datetimes throughout this paper.

This relationship holds for wiki revisions because the revision that existed just prior to episode $e_i@t_i$ is the page that would have existed at the time desired.

For our solution to avoid spoilers, we will use Memento. We will allow an end user to submit two pieces of information to a Memento TimeGate: a datetime $t_a < e_i@t_i$, where $e_i@t_i$ is the episode they have not viewed yet, and the URI-R of the resource they wish to view. From the Memento TimeGate they then get the revision back that existed at the time t_a requested.

Wikis also do not rewrite their links, so links between wiki pages always refer to the current URI-R. A Memento client is needed to redirect users who wish to use

TABLE 6: Notation used in this thesis

Notation	Meaning
e_i	the i^{th} episode in a series
r_j	the j^{th} revision from a wiki
m_k	the k^{th} memento from a web archive
t_p	the p^{th} datetime in a series of datetimes
$r_j@t_p$	the j^{th} revision at datetime t_p
$m_k@t_p$	the k^{th} memento at datetime t_p
$e_i@t_p$	the i^{th} episode at datetime t_p
$m_k \equiv r_j$	the k^{th} memento in the archive is a capture of revision r_j
$r_j \equiv m_k$	the j^{th} revision in the wiki was captured as memento m_k
t_{r_j}	the memento-datetime of wiki revision r_j
t_{m_k}	the memento-datetime of memento m_k
t_{e_i}	the datetime that event e_i occurred

web time travel to the URI-Ms corresponding to their chosen datetime. This way they stay in the past. This is why just visiting the history pages for a given wiki article is not enough, we want to use Memento to help users stay in the time period prior to the spoilers.

As we will show in chapters 4, 5, 6, and 7, the mementos captured by web archives are very sparse and the TimeGate heuristic used for choosing the best memento for the given datetime sometimes produces spoilers.

First, we will see how others have tried to address the spoiler problem.

CHAPTER 3

RELATED WORK

This chapter discusses those previous attempts to analyze spoilers, solve the problem of spoilers, or allow web time travel for MediaWiki. It builds on the narrative in the previous chapter, because it, too, lays the groundwork for those who have come before and why their solutions are useful or are not useful to our thesis.

3.1 EXISTING STUDIES ON SPOILERS

In 2011, Leavitt and Christenfeld conducted a study where 819 participants took part in three experiments [54]. They were given stories to read, and for each story, the researcher created a spoiler paragraph describing the story and revealing the “outcome in a way that seemed inadvertent”. If a subject had already read a story, their data for that story was excluded from the experiment. Each version of each story was rated on a 10-point scale, where 10 was considered *best*. Unexpectedly, as shown in Figure 27, slightly more participants preferred spoiled stories over unspoiled stories. The study also indicated that readers are unable to compare spoiled and unspoiled experiences and thus those who preferred spoiled stories may just prefer spoilers in general.

Schirra, Sun, and Bently conducted a study of **live-tweeting** while the television show *Downton Abbey* was airing [89]. Live-tweeting is a process whereby those watching a television show episode discuss the show on a social media web site, such as Twitter, while the episode is airing. This study consisted of a sample of 2,234 participants who live-tweeted during the highly anticipated third season premier and beyond. The intention of the study was to determine how long users continued to engage in live tweeting after the first episode. They discovered a complex social process with its own evolving rules and customs. Semi-structured interviews were conducted among some of the participants.

Downton Abbey represents a global problem because it airs in the United Kingdom months prior to the United States. Some of the United Kingdom live-tweeters would hold off revealing spoilers, but still live-tweet during the American air dates

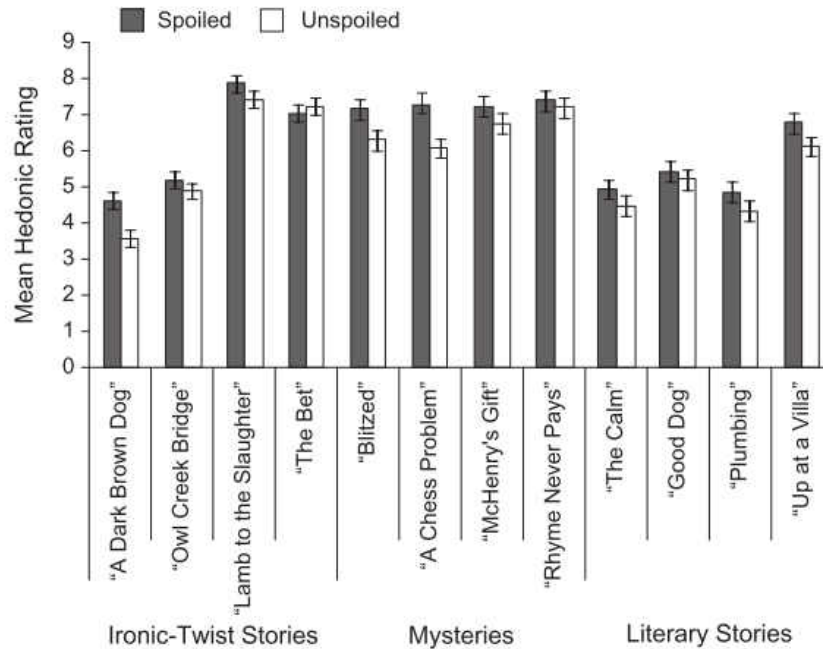


FIG. 27: Results of Leavitt and Christenfeld’s spoilers research, indicating a slight preference for spoiled stories over unspoiled stories. (Error bars represent standard errors)

so that they could vicariously share in the story reveals and plot information as their Americans friends experienced it. Others would concoct methods to communicate major plot twists, such as using ambiguous pronouns, without spoiling the story for their friends. Because the broadcast can experience propagation and transmission delays, some live-tweeters had the show spoiled by others because their friends’ experience differed by a matter of a minute or less, resulting in tweets that arrived to the tweeter before they actually got to experience what the topic of the tweet. Some live-tweeters would avoid social media altogether, finding that their experience could still be spoiled by others. In one case, a live-tweeter stopped watching the show once another twitter user spoiled it for them.

This is also consistent with a study conducted by Johns, also using interviews in a small group of participants who also engaged in *two screen viewing*, a more generic name for live-tweeting [43]. In this study Johns discovered that those who used digital video recording (DVR) devices, such as the TiVO, would avoid social media until they had watched their show. Also, some would eschew DVRs because they

wanted to participate in live-tweeting. This kind of frustration with spoilers indicates a social problem that Leavitt and Christenfeld’s study attempted to indicate was not an issue.

Because of the phenomenon of spoilers in social media, Boyd-Graber, Glasgow, and Zajac conducted an evaluation of machine learning approaches to find spoilers in social media posts [14]. They used classifiers on multiple sources to determine which posts should be blocked. They determined that spoilers are identified by transitive words, such as “kill” that affect the outcome of a plot because they link characters to each other. They also mention that spoilers refer to events “*later* than the viewer’s knowledge of the current work”, suggesting that any machine learning technique used for avoiding spoilers in social media must be smarter than just blocking all posts about a particular topic [32, 42]. Their classifiers were trained by crowdsourcing and pulling in data from the Internet Movie Database¹, TV Tropes², and Episode Guides³ online resources. By utilizing these additional sources, they were able to use machine learning techniques to identify spoilers better than their predecessors, who relied primarily on term matching and small data sets.

Leaver wrote an essay about *The Tyranny of Digital Distance*, further emphasizing the issue of television shows airing in one country months before another [53]. Leaver discusses the same issue experienced by the American *Downton Abbey* fans, but this time with the television show *Battlestar Galactica*. Leaver mentions how *Battlestar Galactica* aired in the United States six months or longer prior to airing in Australia. He argues that the Internet provides near instantaneous communications between fans of a television show, but the broadcast and distribution networks for television content do not engage in a simultaneous release of content, resulting in fans experiencing spoilers because other fans live in a different time zone, or in a country where legal issues are delaying the release of content. He even mentions that using news sources, such as *Google News*⁴ can result in spoilers for those who live in a different country than the one creating the content. He also refutes the argument, put forth by Leavitt and Christenfeld, that spoilers do not affect the enjoyment of fiction, by mentioning that plot leaks for the *Harry Potter* novels were disastrous for fans, resulting in public outcry [17].

¹<http://imdb.com>

²<http://tvtropes.org>

³<http://epguides.com>

⁴<http://news.google.com>

Of particular interest to advertisers are the cases where viewers abandon shows, or online content, due to spoilers. For this reason, there is an actual financial benefit to content producers to remedy these problems [100].

3.2 NOTICES, BLURRING TEXT, AND OTHER TECHNICAL ATTEMPTS AT SPOILER MANAGEMENT

There have been several attempts to address spoilers on the Web. Historically, the solution has been to display a large **spoiler alert** notice on the page [41]. It is expected that this notice will indemnify the site of any harm caused by visitors proceeding to other pages on the site. In practice, it may cause visitors to leave the site, resulting in lost advertising revenue. Figures 28a, 28b, and 28c show example screen captures of these spoiler alerts.

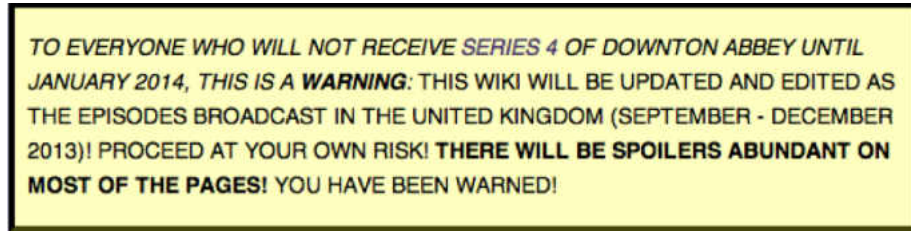
Wikipedia used to include spoiler alerts on pages about fiction, but decided in 2007 that such notices fell into their “No disclaimers in articles” guideline [111].

These warnings do not merely apply to web site visitors. As shown in Figure 29, some wikis even want editors to refrain from adding spoiler content for upcoming episodes so that the majority can enjoy the fictional work as it is released.

The *TV Tropes* web site, as shown in Figure 30, displays text containing spoilers as white text on white background, which can be highlighted by visitors that want to view the hidden content.

Figure 31 shows the demonstration page of the *Spoiler Alert* JavaScript library. [39] As one can see in the figure, the text and images that may contain spoilers can be blurred, preventing visitors from viewing the information. Visitors who have already seen the episode, read the book, or otherwise consumed the fiction they want to read about can just click on the blurred area to remove the blur from the text or image and view the information contained. If a visitor accidentally clicks a section of blurred text, they can click it again to reactivate the blur.

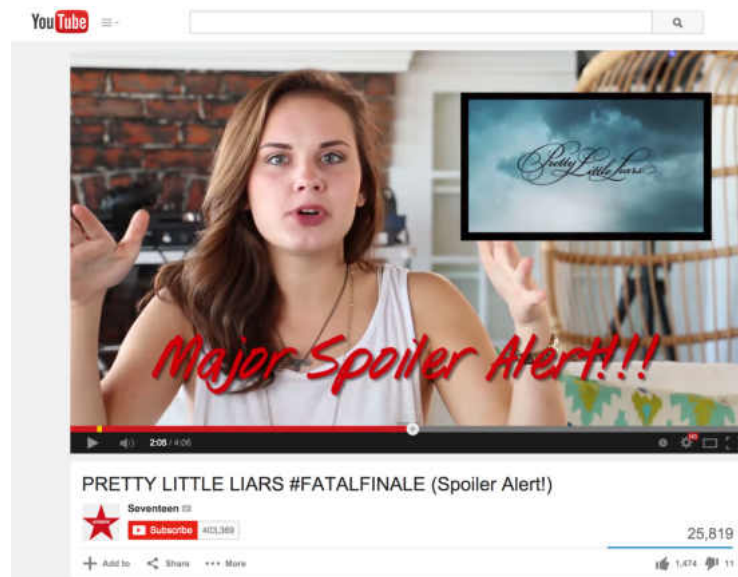
Artjom Kurapov has created a draft HTML microformat that extends HTML so that individual links and images can be annotated for level of violence, nudity, obscenity, and spoilers [50]. Using these microformats, as shown in Listing 3.1, one can annotate links and images with a value from 0 to 100 to indicate these levels which could then be consumed by a browser for action. For example, if a browser sees a value of 100 for `data-xrate-nudity` and the user has specified that they want to avoid nudity, then the browser could block the image. In the same way, the



(a) Spoiler Alert Notice from the Wiki for the show *Downton Abbey*, captured on December 18, 2013 from http://downtonabbey.wikia.com/wiki/Downton_Abbey_Wiki

MAJOR SPOILER ALERT: This story contains many details of Thursday's season finale episode of ABC's *Scandal*.

(b) Spoiler Alert Notice from a *Deadline Hollywood* article about the TV show *Scandal*, captured on September 1, 2014 from <http://deadline.com/2014/04/scandal-spoiler-season-finale-abc-shonda-rhimes-716318/>

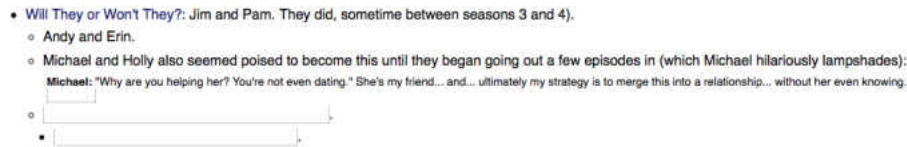


(c) Spoiler Alert Notice from a YouTube video page discussing the season finale of the TV show *Pretty Little Liars*, captured on September 1, 2014 from <https://www.youtube.com/watch?v=LkR2FhcbMTE>

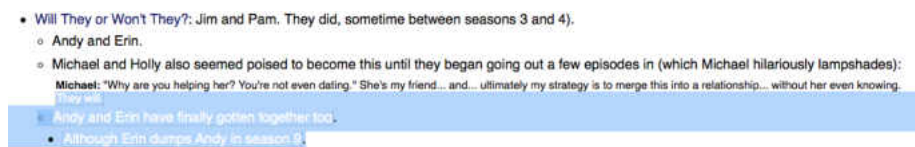
FIG. 28: Examples of Spoiler Notices on the Web

The World of Ice and Fire (US [Ⓔ], UK [Ⓔ]) will be released October 28. [\[edit\]](#)
 Editors, please refrain from including information from the book prior to
 November 27.

FIG. 29: Guidance for wiki editors for the site *A Wiki of Ice and Fire*, indicating that they should not include plot details for an upcoming book, avoiding the addition of spoilers to existing pages



(a) The spoiler text has appears white on white background, hiding it from view



(b) The spoiler text can be highlighted by the user, revealing it

FIG. 30: *TV Tropes* web site examples of spoiler text shown as white text on white background for the television show *The Office*

browser could block access to a spoiler.

These attempts at warning the user, no matter how good-intentioned, do not actually meet our goals. We want to be able to browse the version of the page without the spoiler data at all, which is not possible with these notices or even the blurred text provided by the JavaScript library. Even knowing that the spoilers are there can be dangerous, as the surrounding text can offer clues as to what information is contained within. For example, the blurred text may talk about a character's death, but the reader may infer that the character is dead due to the fact that non-blurred text all refers to the character in the past tense.

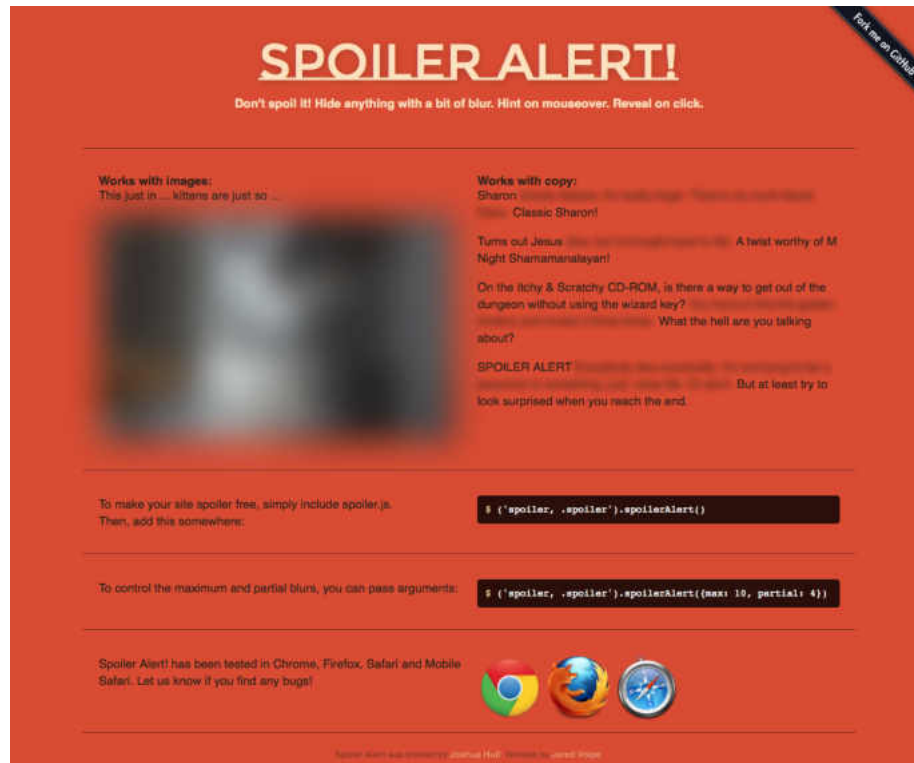


FIG. 31: Demonstration web page for the Spoiler Alert JavaScript library, showing blurred text and images instead of spoiler information

Listing 3.1: Examples of xrate microformats for avoiding spoilers, pornography, and violence in links and images

```
<a href="http://www.example.com/who-my-character-fell
-in-love-with" data-xrate-spoiler="100" data-xrate-
sex="20">link on information about this episode</a>


```

3.3 EXISTING SOFTWARE THAT ATTEMPTS TO HELP USERS AVOID SPOILERS

Apps such as Tumblr Savior, Facebook Posts Filter, Open Tweet Filter, and TweetDeck all have features that prevent the viewing of spoilers. These applications block content as the user views it.

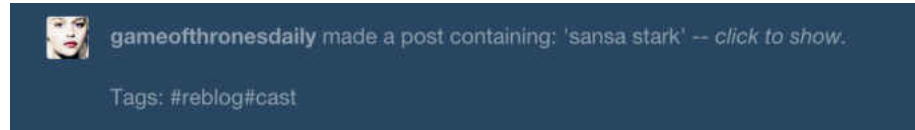
Tumblr Savior is an extension to the Google Chrome Browser that blocks entries in a Tumblr user’s feed [97]. Figure 32a shows what Tumblr posts look like when blocked with this tool. A user can click on the text “click to show” in order to view the content that has been blocked. Figure 32b shows the content that was blocked in this example. The white box outlined in red on the left containing an italic capital letter T shows that this post was blocked by Tumblr Savior.

Facebook Posts Filter is an extension to the Google Chrome Browser that takes keywords to specify which Facebook posts should be blocked [104]. Unlike Tumblr Savior, the Facebook Posts Filter tool prevents the Facebook post from displaying in the Facebook feed entirely, meaning a user is completely unaware of the post’s existence. Figure 33a shows the configuration window for Facebook Posts Filter. Facebook Posts Filter is configured from the browser.

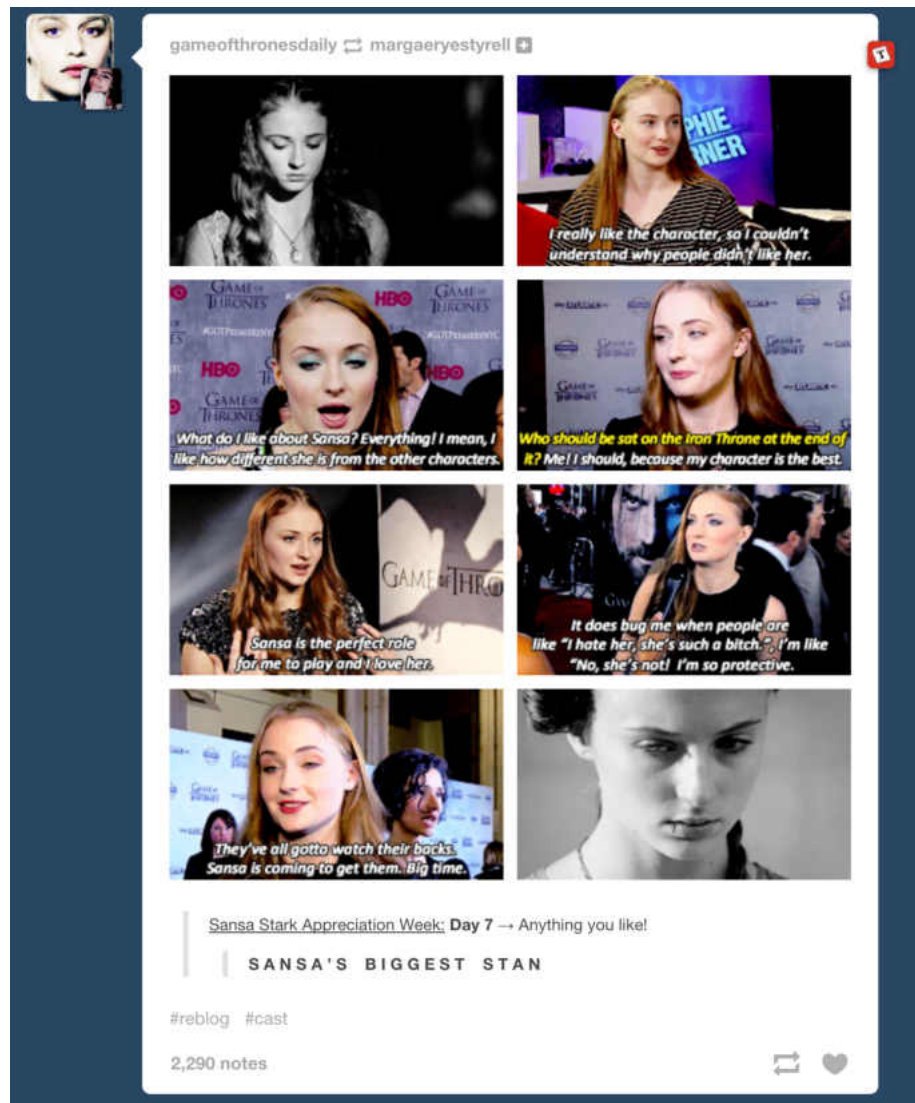
Open Tweet Filter is another extension to the Google Chrome Browser that takes keywords which specify which Tweets should be blocked from a Twitter feed [98]. Just like the Facebook Posts Filter, it prevents the tweets from showing in the Twitter feed entirely. Figure 33b shows its configuration window, displayed on top of a twitter feed. Open Tweet Filter embeds itself into the Twitter web site and is configured based on a menu option chosen from one’s Twitter home page.

TweetDeck is an application that can be installed on mobile devices and can also be accessed as a web application as shown in Figure 36a. It allows a user to specify a series of strings as shown in Figure 36b. If any tweets exist in the user’s Twitter feed that contain these strings, then those tweets will no longer appear in that user’s twitter feed. The application is rather simplistic in its string matching, blocking whole tweets based on a simple equality metric, rather than determining if the tweet itself has anything to do with the television show or book trying to be avoided.

Spoiler Shield [81] works with Facebook and Twitter to block posts that contain spoilers. Figure 34 shows Spoiler Shield blocking posts about *Game of Thrones* on Facebook. Spoiler Shield allows a user to select certain television shows, sports teams, and celebrities to avoid. Figure 35a shows the configuration screen indicating



(a) Demonstration of content blocked by Tumblr Savior on the Tumblr Web Site



(b) Demonstration of content that had been blocked by Tumblr Savior

FIG. 32: Examples of Tumblr Savior

Facebook Posts Filter

Setup the keywords to hide posts on Facebook:

Keywords

game of thrones

Enter the keywords separated by commas

Save

Help the developer to maintain this project, donate:

Donate

This project is developed by [Sergio Vilar](#)

(a) Demonstration of configuration page for Facebook Posts Filter

Filters

Excluding tweets containing terms:

#gameofthrones

Excluding tweets from people:

Show report of filtered tweets.

Clear OpenTweetFilter Settings Disable

(b) Demonstration of configuration page for Open Tweet Filter

FIG. 33: Examples of configuration screens for social media filter programs that can be used to block spoilers entirely

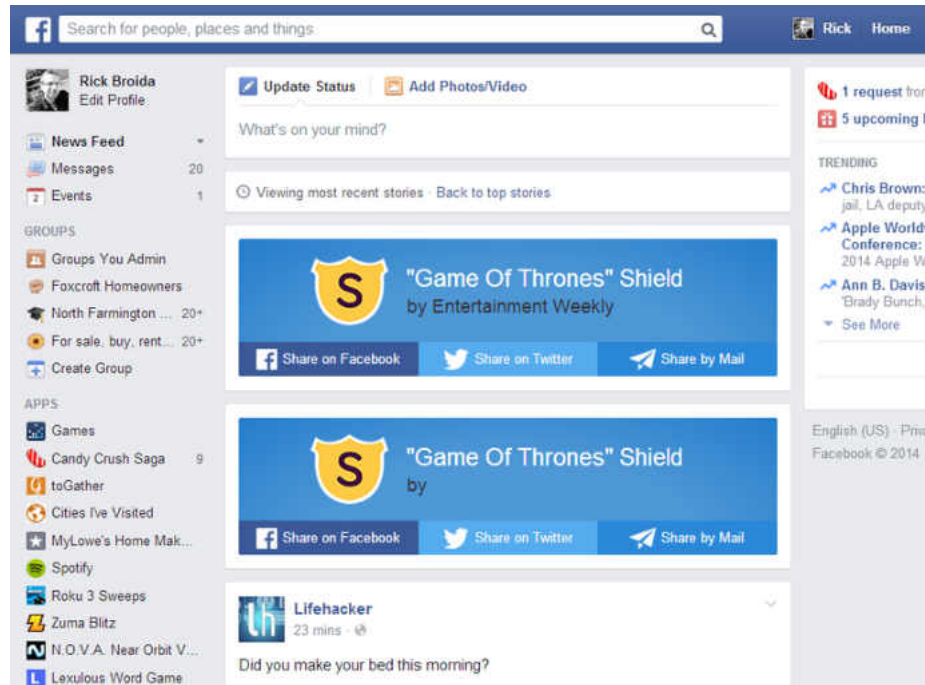


FIG. 34: Spoiler Shield For Chrome posts about *Game of Thrones* on Facebook

which categories can be chosen. Once a category is chosen, as shown in Figure 35b, we can select specific items, like the television shows, to avoid. Even though spoiler shield is still using some measure of text matching, it appears to be a little bit more intelligent than the other blocking software discussed so far, providing coverage for an entire television show's terms, rather than forcing the user to specify them all themselves.

The Netflix web site⁵ offers the Netflix Spoiler Foiler, which masks entries in a user's Twitter feed for the TV shows *House of Cards* [76] and *Breaking Bad* [77]. This application, seen in Figure 37a, is heralded as an advancement in spoiler protection [18], but does not always work. Figure 37b shows the application blocking a tweet potentially containing a spoiler for the television show *House of Cards*, but as Figure 37c shows, the actual tweet contained nothing about the television show *House of Cards*, instead espoused political commentary.

All of these applications provide the ability to block content from a user's social media feed. This is not really the problem we are trying to solve. We want to still

⁵<http://www.netflix.com>

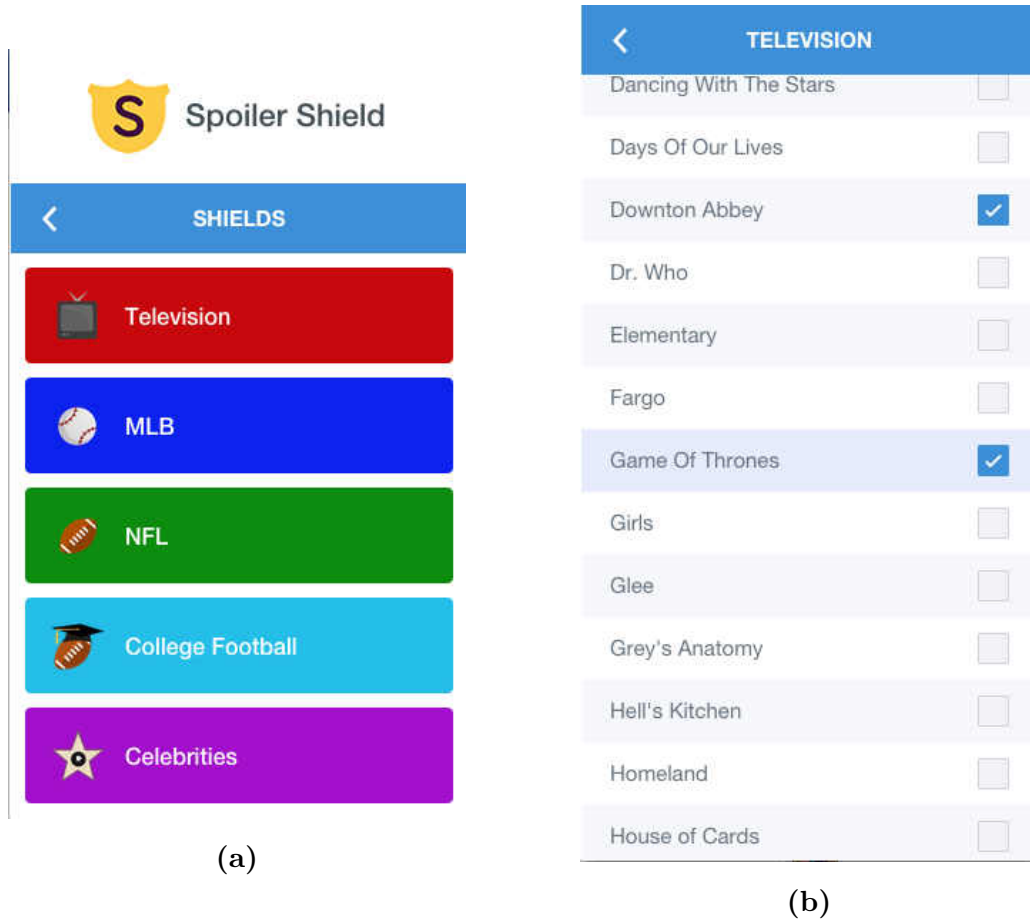
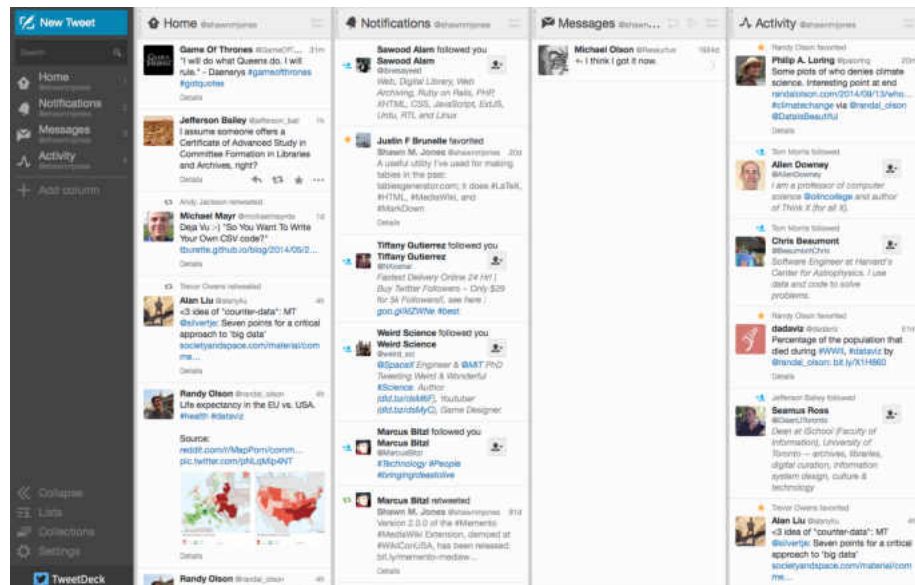
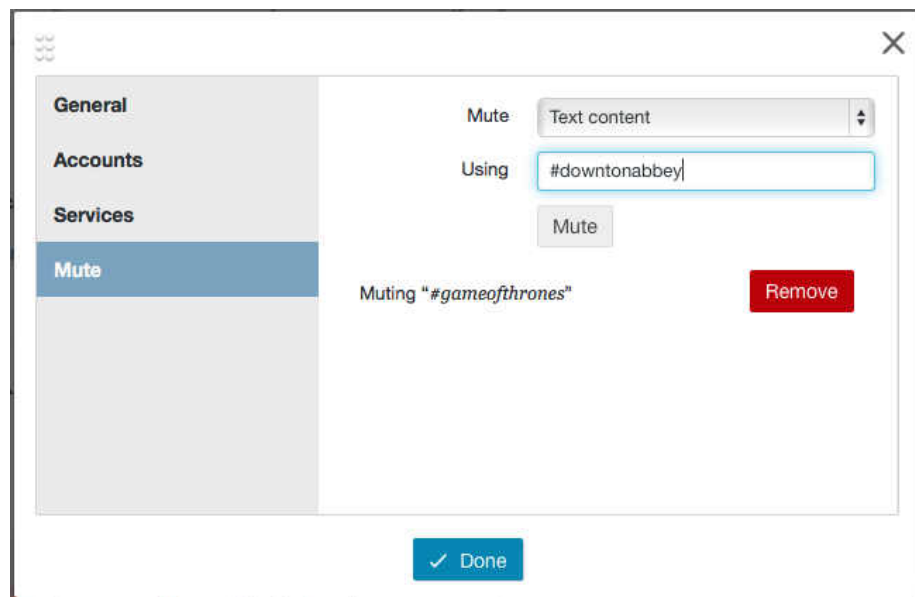


FIG. 35: Screenshots of Spoiler Shield configuration screens

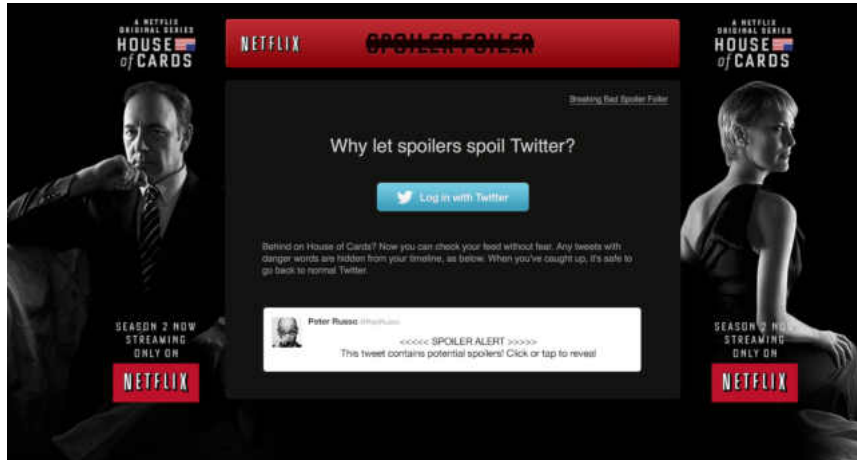


(a) The web interface for the TweetDeck web application

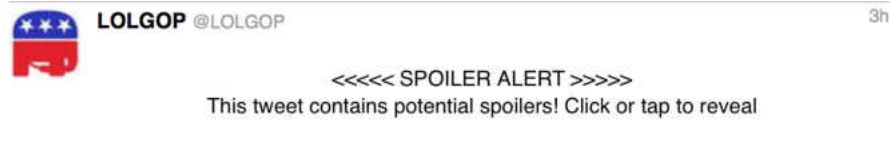


(b) The configuration screen for the TweetDeck web application, showing an attempt at avoiding spoilers for *Game of Thrones* and *Downton Abbey*

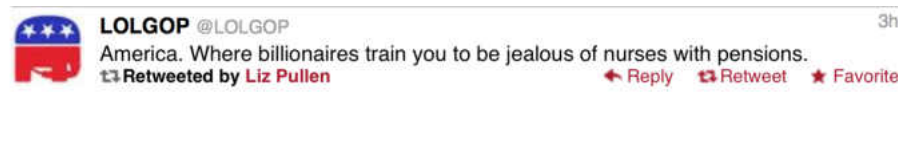
FIG. 36: Screenshots of the TweetDeck application



(a) Main page of the Netflix *House of Cards* Spoiler Foiler web application



(b) Example of the Netflix *House of Cards* Spoiler Foiler web application blocking a tweet containing a perceived spoiler



(c) Example of the tweet blocked by the *House of Cards* Spoiler Foiler web application, which does **not** contain a spoiler for *House of Cards*

FIG. 37: Screenshots of two Spoiler Foiler web applications created by Netflix

allow users to read about their fiction without getting spoiler content. Using these applications would prevent the user from getting any information about their fiction. Also, we are focusing on wikis rather than social media.

3.4 EXISTING STUDIES OF WIKIS

With the introduction of Wikipedia, a lot of interest was generated on the usefulness of wikis. Most of the research has centered on Wikipedia, as it is the largest wiki ever maintained.

Giles discusses an “expert-led investigation carried out by *Nature*” [31] in which articles from both Wikipedia and Encyclopædia Britannica were peer reviewed by experts. These experts were not told which source an article came from. Surprising at the time, they found that Wikipedia’s accuracy rate was as high as the venerable Britannica. Though there is controversy of the types of topics included in Wikipedia, such as theories not fully explored or news stories that have not been resolved, this type of currency is also deemed to be one of Wikipedia’s strengths. One of the recommendations that came out of this study was that experts contribute to Wikipedia, rather than trying to dissuade others from using it. By 2011, the inclusion of experts is still controversial [52].

This concern has spawned additional studies on the quality of articles in Wikipedia. Hu, Lim, Sun, et al. came up with several metrics for automatically evaluating Wikipedia articles, with the goal being to score each article in some way for the consumer [38]. They discovered that article length is a metric of quality, but better models exist, such as PROBVIEW, which assigns probabilities to each word having been reviewed (and maintained) by previous editors.

Almedia, Mozafari, and Cho produced one of the first studies of the behavior of contributors to Wikipedia [5]. The authors discover that there are distinct groups of Wikipedia contributors. One group, consisting of about 5000 contributors, contributes the majority of articles. They also determined that 70% of Wikipedia contributors just revise articles rather than creating new ones, thus the burden of making new articles falls to the other 30% of contributors. They suggest that as the number of articles increase, the contributors’ attention is split amongst more and more content, resulting in the larger number of revising contributors rather than article creators.

Vong, Lim, Sun, et al. have developed models of evaluating Wikipedia articles

so they can be flagged as *controversial* [105]. This way editors can focus their efforts on resolving controversies in particular articles, but also allowing others to see which controversial topics in Wikipedia are indicative of the real world controversies, allowing for further areas of study.

In 2010, Lucassen and Schraggen again evaluated the “trustworthiness” of Wikipedia articles [56]. They discuss how, by 2010, Wikipedia contains an Editorial Team that evaluates article quality and flags those articles that are considered to be of good quality and those that need work. Their contribution is a series of features that indicate how Wikipedia users evaluate articles. These features can then be used in the future for further evaluation by experts.

We highlight these studies to indicate that there has been a lot of study on what Wikipedia can be. The fan-based wikis in which we are attempting to avoid spoilers tend to be central hubs of activity for those seeking to find information on their favorite fiction. Wikipedia has undergone an evolution from completely closed to completely open to now having recommendations of articles by committee. The wiki fan sites that we have reviewed are in various stages of this evolution, depending on how large a user base they have.

Additionally, there has been some effort of preserving wiki pages outside of the Internet Archive. Popitsch, Mosser, and Phillipp have created the UROBE project for archiving wiki representations in a generic format that can then be reconstituted into many other formats for data analysis [82]. Interestingly, they anticipate attaching their process to Memento at some point later in their research so that past versions of their archives can be accessed by datetime. As of the paper’s publication, they were only preserving the content of wikis externally, albeit via a different method.

3.5 EXISTING SOFTWARE THAT PROVIDES PAST VERSIONS OF MEDIAWIKI PAGES

The Memento Project has provided support for time travel capability with Wikipedia [102], in the form of a *Wikipedia Proxy*. Figure 38 shows the use of proxy providing TimeGate functionality because Wikipedia does not natively support it. A Memento client, such as the Memento for Chrome Extension, allows the user to to query the proxy as a TimeGate. The proxy then queries Wikipedia’s web Application Programming Interface (API) to find the best memento for a given datetime. Even though this proxy exists, it is not optimal. It adds an additional HTTP

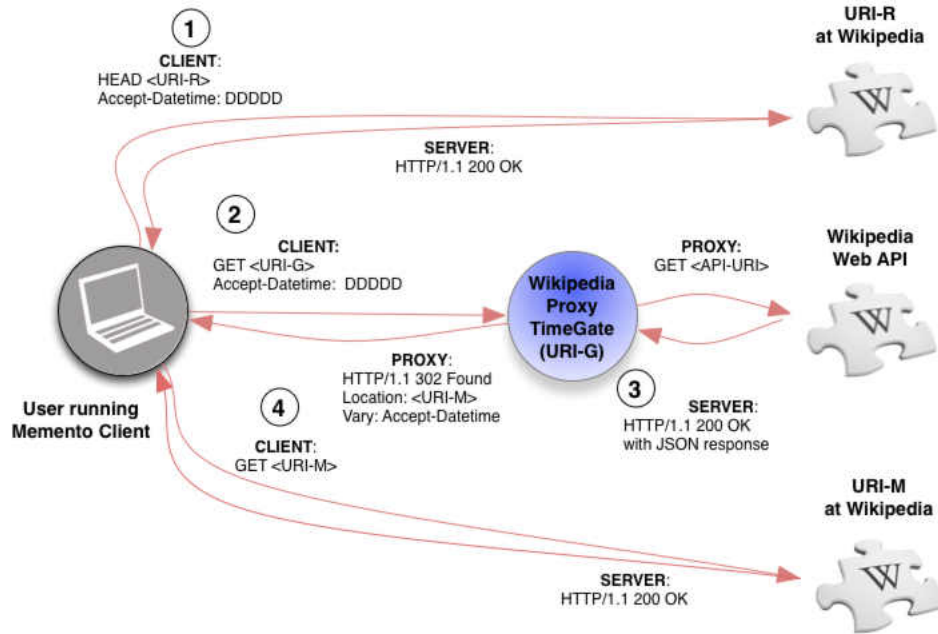


FIG. 38: High level process for the use of the Memento Wikipedia Proxy

request-response step. It also does not address our spoiler problem for all wikis. The proxy is customized for Wikipedia. Additional proxies would need to be developed for other wiki sites in order to use this solution.

Interest in time travel capability does exist in the MediaWiki community, as is evidenced by the Time Machine Extension [88]. The Time Machine Extension, shown in Figure 39 allows one to choose a date in the past to browse wiki pages. It stores the date selected in a cookie and the user must delete the cookies from their web browser in order to view the current version of wiki pages again. Though it could be used to avoid spoilers in wikis, it only works within a single wiki and provides no access to external sites or archives.

The BackwardsTimeTravelExtension provides similar capability, but is produced by a different author [13]. This extension works by supplying the date as an extra parameter to the URI. For example, if one wanted to browse the wiki on the date of April 24, 2010 at 1:00 pm, one would add the text `&epoch=20100424130000` to the URI in the browser's address bar. The goal of the BackwardsTimeTravelExtension is to faithfully reproduce a previous version of a MediaWiki page, matching the dates of the revisions of the images and other embedded content to the dates of the revisions of the main page. This is a separate, but related area of study referred to

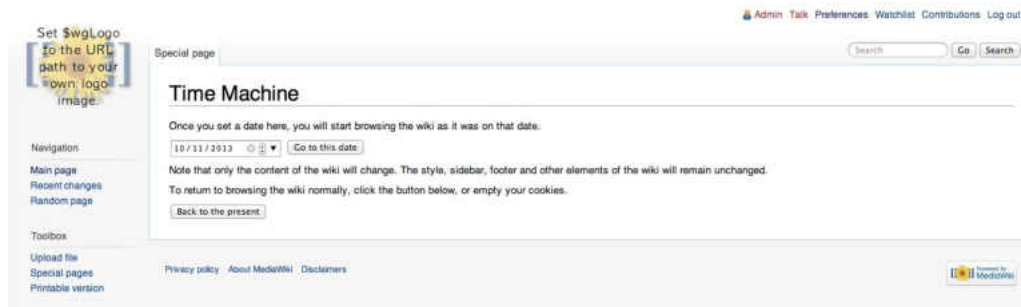


FIG. 39: The operations screen for the MediaWiki Time Machine extension

as **temporal coherence**. Like the Time Machine extension, it can be used to avoid spoilers in wikis, but does not provide seamless transition between wikis and the rest of the web.

Parsoid offers the ability to turn MediaWiki syntax into HTML documents while also attempting to preserve images, stylesheets, and other embedded content [26]. Figure 40 is a design diagram for Parsoid. It does not provide real-time access to all of the revisions of a MediaWiki page, but could conceivably be a way to archive and preserve past revisions of MediaWiki pages for posterity.

The Collection extension, is used to preserve wiki pages, with the intent of rendering them with the application mwlib [90] and preserving them in book form for physical reproduction with a service like PediaPress [79]. Figure 41a shows the configuration screen for the Collection extension, allowing a user to select certain articles from a wiki for inclusion. Figure 41b shows examples of books printed from PediaPress after the Collection extension is used to curate a wiki. This extension only works with the version of the page captured when the book is created by a user and so does not offer real-time access to all of the revisions of a MediaWiki page. It is designed as an archiving tool, but not does not provide a classification scheme, cataloguing, or finding aids for acquiring these past revisions.

One could manually perform datetime negotiation using MediaWiki’s history pages, but this is very time consuming for the individual.

As noted above, one could use the MediaWiki API to perform the functions of Memento, but only a MediaWiki-aware client could construct URIs from the data returned from the API, which would not allow a user to seamlessly avoid spoilers on both their fan wiki and the web at large.

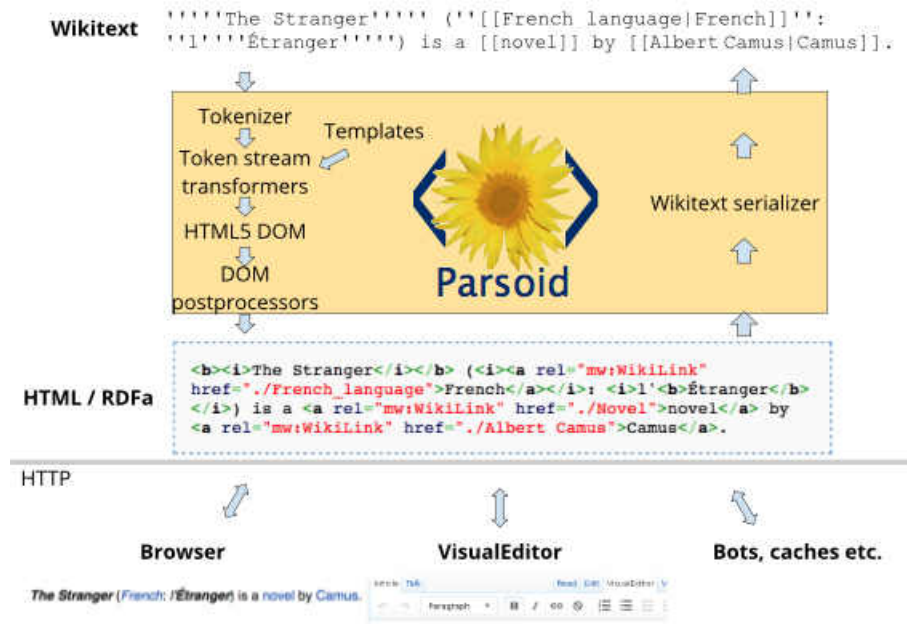


FIG. 40: Conception Diagram of the Parsoid MediaWiki application (image created by J. D. Forrester, Gabriel Wicke, and Trevor Parscal)

It is not merely useful to stay within a wiki to avoid spoilers, hence we also want the user to have the ability to utilize Memento for the rest of the web as well. Just last year, the television show *Big Bang Theory* aired an episode that actually revealed the ending to an episode of *The Walking Dead* [44]. If a user was a fan of both series, they may want to walk between both fan wikis to avoid spoilers while still reading about these shows. This is not possible with an extension that merely stays within the wiki. A lower-level protocol must be invoked, like that provided by Memento, to provide the seamless transition between both resources at the same datetime.

3.6 SUMMARY

In this section, we discussed others' attempts at identifying, analyzing, and tackling the spoiler problem. We have also looked at how others have studied wikis in the past and how previous software for wikis could have been used to address the spoiler problem. That said, we are looking for a more holistic solution that can apply across wikis and the entire web, thus we come back to Memento and how it can be used to select a specific datetime for avoiding spoilers. As we will see in the

next chapter, even Memento's ability to provide us a spoiler-free page is not without complications.

Collection
From Simple English Wikipedia - the free encyclopedia that anyone can change

You can collect articles, generate and download a PDF file from article collections, order books from a print-on-demand partner and save article collections for later use or to share them.

See [Help:Collections](#) for more information about collections.

My Collection

Title: Music Wiki Selection
Subtitle: Articles from Wikipedia

Contents
[Create new chapter] [Sort articles alphabetically] [Clear collection]

- **▼ Basic topics** [Rename] [Remove]
 - ▲ ▼ Falsetto [Remove]
 - ▲ ▼ Singing [Remove]
 - ▲ ▼ Soprano [Remove]
 - ▲ ▼ Mezzo-soprano [Remove]
 - ▲ ▼ Baritone [Remove]
 - ▲ ▼ Harpsichord [Remove]
- **▼ Composers** [Rename] [Remove]
 - ▲ ▼ Ludwig van Beethoven [Remove]
 - ▲ ▼ Johannes Brahms [Remove]
 - ▲ ▼ Wolfgang Amadeus Mozart [Remove]
 - ▲ Joseph Haydn [Remove]

Order Printed Book

You can order a printed book containing your article collection by visiting one of the following print-on-demand partners:

- [Order book from PediaPress - About PediaPress](#)

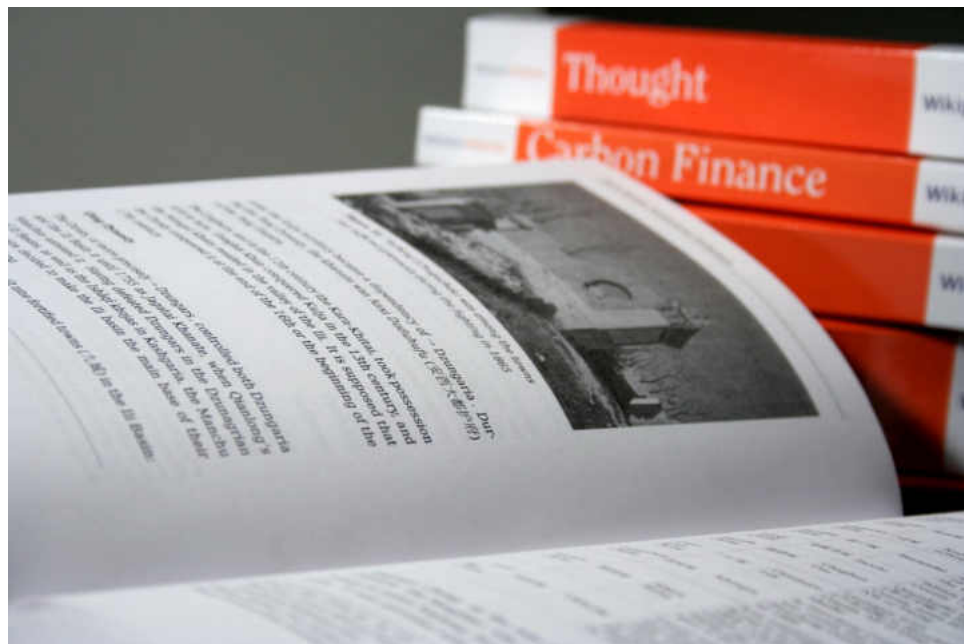
Download Collection as PDF

To download an automatically generated PDF file of your article collection, click the button.

Save Collection

If you want to save collections for later use, please [log in](#) or [create an account](#).

(a) The operations screen for the MediaWiki Collection extension



(b) Examples of books produced by PediaPress using the MediaWiki Collection extension

FIG. 41: The use and products of the MediaWiki *Collection* extension

CHAPTER 4

SURVEY OF TIMEGATE HEURISTICS

When the user selects a desired datetime prior to the episode they have not yet seen, the TimeGate is what determines which memento they are redirected to. In the case of spoilers, the wrong heuristic can redirect the user to a spoiler even though they requested a datetime prior to the event that would have caused the spoiler. Because of this possibility, we identify here several possible heuristics for use with Memento TimeGates and why some are preferred over others when avoiding spoilers.

4.1 GENERIC TIMEGATE HEURISTIC CONSIDERATIONS

Memento TimeGates accept two arguments from the user: desired datetime (specified in the Accept-Datetime header) and a URI-R; and they return the *best* URI-M using some heuristic. If we let R represent a URI-R, M represent a URI-M and t represent a desired datetime, then a TimeGate heuristic can be expressed mathematically [4] as:

$$M = \mathcal{H}(R, t) \tag{3}$$

RFC 7089 leaves the heuristic of finding the *best* URI-M up to the implementor, stating that “the exact nature of the selection algorithm is at the server’s discretion but is intended to be consistent” [101]. For avoiding spoilers, we need to consider the different heuristics and the cases where some are superior to others. For this, we modify our mathematical nomenclature from (3) as follows:

$$M = \mathcal{G}^h(R, t_a) \tag{4}$$

where h is the heuristic chosen for use, and we use the term t_a as the desired datetime to be consistent with the rest of this thesis.

Ideally, every URI-R has a corresponding URI-T referring to a TimeMap listing every URI-M and associated datetime that has been captured for this URI-R. Whatever the h chosen for use, a TimeMap is still involved in the decision making process, even though the user is not aware of it. A simplistic generic algorithm for \mathcal{G}^h from Equation (4) is listed in Algorithm 1. The goal is to find the memento with the

```

 $\mathcal{G}^h(R, t_a)$ 
1   $T = \text{GETTIMEMAP}(R)$ 
2  if  $T == \text{NULL}$ 
3      error “no timeMap for  $R$ ”
4      return  $\text{NULL}$ 
5   $m = \text{GETNEXTMEMENTOFROMTIMEMAP}(T)$ 
6   $\text{mincost} = \infty$ 
7   $\text{bestm} = ""$ 
8  while  $m \neq \text{NULL}$ 
    // loop through the TimeMap until we run out of mementos
9       $m.\text{cost} = C(m, t_a, b_L, b_u)$ 
10     if  $m.\text{cost} < \text{mincost}$ 
11          $\text{mincost} = m.\text{cost}$ 
12          $\text{bestm} = m$ 
13      $m = \text{GETNEXTMEMENTOFROMTIMEMAP}(T)$ 
14 return  $\text{bestm}$ 

```

Algorithm 1: Generic algorithm for a TimeGate

lowest cost *relationship* to the desired datetime t_a . The different heuristics h that we will discuss in subsequent sections indicate the type of relationship that we prefer to evaluate.

This algorithm is relatively simple. On line 1, it acquires the TimeMap for R . In this case a TimeMap is the list of all mementos for a given URI-R. On line 2, it checks for the existence of a TimeMap for R . If none exists, then it returns with an error, continuing otherwise. On line 5, it acquires the first memento from a generator function `GETNEXTMEMENTOFROMTIMEMAP`, it is assumed that `GETNEXTMEMENTOFROMTIMEMAP` returns each memento from a TimeMap in the order of oldest to newest. On line 8, it begins the loop through all of the mementos in the TimeMap. It uses the *cost function* C on line 9 to compute the cost of memento m in comparison to desired datetime t_a . We will discuss C below. The comparison on line 10 just determines if we have found a minimum cost that is

$$C(m, t_a, b_L, b_u) = \begin{cases} \infty & \text{if } \neg(b_L \leq m_T \leq b_u) \\ 0 & \text{if } (m_T = t_a) \wedge (b_L \leq m_T \leq b_u) \\ |t_a - m_T| & \text{if } (m_T < t_a \vee m_T > t) \wedge (b_L \leq m_T \leq b_u) \end{cases} \quad (5)$$

lower than a previously encountered minimum; if so, we store it as the new minimum. Line 13 gets a memento for the next run. By line 14, we have found the memento with the minimum cost and will return it. The returned memento contains attributes that can be accessed, such as *bestm.uri* for URI-M and *bestm.datetime* for Memento-Datetime. Assuming C and `GETNEXTMEMENTOFROMTIMEMAP` can run in constant time, this algorithm runs in $O(n)$ time, where n is the number of mementos in the TimeMap.

Our cost function C uses m as the memento to be evaluated, t_a as the desired datetime, b_L as the lower bound datetime for the mementos under consideration, and b_u as the upper bound datetime for the mementos under consideration. It is the values of each of these parameters that determines, in most cases, which \mathcal{G}^h is used.

Also, under consideration in each case is also m_T for the Memento-Datetime of the memento, and m_L as the Last-Modified datetime of the memento, if it exists.

Equation (5) shows this cost function. This equation has three cases.

In the first case, if the memento datetime m_T exists outside the bounds of b_L and b_u , then we do not want it to be considered, hence we make the cost ∞ . Line 10 of Algorithm 1 shows the comparison for this case.

The second case is simple. If, per chance, the desired datetime t matches the memento-datetime m_T under consideration, and we fall within the range of $b_L \dots b_u$, return 0. It is a simple, no cost case because the user is getting exactly what they asked for.

Finally, if all other cases have not been met, we use the absolute value of the difference between the memento-datetime m_T and the desired datetime t_a as the cost.

In the sections below, we discuss how the parameters of this cost function can be altered for each TimeGate heuristic, producing different results. These heuristics are important to avoiding spoilers because some heuristics do not reliably protect us from spoilers.

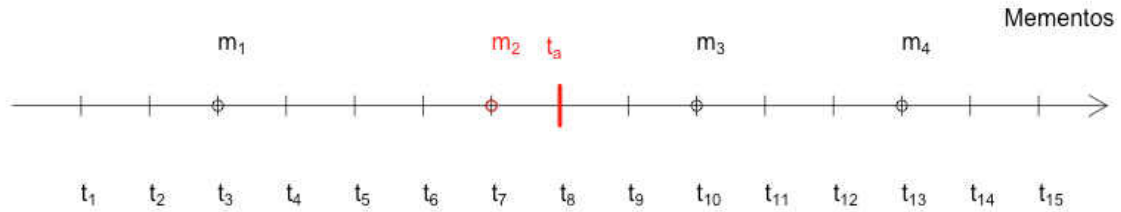


FIG. 42: Demonstration of the *mindist* heuristic, in this case $m_2@t_7$ is chosen because it is closest to t_a

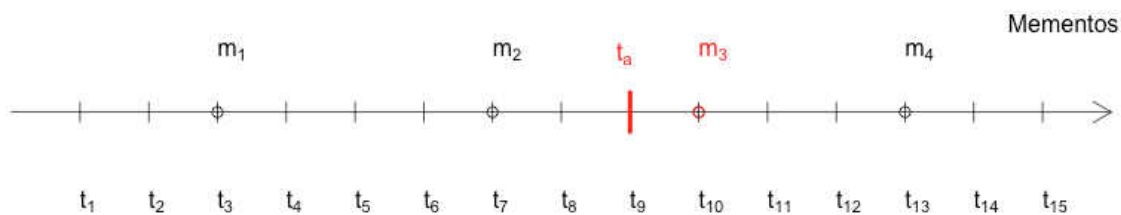


FIG. 43: Demonstration of the *mindist* heuristic; in this case $m_3@t_{10}$ is chosen because it is closest to t_a

4.2 TIMEGATE HEURISTICS UNDER CONSIDERATION

This section discusses those heuristics that determine the best memento to be returned. This not a complete list, as new heuristics are being explored as new use cases for Memento arise, but it provides us with a list of heuristics to compare and contrast for use in avoiding spoilers.

Note that the choices made by these heuristics are based on structured metadata as opposed to a review of content, quality of the memento, or other factors.

4.2.1 CLOSEST (MINDIST)

Closest, or **mindist**, finds the closest memento to the given desired datetime t . Closest uses the cost function, as shown in Equation (6). We used the value of ∞ for the parameters b_L and b_u because we want to evaluate the cost across the entire

TimeMap.

$$C(m, t_a, \infty, \infty) \tag{6}$$

Figure 42 shows the selected memento from an example TimeMap of four mementos. In this case t_7 has the minimum distance from desired datetime $t_a@t_8$ and hence $\mathcal{G}^{mindist}(R, t_8) = m_2@t_7$ is the memento returned using the mindist heuristic.

Alternatively, Figure 43 shows the selected memento using the same example TimeMap. In this case $t_a \equiv t_9$. This results in $\mathcal{G}^{mindist}(R, t_9) = m_3$ because $m_3@t_{10}$ is the closest memento to t_9 .

Mindist is best used for web archives, which are typically sparse, meaning they may have missed many revisions of a page. In this case, a user would want the closest memento they can get to the date they are requesting because the dates of capture may be wildly distant from one another.

Consider the example where a memento was captured from a URI-R on September 23, 2004 and a second was captured on October 7, 2009. Now, let the user choose a desired datetime of October 1, 2009. Because we do not have very many mementos to choose from, the October 7, 2009 memento is *best* in this case because it is most likely to represent the general time period the end user was looking for.

Because of the fact that it may choose mementos from a date after the desired datetime, mindist is not a reliable heuristic for avoiding spoilers.

4.2.2 CLOSEST, BUT NOT AFTER (MINPAST)

Closest, but not after, which we will refer to as **minpast**, finds the closest memento to the desired datetime t_a , but without going over t_a .

To achieve minpast, one alters the cost function C as shown in Equation (7). The values of m and t_a are unchanged, but we do provide the upper bound for value b_u as t_a and leave the lower bound b_L set to ∞ as in mindist.

$$C(m, t_a, \infty, t_a) \tag{7}$$

This forces the cost function to return ∞ for any memento evaluated with a datetime *after* t_a , meaning that even if a memento after t_a has the minimum distance, it will still not be considered.

Figure 44 shows an example TimeMap with four mementos. The value of t_a is set to t_8 . In this case $m_2@t_7$ is the closest memento that does exceed t_8 , so $\mathcal{G}^{minpast}(R, t_8) = m_2@t_7$.

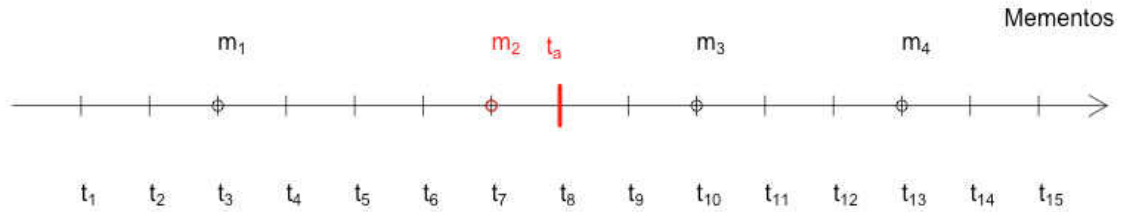


FIG. 44: Demonstration of the *minpast* heuristic, in this case $m_2@t_7$ is chosen because it is closest, but not greater than, t_a

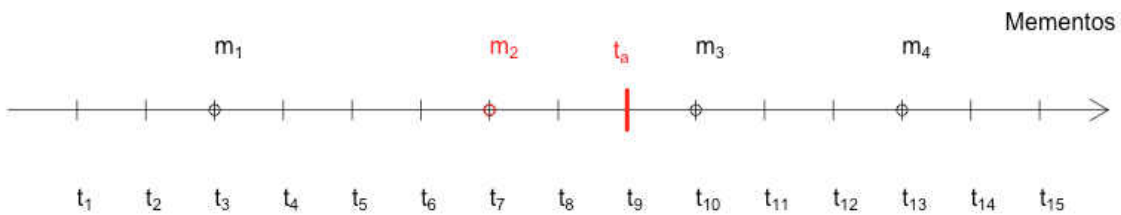


FIG. 45: Demonstration of the *minpast* heuristic, in this case $m_2@t_7$ is still chosen because it is closest, but not greater than, t_a , even though $m_3@t_{10}$ has the minimum distance

Figure 45 shows the same example TimeMap. In this case, the value of t_a is set to t_9 , resulting in a different effect than *mindist*. Here we get the same result as the last example: $\mathcal{G}^{minpast}(R, t_9) = m_2@t_7$. Even though $m_3@t_{10}$ is closer to t_9 , t_{10} exceeds t_9 , so it cannot be considered by *minpast*.

Minpast is best used for archives that are abundant with mementos. Ideally, *minpast* should be used if every revision of a resource has been archived, as with wikis. For wikis, the value of desired datetime t_a corresponds to a revision that actually existed at the time of t_a . For web archives that are not abundant, information may be lost because they may not have captured all revisions.

Consider the example with the following mementos from a URI-R: April 20, 2010; April 21, 2010; and April 24, 2010. With a wiki, let each of these mementos be a page revision. If we let the value of the desired datetime be April 23, 2010, then we know that the memento from April 21, 2010 is the actual revision of the page as it

looked on April 23, 2010, because there were no changes until April 24.

Now consider the same mementos, but the resource is not a wiki, so these mementos are not page revisions, but captures in a web archive. If we still use our desired datetime of April 23, 2010, and we use the minpast heuristic, then we will get the memento from April 21, 2010; but we will not know if we missed a useful revision on April 23 or April 22. It might be that April 24 is a better match depending on what information the user was searching for.

Also, consider the case, as mentioned in the last section, where the distance between t_a and the closest memento produced by minpast is vast. Is minpast best in that case? If we do not know the state of the resource at the time of t_a , as in web archives, then it is more likely that the user will benefit from *mindist* than *minpast* because they will get a memento close to the time period they wish to view.

Minpast can be used to avoid spoilers. If we select a value for t_a prior to the event we want to avoid, then minpast will not find any mementos after t_a . It is best used for wikis where we have access to all revisions because we can definitively state that the memento returned is the page as it existed at t_a .

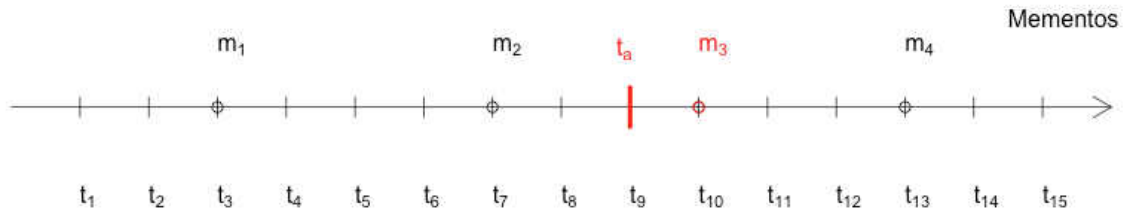


FIG. 46: Demonstration of the *minfutr* heuristic, in this case $m_3@t_{10}$ is chosen because it is closest, but not less than, t_a

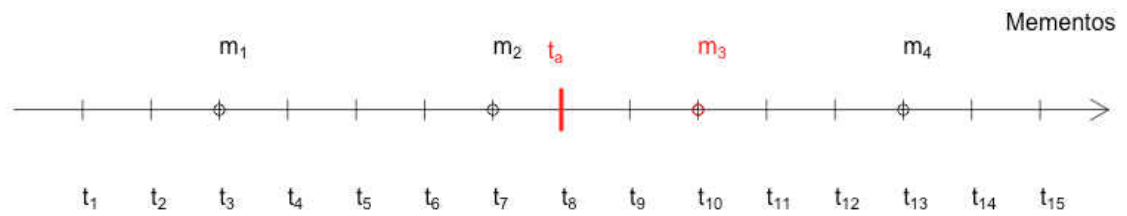


FIG. 47: Demonstration of the *minfutr* heuristic, in this case $m_3@t_{10}$ is still chosen because it is closest, but not less than, t_a , even through $m_3@t_7$ has the minimum distance

4.2.3 CLOSEST, BUT NOT BEFORE (MINFUTR)

Closest, but not before, which we refer to as **minfutr**, finds the closest memento to the desired datetime t , but without considering any datetimes prior to t . It is the opposite of minpast.

To achieve minfutr, the cost function is constructed as shown in Equation (8). The values of m and t_a are unchanged, but we provide a *lower* bound for the value of b_L as t_a and leave the upper bound b_u set to ∞ as in mindist.

$$C(m, t_a, t_a, \infty) \tag{8}$$

This forces the cost function to return ∞ or an undefined value for any memento evaluated with a datetime *before* t_a , meaning that even if a memento before t_a has the minimum distance, it will still not be considered.

Figure 46 shows which memento is chosen for $t_a = t_9$. In this case $\mathcal{G}^{minfutr}(R, t_9) = m_3@t_{10}$ because m_3 is the closest memento to t_a that does not precede t_a . This result is no different than mindist.

Figure 47 shows what happens when we change the value of t_a to t_8 . Here we see $\mathcal{G}^{minfutr}(R, t_8) = m_3@t_{10}$ because, even though $m_2@t_7$ is closer to $t_a@t_8$, it comes before t_8 and thus cannot be considered by minfutr.

Like minpast, minfutr is best used with archives abundant in mementos. Minfutr also works best with wikis because one can use it to find the first occurrence of an article after a given date. Consider the opposite of the spoiler problem, where one wants to find the first published representation of an article after the episode has been released. In the case of the wiki, one can reliably answer this request. In the case of a sparse web archive, one can possibly get a memento that is years away from the desired datetime, and not reliably find the first published representation.

Minfutr, in an abundant web archive, can also be used to determine the first reported case of an event, such as a news story (e.g., first memento of a post-9/11 world) or a case of disease (e.g., first reported complaint of seasonal flu). Minfutr can be used to backtrack an event to find the source of information, allowing researchers to study the flow of information through the web over time. Of course, such research is again only possible in an abundant archive, or a content management system that preserves past representations, such as a wiki.

Minfutr, by its very nature, cannot be used to avoid spoilers. It is best for those looking for the first incident of spoilers, or for the reaction after some event.

4.2.4 CLOSEST, BUT BOUNDED (MINNEAR)

Minnear is a generic use of the bounds provided by b_L and b_u . It is possible that these bounds may be placed for machine performance reasons or due to connectivity problems for certain URIs in the TimeMap. Whatever the reason, minnear merely places bounds around the mementos to be considered, much like minpast and minfut, but for bounded values other than t_a .

Minnear uses the cost function in its pure form, as shown in Equation (9).

$$C(m, t_a, b_L, b_u) \tag{9}$$

Depending on the values of b_L and b_u , minnear still falls back to the minimum distance metric for determining the best memento, and hence may provide spoilers for a given value of t_a , making it unreliable for avoiding them.

Minnear presents an opportunity for additional heuristics to be generated from these parameters, each for special cases. For this reason, we will not go into more detail with minnear at this time, because it is an opportunity for future work.

4.2.5 EQUAL, BUT NOT AFTER (EQPAST)

Equal, but not after, referred to as **eqpast**, is like minpast, but the exception is that the content of the mementos is considered in the decision-making process.

If two mementos exist on either side of the desired datetime t_a , then this heuristic would compare their content, and, if equal, choose the one from the past.

An example algorithm for eqpast is shown in Algorithm 2. This heuristic uses minpast and minfut to find the mementos on either side of t_a , then it determines if their content matches. If their content is a match, then it returns the one given by minpast. If they do not match, then it falls back to mindist. This algorithm uses minpast and minfut which utilize the cost function on all mementos in the TimeMap and hence run in $O(n)$ time, but we also need to perform equality on two representations, consisting of c characters (or bytes); and must iterate through all of those characters to determine equality. Thus, the running time is $O(n + c)$.

Eqpast is currently used for studies in evaluating **temporal coherence**, in which archived pages contain embedded resources whose archived mementos must also match the same time period as the memento that embedded them. If we know that the representation on either side of t is the same, then one can improve the performance of web archives by only storing one of each resource.

```

 $\mathcal{G}^{eqpast}(R, t_a)$ 
1   $T = \text{GETTIMEMAP}(R)$ 
2  if  $T == \text{NULL}$ 
3      error “no timeMap for  $R$ ”
4      return  $\text{NULL}$ 
5   $leftm = \mathcal{H}^{minpast}(R, t)$ 
6   $rightm = \mathcal{H}^{minfutr}(R, t)$ 
7  if  $leftm.content == rightm.content$ 
8      return  $leftm$ 
9  return  $\mathcal{H}^{mindist}(R, t)$ 

```

Algorithm 2: Example eqpast algorithm for TimeGate

4.2.6 EQUAL, BUT NOT BEFORE (EQFUTR)

Equal, but not before, referred to as **eqfutr** is like eqpast, but it returns the memento greater than the desired time t .

Algorithm 3 shows an example of how this heuristic might be implemented. Like eqpast, it also uses minpast and minfutr to find the mementos on either side of t , and then determines if their content matches. If they match, then it returns the memento given by minfutr. If they do not, it falls back to mindist. The worst case running time is still $O(n + c)$ just like eqpast.

Eqfutr is also used to evaluate temporal coherence and is also used to study methods of improving the performance of web archives.

4.2.7 SIMILAR, BUT NOT AFTER (SIMPAST)

Similar, but not after, referred to as **simpast**, is like eqpast, but instead of comparing the content of each memento for complete equality a similarity function is used. The result of the similarity function is evaluated against some *threshold* value indicating the level of acceptable similarity between the two mementos.

Algorithm 4 shows an example algorithm for simpast. The function SIMILARITY provides a similarity measure for both mementos on either site of the desired datetime


```

 $\mathcal{G}^{eqfutr}(R, t)$ 
1   $T = \text{GETTIMEMAP}(R)$ 
2  if  $T == \text{NULL}$ 
3      error “no timeMap for  $R$ ”
4      return  $\text{NULL}$ 
5   $leftm = \mathcal{H}^{minpast}(R,t)$ 
6   $rightm = \mathcal{H}^{minfutr}(R,t)$ 
7  if  $leftm.content == rightm.content$ 
8      return  $rightm$ 
9  return  $\mathcal{H}^{mindist}(R, t)$ 

```

Algorithm 3: Example eqfutr algorithm for TimeGate

t . If the similarity measure returned is less than or equal to the *threshold* value, then the one with a datetttime less than t is chosen. If the similarity measure returned is greater, then we fall back to mindist.

It is difficult to determine specifically what the worst case running time is for this algorithm, because it depends on the worst case running time of the similarity function. The common similarity function shown in equation 10 is called **cosine similarity** [16].

$$\text{Cosine}(D, Q) = \frac{\sum_{j=1}^t d_j \cdot q_j}{\sqrt{\sum_{j=1}^t d_j^2 \cdot \sum_{j=1}^t q_j^2}} \quad (10)$$

Here, D and Q both represent the documents to be compared. The terms d_j and q_j represent the corresponding features (such as the occurrence of each word) to be compared between the two documents. If we consider f to be the number of features compared between the documents, and we consider a computer program executing this formula, then we would likely have a loop to calculate the numerator, executing in time $O(f)$ and another series of loops to calculate the denominator, also executing in time $O(f)$, leading to an overall worst case execution time of $O(f)$ for this similarity function.

```

 $\mathcal{H}^{simpast}(R, t)$ 
1   $T = \text{GETTIMEMAP}(R)$ 
2  if  $T == \text{NULL}$ 
3      error "no timeMap for  $R$ "
4      return  $\text{NULL}$ 
5   $leftm = \mathcal{H}^{minpast}(R,t)$ 
6   $rightm = \mathcal{H}^{minfutr}(R,t)$ 
7  if  $\text{SIMILARITY}(leftm.content, rightm.content) \leq \text{threshold}$ 
8      return  $left$ 
9  return  $\mathcal{H}^{mindist}(R, t)$ 

```

Algorithm 4: Example simpast algorithm for TimeGate

So, if it takes $O(n)$ to execute minpast and minfutr, and $O(f)$ to execute the similarity measure, then, if we use cosine similarity, then we get a worst-case execution time of $O(n + f)$. Furthermore, if each word in the document is a feature, then it is possible that more words exist in the document than there are mementos in the TimeMap, leading to $O(f)$ as the worst case running time.

Using alternative similarity measures often use at least a subset of the words in the document, still leading to a running time of $O(n + f)$ or $O(f)$.

It is for this reason that running simpast is an expensive operation and is only used in research applications.

4.2.8 SIMILAR, BUT NOT BEFORE (SIMFUTR)

Similar, but not before, referred to as **simfutr** is just like simpast, except that the memento with a datetime greater than the desired datetime t is chosen if the two mementos have a similarity score less than the *threshold* value.

Algorithm 5 shows an example algorithm for simfutr. It is just like simpast until line 8, where the result of minfutr is used instead of the result of minpast. Because we are using a similarity measure, just like with simpast, the best case running time is $O(f + n)$, making it just as expensive as simpast, and more expensive than any other algorithm discussed.

```

 $\mathcal{H}^{simfutr}(R, t)$ 
1   $T = \text{GETTIMEMAP}(R)$ 
2  if  $T == \text{NULL}$ 
3      error "no timeMap for  $R$ "
4      return  $\text{NULL}$ 
5   $leftm = \mathcal{H}^{minpast}(R,t)$ 
6   $rightm = \mathcal{H}^{minfutr}(R,t)$ 
7  if  $\text{SIMILARITY}(leftm.content, rightm.content) \leq \text{threshold}$ 
8      return  $right$ 
9  return  $\mathcal{H}^{mindist}(R, t)$ 

```

Algorithm 5: Example simfutr algorithm for TimeGate

4.3 CONCLUSIONS FOR AVOIDING SPOILERS

Now that we have evaluated the heuristics and example algorithms for TimeGates, we can determine which are most useful for avoiding spoilers.

Table 7 shows a listing of each heuristic encountered and some of the information we have discussed. Each heuristic has been listed for comparison. As we can see eqfutr, eqpast, simfutr, and simpast all require the use of the content of the memento before making a decision.

Those using algorithm 1 only have a running time of $O(n)$. The eqfutr, eqpast, simfutr, and simpast heuristics are more expensive because they require comparing two documents. It is probable that eqfutr and eqpast have a slight edge over simfutr and simpast because they compare equality of strings only, whereas simfutr and simpast require a features comparison.

We have also evaluated them for their ability to reliably avoid spoilers. To reliably avoid spoilers, a heuristic must never return a memento after the desired datetime t_a .

Mindist is not spoiler safe. As noted before, a user can request a memento at desired datetime t and get a memento that exists after t_a because the distance between t_a and its memento-datetime is shorter than that of a memento prior to t_a .

TABLE 7: Summary of TimeGate Heuristics

Heuristic	Uses cost metric	Uses content of mementos	Potential running time	Reliably avoids spoilers?
mindist	X		$O(n)$	no
minpast	X		$O(n)$	yes
minfutr	X		$O(n)$	no
minnear	X		$O(n)$	no
eqfutr	X	X	$O(n + c)$	no
eqpast	X	X	$O(n + c)$	no
simfutr	X	X	$O(f + n)$	no
simpast	X	X	$O(f + n)$	no

It is possible that spoilers can be avoided, but this is largely dependent on how often mementos are captured for the resource.

Minpast is spoiler safe. As noted above, minpast uses t_a as an upper bound on which mementos to consider when returning one. This means that any memento with a datetime after t_a will automatically receive a score of ∞ or undefined, removing them from consideration.

Minfutr is not spoiler safe. By very definition, it uses t_a as a lower bound on which mementos to consider. This means any memento with a datetime before t_a will receive a score of ∞ or undefined, removing them from consideration. Seeing as we are looking to avoid spoilers by finding mementos prior to t_a and this uses t_a as a lower bound, there is no way to acquire a memento prior to t_a , thus minfutr can not be spoiler safe.

Minnear is not spoiler safe. Unless the upper bound of the cost function is set at t , there is always a chance of acquiring a temporal revision after t_a .

Eqfutr is not spoiler safe. It evaluates two temporal revisions of a given resource, one before t_a and one after t_a . If they have *the exact same content*, then it returns the one after t_a . Because they have the exact same content, there is no probability of the memento after t_a containing information that occurred in the episode we are trying to avoid. One would think that this means we could avoid spoilers, but eqfutr falls back to using mindist if the two mementos on either side of t_a do not have the same content. Seeing as mindist is not spoiler safe, we cannot *reliably* avoid spoilers

with eqfutr.

Eqpast is not spoiler safe. It also evaluates two mementos of a given resource, one before t_a and one after t_a . If they have *the exact same content*, then it returns the one before t_a . Like minpast, it effectively has an upper bound of t_a . Unlike minpast, however, it falls back to using mindist if the two mementos on either side of t_a do not have the same content. Seeing as mindist is not spoiler safe, we cannot *reliably* avoid spoilers with eqpast.

Simfutr is not spoiler safe. Like eqfutr and eqpast, it evaluates two mementos on either side of t_a . The exception is that it uses a similarity metric on their content rather than pure equality. Even though the two mementos may be similar enough for the threshold desired, it is still possible that spoiler information may be contained in the one chosen from the future of t_a , thus it is not *reliably* spoiler safe. Also, simfutr falls back to mindist, which is not spoiler safe.

Simpast is not spoiler safe. Like simfutr, simpast uses a similarity metric to evaluate two mementos on either side of t_a . Even though simpast effectively has an upper bound of t_a if both mementos have a similarity metric under the desired threshold, it will still fall back to mindist if their similarity is above the desired threshold. Seeing as mindist is not spoiler safe, we cannot *reliably* avoid spoilers with simpast.

Conceivably, one could replace the fallback heuristic in eqpast, eqfutr, simfutr, and simpast with minpast and make those heuristics spoiler safe. If we did that then minpast would still be preferred because its algorithm runs in $O(n)$ time while the others are slower.

So, out of the heuristics considered, only minpast is reliably spoiler safe and efficient enough for consideration. This discussion and evaluation is necessary because, as we will see in the following chapters, TimeGates for wikis can use minpast whereas TimeGates for web archives tend to use mindist because archives do not contain every memento. This means that we can avoid spoilers in wikis if the TimeGate heuristic used for them is minpast.

CHAPTER 5

SPOILER AREAS CREATED BY MINDIST

Now that we have a vocabulary of heuristics to work with, we can study the differences between the minpast and mindist heuristics as they apply to wikis. As noted in section 2.5, wikis, and some other content management systems, are effectively a form of archive because they keep every revision of a page. As noted in section 4.2.1, web archives use mindist because they are sparse in comparison to wikis and contain missed updates for the pages they archived.

Remember, as noted in section 2.6, we want to allow the user to avoid spoilers in episode e_i by selecting a datetime $t_a < t_{e_i}$. How does the use of the mindist heuristic in web archives impact our ability to avoid spoilers on the web?

By studying mindist using wiki revisions and the mementos corresponding to them, we find out that it is not just *possible* but also *probable* that one can encounter a spoiler for a given resource in web archives.

5.1 SPOILER AREAS CREATED BY MINDIST HEURISTICS

As noted before, the use of *mindist* can lead to cases where a memento containing spoilers is returned to the user, even though the user selects a datetime prior to the episode they have not seen yet. The set of datetimes where the user is redirected to a memento after the episode, even though they chose a datetime prior to the episode is defined as a **spoiler area**.

The set of datetimes where the user is directed to a spoiler, even though they chose a datetime prior to the episode they are avoiding, and where the web archive has not yet started archiving the resource, is referred to as a **pre-archive spoiler area**. Figure 48 shows two pre-archive spoiler areas. This spoiler area is created if the user tries to select a datetime prior to episode $e_3@t_{11}$, but the mindist heuristic delivers them to $m_1@t_{14} \equiv r_j@t_{13}$, which is after $e_3@t_{11}$. The user intended to avoid spoilers for episode e_3 , but got them nonetheless.

Still referring to Figure 48, it should also be noted that the pre-archive spoiler area for episode e_3 stretches from the first episode e_1 to just prior to e_3 . Also, the

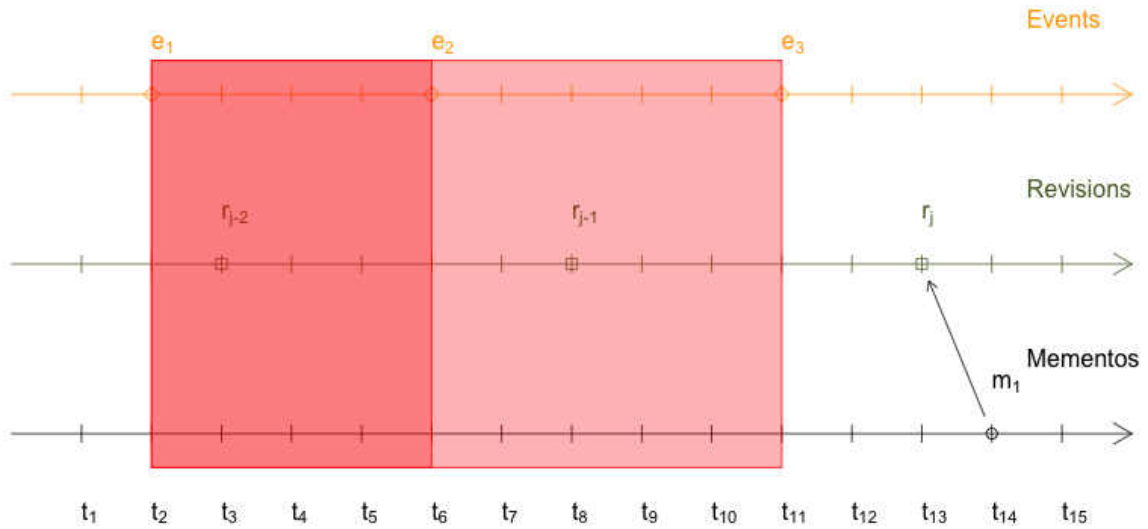


FIG. 48: Example of *pre-archive spoiler areas* (shown in light red) created using the mindist heuristic; the overlap of the spoiler areas for episodes e_3 and e_2 is shown in darker red.

pre-archive spoiler area for episode e_2 stretches from the first episode e_1 to just prior to e_2 . This means that the spoiler area for e_3 includes the spoiler area for e_2 .

What makes this a pre-archive spoiler area?

Remember the cost calculation as part of mindist, covered in Chapter 4. If a user picks a datetime prior to the first memento, then the cost between the first memento and that datetime is the lowest out of all mementos in the TimeMap, leading them to the first memento. Also, mementos are always created after wiki revisions, so mementos in the web archive will always be *late* by comparison. Combine this with missed updates, and the pre-archive spoiler area can be quite large.

So, for a pre-archive spoiler area to exist, the following conditions must be present:

1. The TimeGate for the resource uses the mindist heuristic
2. We have access to all revisions of a given resource
3. The memento-datetimes times for all revisions of a resource are defined and known
4. Event e must occur prior to the first memento recorded in the archive

5. Event e must occur prior to revision r_i corresponding to the first memento m_1 (i.e., $r_i \equiv m_1 \wedge t_e < t_{r_j}$)

Given episodes e_1 to e_i , which occur just prior to the first archived revision $r_j \equiv m_1$, this gives us the definition of a pre-archive spoiler area for episode e_i defined by function \mathcal{S}_a over the interval t_s and ending at finish datetime t_f produced by Equation (11).

$$[t_s, t_f] = \mathcal{S}_a(e_i) = \begin{cases} (t_{e_1}, t_{e_i}) & \text{if } t_{e_i} < t_{r_j} \wedge r_j \equiv m_k \\ (0, 0) & \text{otherwise} \end{cases} \quad (11)$$

So, any datetimes prior to the first memento lead to a pre-archive spoiler area. This is somewhat understandable, seeing as it takes time for the web archive to learn about a resource and start archiving it. What about once archiving has occurred?

Figure 49 shows an **archive-extant spoiler area**. Let a user select a datetime prior to $e_i@t_{11}$. To avoid spoilers, the user needs to be directed to memento m_{k-1} corresponding to revision r_{j-1} .

Unfortunately, if the user selects a datetime in the area between t_9 and $e_i@t_{11}$, mindist will deliver them memento $m_j@t_{13}$, even though they chose a datetime prior to t_{11} . Memento $m_j@t_{13} \equiv r_j@t_{12}$, which is *after* the datetime t_{11} that the user was trying to avoid. Because the user chose a datetime prior to the episode containing spoilers, but the user is redirected to a memento containing spoilers anyway.

Why is this a spoiler area? Remember that mindist finds the minimum distance between the time t_a specified by the user and any given memento. In Figure 49, we have mementos $m_{k-1}@t_5$ and $m_k@t_{13}$. We denote the midpoint between mementos as h (for halfway). The midpoint between $m_{k-1}@t_5$ and $m_k@t_{13}$ is $h@t_9$, calculated as shown in Equation (12). This means that any value t_a such that $t_9 < t_a < t_{13}$ will produce memento m_j and any value t_a such that $t_a < t_9$ will produce memento m_{j-1} .

$$t_h = \frac{t_i + t_j}{2} \quad (12)$$

So, for a archive-extant spoiler area to exist, the following conditions must be present:

1. The TimeGate for the resource uses the mindist heuristic

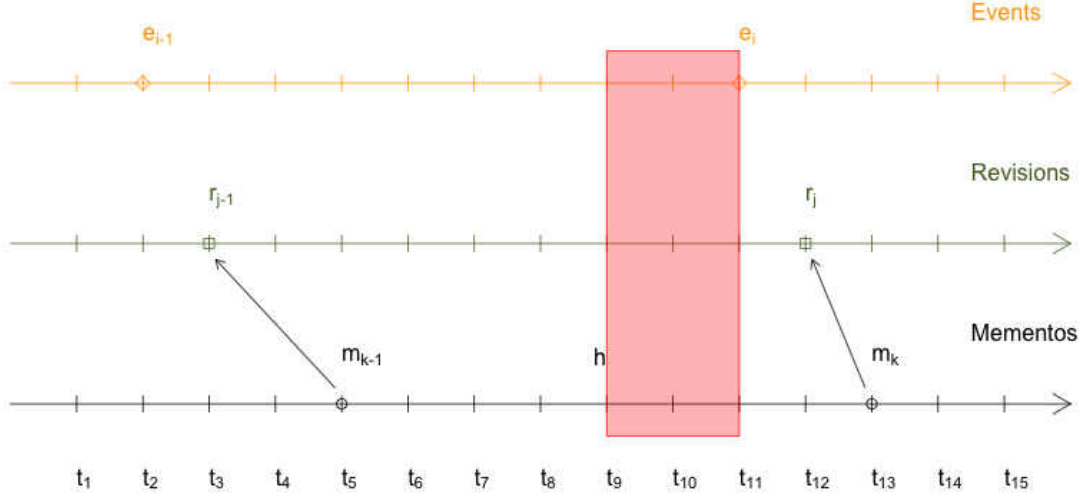


FIG. 49: Example of a *archive-extant spoiler area* (shown in light red) created by using the mindist heuristic, h is the midpoint between m_{k-1} and m_k

2. We have access to all revisions of a given resource
3. The memento-datetimes times for all revisions of a resource are defined and known
4. Event e must occur between the memento-datetimes of two consecutive mementos m_{k-1} and m_k (i.e., $t_{m_{k-1}} < t_e < t_{m_k}$)
5. Event e must occur prior to revision r_i corresponding to memento m_j (i.e., $r_j \equiv m_k \wedge t_e < t_{r_j}$)
6. The midpoint t_h , specified by (12), between m_{j-1} and m_j must occur prior to event e : (i.e., $t_{m_{k-1}} < t_h < t_e < t_{m_k}$)

Given consecutive mementos m_{k-1} and m_k , the midpoint t_h between them, and revision $r_j \equiv m_k$, this gives us the definition of a spoiler area defined by function \mathcal{S}_b over the interval beginning at start datetime t_s and ending at finish datetime t_f

produced by Equation (13).

$$[t_s, t_f] = \mathcal{S}_b(e) = \begin{cases} (t_h, t_e) & \text{if } t_h < t_e < t_{r_i} \wedge r_j \equiv m_k \wedge \\ & t_h = \frac{t_{m_{k-1}} + t_{m_k}}{2} \\ (0, 0) & \text{otherwise} \end{cases} \quad (13)$$

The return value of $(0, 0)$ exists for conditions where this relationship fails to hold. There are actually several conditions where there are no spoilers. These conditions are named based on the order of the halfway mark h , revision that has been archived r , and event e . For example, if a condition exists after archiving and the order of occurrences is halfway mark h followed by revision r followed by event e , then the condition is referred to as *Archive-Extant Safe HRE*.

Figure 50 shows the **Archive-Extant Safe HRE** condition where there is no spoiler area. Many of the spoiler area conditions hold. For instance, $m_k@t_{13} \equiv r_j@t_{10}$ and $t_h < t_{r_j}$. The exception is that $t_{r_j} < t_{e_i}$. If the mindist heuristic directs a user to m_k , we do not get a spoiler because the revision r_j occurred prior to the event e_i and the contents of that revision were not influenced by the information contained in event e_i .

Figure 51 showing the **Archive-Extant Safe RHE** condition also contains no spoiler area. The user attempting to choose a time t_a where $t_h < t_a < t_{e_i}$ will be directed to $m_k@t_{13}$, which is after $e_i@t_{11}$. What is different is that $m_k@t_{13} \equiv r_j@t_6$ and $r_j@t_6$ is prior to the midpoint $h@t_9$. Because $t_{r_j} < t_e$, we have no spoiler area.

Figure 52 showing the **Archive-Extant Safe EHR** condition contains no spoiler areas. If a user selects $t_a < t_{e_i}$, they will be directed to $m_{k-1}@t_5$. This is because event $e_i@t_7$ occurs prior to midpoint $h@t_9$. The first revision containing information about $e_i@t_7$ is $r_j@t_{10}$, which is after $h@t_9$.

Figure 53 shows the **Archive-Extant Safe ERH** condition where there is no spoiler area. Just like Archive-Extant Safe EHR, Archive-Extant Safe ERH has the conditions where $e_i@t_6$ occurs before $h@t_9$. Additionally, $r_j@t_8$ occurs before $h@t_9$. A user selecting $t_a < t_{e_i}$ will be directed to $m_{k-1}@t_5$, which was created from revision $r_{j-1}@t_3$, prior to event $e_i@t_6$.

Figure 54 shows the **Archive-Extant Safe REH** condition containing no spoiler areas. Just like Archive-Extant Safe ERH, Archive-Extant Safe REH has both $r_j@t_6$ and $e_i@t_7$ occurring prior to $h@t_9$. This leads any user selecting $t_a < t_{e_i}$ to memento $m_{k-1}@t_5$. In this case, even $m_k@t_{13}$ fails to contain spoilers, because $m_k@t_{13} \equiv r_j@t_6$ and $r_j@t_6$ takes place before event $e_i@t_7$.

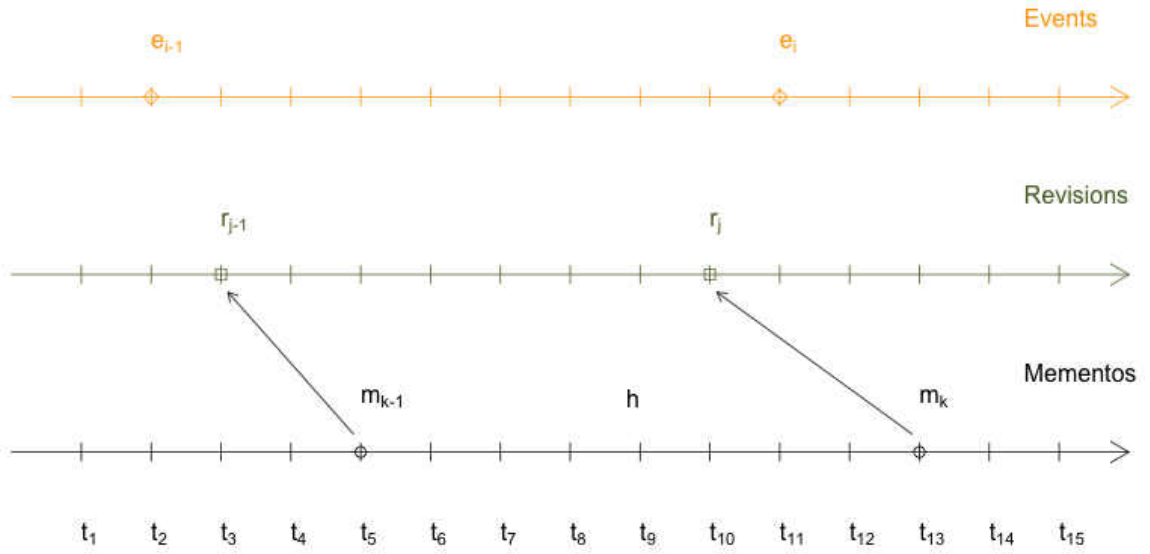


FIG. 50: Example of the condition: *Archive-Extant Safe HRE* for event e_i

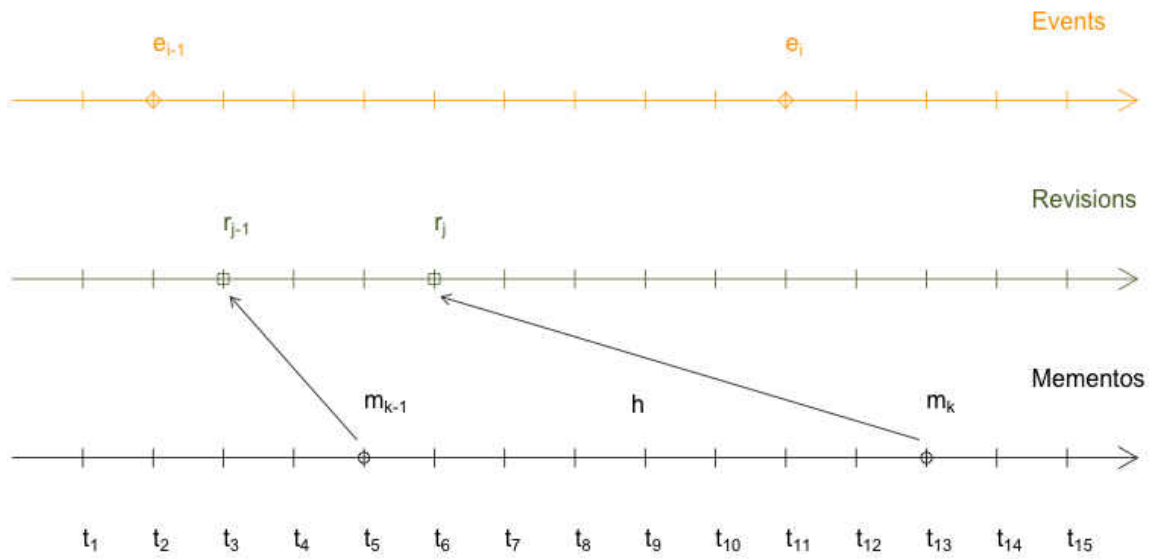


FIG. 51: Example of the condition: *Archive-Extant Safe RHE* for episode e_i

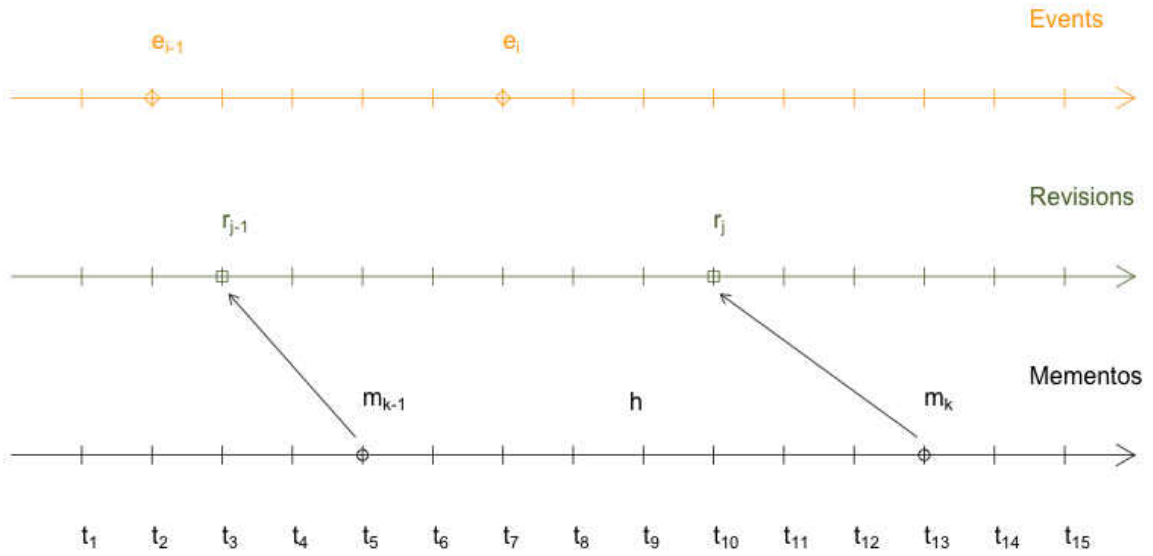


FIG. 52: Example of the condition: *Archive-Extant Safe EHR* for event e_i

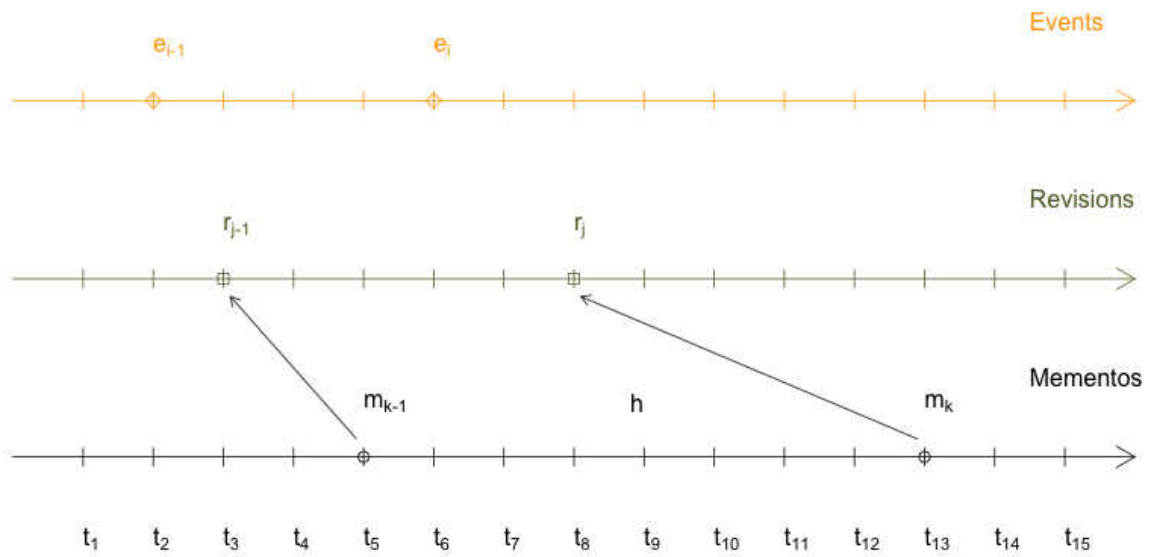


FIG. 53: Example of the condition: *Archive-Extant Safe ERH* for event e_i

Figure 55 shows the **Pre-Archive Safe** condition for event e_3 . There is a spoiler area for event $e_2@t_6$, but not for $e_3@t_{11}$. Why is this? After all $m_1@t_{13}$ is the first memento, and any datetime t_a prior to t_{m_1} will result in the user being directed to m_1 . The difference is that the revision $m_1@t_{13} \equiv r_j@t_{10}$ and $r_j@t_{10}$ occurs prior to $e_3@t_{11}$.

After the latest memento m_n , there are no spoiler areas. Two conditions exist in this case.

The first is the **Post-Archive Safe ER** condition. There are no spoiler areas because, even though the event e occurred prior to revision r , there is no memento corresponding to r yet, hence the user will be directed to a memento prior to r .

Figure 56 shows an example of this condition. Memento $m_n@t_5$ is the latest memento and event $e_i@t_6$ comes after it, followed by $r_j@t_8$. Note that $m_n \neq r_j$, because $m_n \equiv r_{j-1}$. There can be no spoiler area for e_i because any datetime chosen prior to t_6 will bring the user to memento $m_n@t_5$, which already exists prior to t_6 .

Alternatively, there exists the **Post-Archive Safe RE** condition. There are no spoiler areas in that situation because the event e occurred after revision r , meaning it would fail one of our existing tests for an archive-extant spoiler area.

Figure 57 shows this condition. Memento $m_n@t_5$ is the latest memento and revision $r_j@t_8$ comes after it. Event $e_i@t_{10}$ comes after $r_j@t_8$ as well. Just like before, $m_n \neq r_j$, but even if they were equivalent, there would still be no spoiler area because $t_{r_j} < t_{e_i}$.

Using these conditions we can find the datetimes where one might encounter a spoiler for a given event, but what about for an entire series of events?

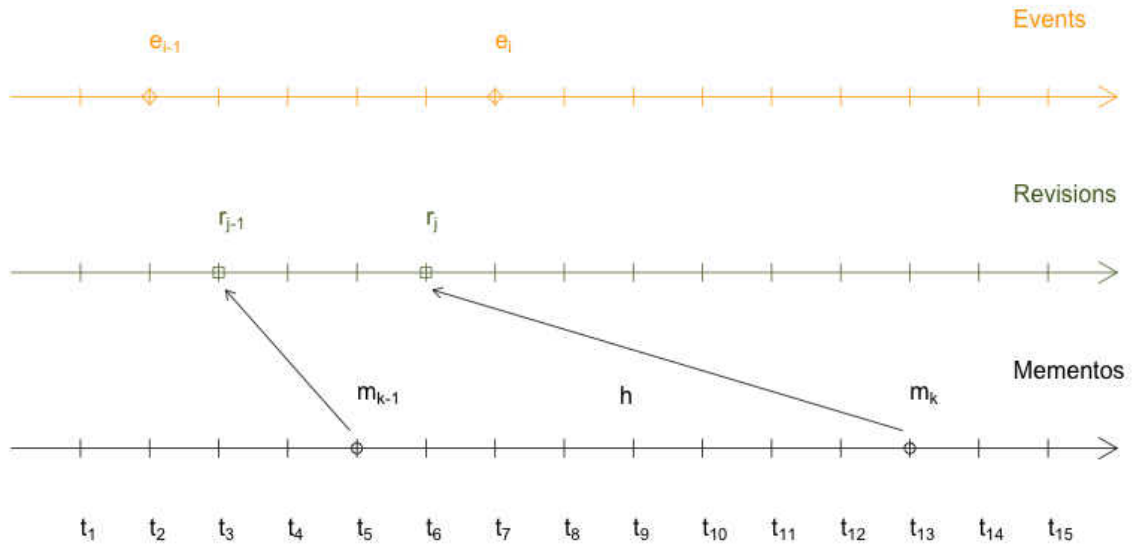


FIG. 54: Example of the condition: *Archive-Extant Safe REH* for event e_i

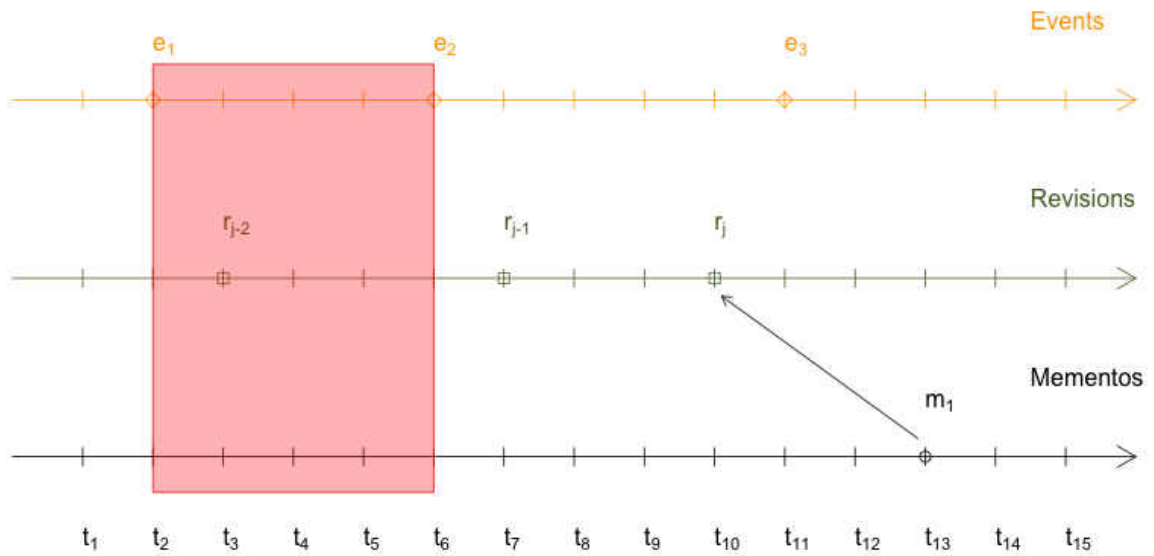


FIG. 55: Example of the condition: *Pre-Archive Safe* for event e_3 ; spoiler area exists for event e_2 , but not e_3

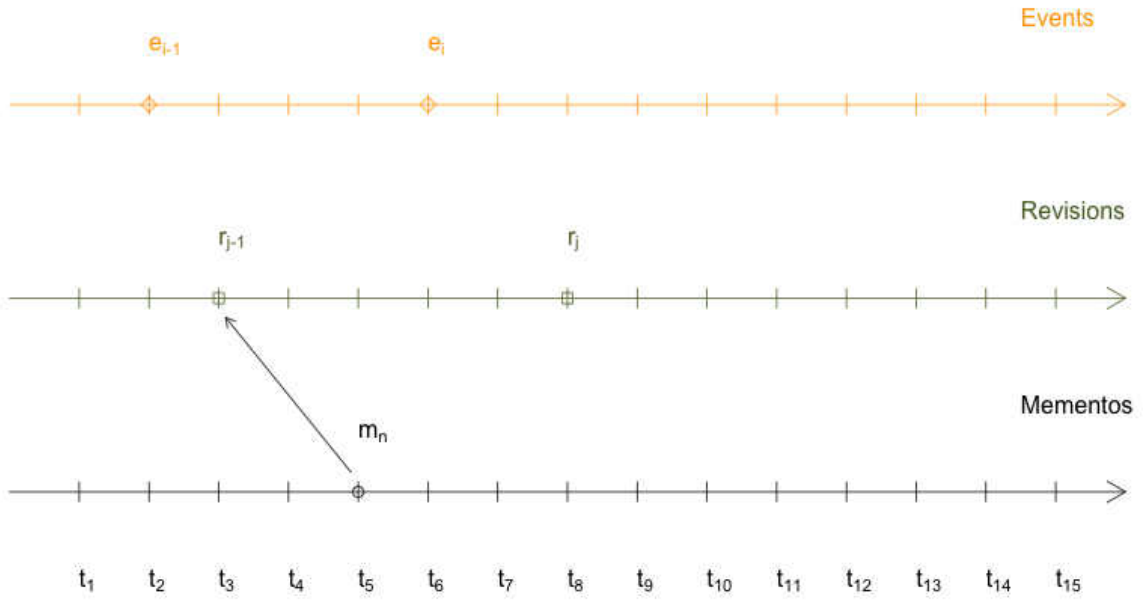


FIG. 56: Example of the condition: *Post-Archive Safe ER* for event e_i

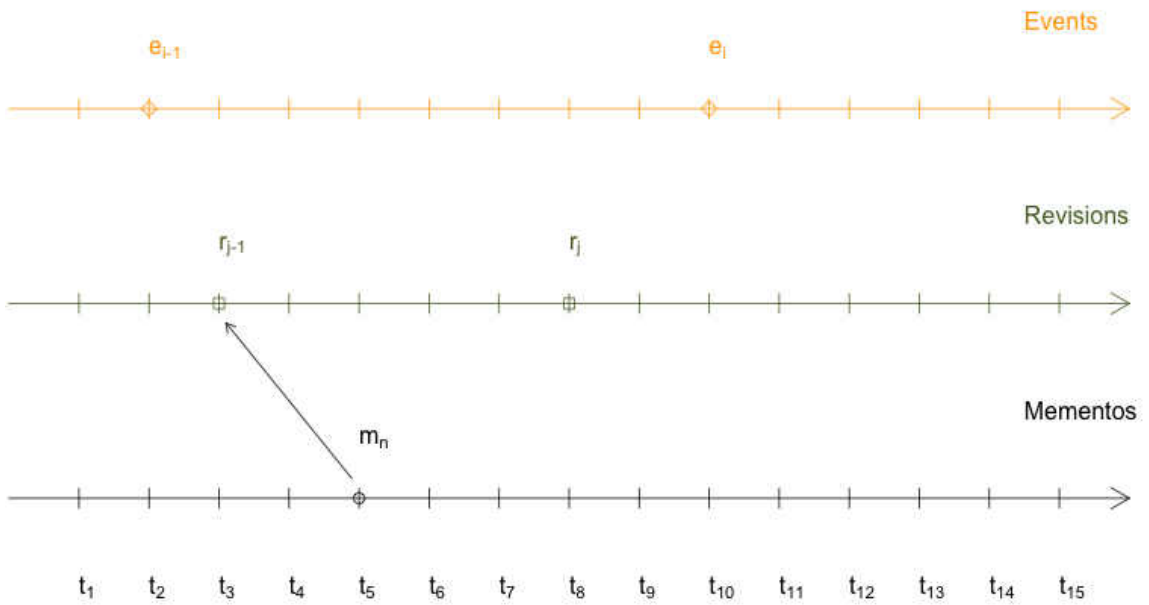


FIG. 57: Example of the condition: *Post-Archive Safe RE* for event e_i

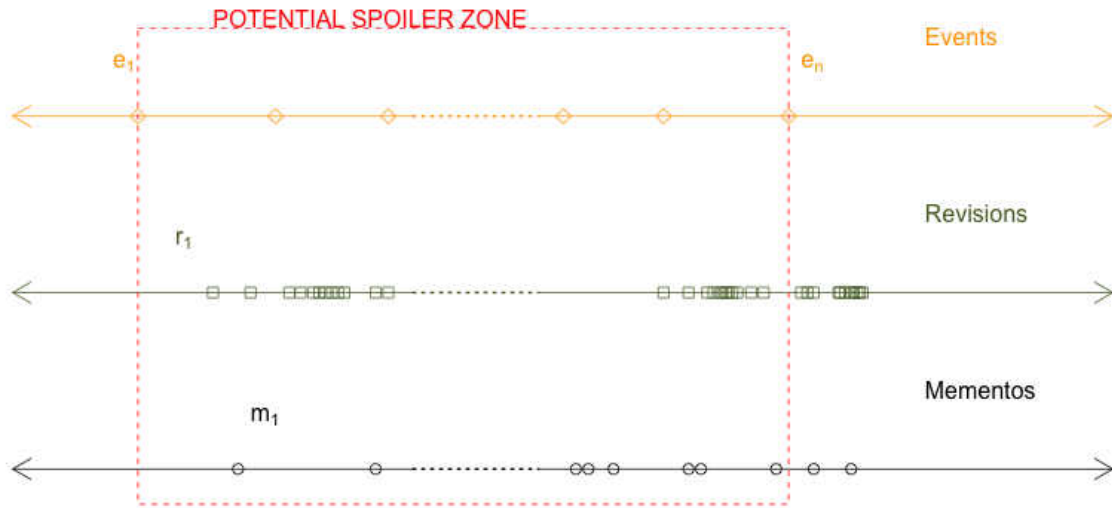


FIG. 58: Example of a potential spoiler zone, stretching from t_{e_1} to t_{e_n}

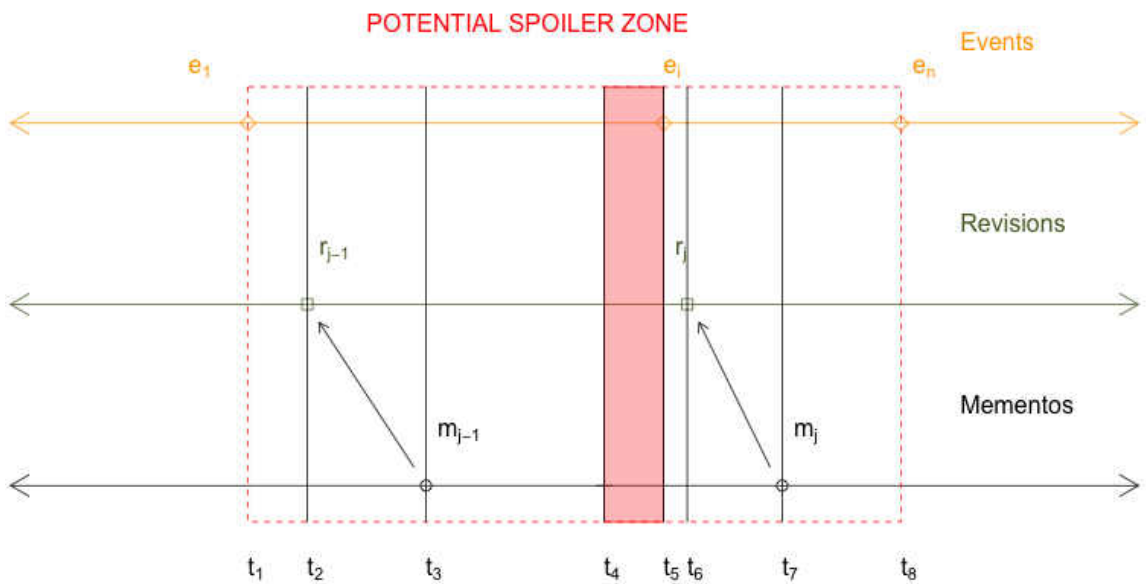


FIG. 59: Example of a spoiler area (light red area) for episode e_i inside potential spoiler zone (dotted red rectangle), stretching from t_{e_1} to t_{e_n}

5.2 CONSIDERATIONS FOR MULTIPLE EVENTS AND AGGREGATING SPOILER AREAS

The spoilers we have been discussing are not for single events, but episodes of a larger story. **Our assumption is that *single* episode stories are revealed in an instant, on a specific date, and anyone trying to avoid spoilers for such a story can not be served by our method.**

So, how does one handle multiple episodes? What does that mean for our spoiler areas? For a given resource, using mindist, what is the chance of attempting web time travel with Memento and getting a spoiler?

First we define a **potential spoiler zone** across the length of the series we are looking at. The start datetime of the potential spoiler zone is t_{e_1} , the datetime of the first episode. The end datetime of our potential spoiler zone is t_{e_n} , the datetime of the last (or latest) episode. We assume that a user searching for datetimes prior to the first event e_1 should get no spoilers, so that is the lower bound. We also assume that no additional spoilers can be revealed after the last event e_n . This provides a single area in which we can determine the probability of getting a spoiler for a single episode in the series. Figure 58 shows an example of such a zone.

Figure 59 shows a spoiler area inside a potential spoiler zone. Consider randomly choosing a desired datetime within this zone. What is the probability of landing inside the spoiler area for given episode e_i ?

Probability is defined as the number of times something can occur divided by the total number of outcomes [113]. The smallest unit of datetime on the web is the second. We cannot gain more precision over time due to the fact that HTTP headers (and hence Memento-Datetimes) use the second as the smallest unit. Therefore, if we let s be the number of seconds between e_1 and e_n in which one can encounter spoilers, and we let c be the number of seconds between e_1 and e_n , then the probability of encountering a spoiler is shown by equation (14).

$$Pr(\text{spoiler}) = \frac{s}{c} \tag{14}$$

Algorithm 6 shows how one would calculate the probability of encountering a spoiler for a given resource.

Once we have determined the probability of encountering a spoiler for a resource

```

SPOILER-PROBABILITY( $E, R, M$ )
1   $A = \text{FIND-SPOILER-AREAS}(E, R, M)$ 
2   $c = 0$ 
3   $s = 0$ 
4  for  $v = t_{e_1}$  to  $t_{e_n}$ 
5       $c = c + 1$ 
6      if  $\text{IN-SPOILER-AREA}(A, v)$ 
7           $s = s + 1$ 
8   $p = \frac{s}{c}$ 
9  return  $p$ 

```

Algorithm 6: Algorithm for finding the probability of spoilers between e_1 and e_n for a given resource, E is the list of datetimes for events, R is the list of datetimes for revisions, and M is the list of datetimes for mementos

within the Internet Archive, we can then use that probability to compare that resource to others. In this way we can determine how safe a given URI is for users who want to avoid spoilers using the Wayback Machine or a Memento TimeGate that uses the mindist heuristic.

5.3 SUMMARY

In this chapter, we explored the existence of **spoiler areas**, which are defined as the sets of datetimes where a user will be directed to a memento containing spoilers even though they selected a desired datetime prior to the event they were trying to avoid. These spoiler areas are defined based on the positions of revisions, events, and mementos in the timeline for a given resource. Table 8 shows a summary of all conditions involving event e , revision r , memento $m \equiv r$ and midpoint h . In these cases m_1 and m_n are the first and last mementos, respectively.

Also, we determined how to calculate the probability of encountering a spoiler for a given resource. This function can be used to examine a resource in the Internet Archive and determine how safe that resource is to those attempting to avoid spoilers. We will do this in the next chapter.

TABLE 8: Conditions for relationships between episodes denoted by e , revisions denoted by r , mementos denoted by m , and a midpoint between mementos denoted by h . Mementos m_1 and m_n denote first and last mementos, respectively.

Order of Occurrence	Condition Name	Disposition	Description
e, r, m_1	Pre-archive Spoiler Area	Spoiler for e in area between e_1 and e	$\forall t_a : t_{e_1} < t_a < t_e \wedge m_1 \equiv r$ $\mathcal{G}^{mindist}(R, t_a) = m_1$ spoiler because $t_r > t_e$
r, e, m_1	Pre-Archive Safe	Safe for e	$\forall t_a : t_{e_1} < t_a < t_e \wedge m_1 \equiv r$ $\mathcal{G}^{mindist}(R, t_a) = m_1$ not spoiler because $t_r < t_e$
m_{k-1}, h, e, r, m_k	Archive-Extant Spoiler Area	Spoiler for e in area between h and e	$\forall t_a : t_a < t_e \wedge m_k \equiv r$ $\mathcal{G}^{mindist}(R, t_a) = m_k$ spoiler because $t_r > t_e$
m_{k-1}, h, r, e, m_k	Archive-Extant Safe HRE	Safe for e	$\forall t_a : t_a < t_e \wedge m_k \equiv r$ $\mathcal{G}^{mindist}(R, t_a) = m_k$ not spoiler because $t_r < t_e$
m_{k-1}, r, h, e, m_k	Archive-Extant Safe RHE	Safe for e	$\forall t_a : t_h < t_a < t_e \wedge m_k \equiv r$ $\mathcal{G}^{mindist}(R, t_a) = m_k$ not spoiler because $t_r < t_e$
m_{k-1}, r, e, h, m_k	Archive-Extant Safe REH	Safe for e	$\forall t_a : t_a < t_e \wedge m_k \equiv r$ $\mathcal{G}^{mindist}(R, t_a) = m_{k-1}$ not spoiler because $t_r < t_e$
m_{k-1}, e, h, r, m_k	Archive-Extant Safe EHR	Safe for e	$\forall t_a : t_a < t_e \wedge m_k \equiv r$ $\mathcal{G}^{mindist}(R, t_a) = m_{k-1}$ not spoiler because $t_e < t_h$
m_{k-1}, e, r, h, m_k	Archive-Extant Safe ERH	Safe for e	$\forall t_a : t_a < t_e \wedge m_k \equiv r$ $\mathcal{G}^{mindist}(R, t_a) = m_{k-1}$ not spoiler because $t_e < t_h$
m_n, e, r	Post-Archive Safe ER	Safe for e	$\mathcal{G}^{mindist}(R, t_a) = m_n$ even though $t_r > t_e$, not spoiler because $r \neq m_n$
m_n, r, e	Post-Archive Safe RE	Safe for e	$\mathcal{G}^{mindist}(R, t_a) = m_n$ not spoiler because $t_r < t_e$

CHAPTER 6

MEASURING SPOILER PROBABILITY IN POPULAR WIKIS

In this chapter we use data from actual wiki pages and the Internet Archive to show that spoiler areas do exist for wiki fan sites. Using the sources in Table 10 we can fill in the values needed for Equations (11) and (13).

6.1 STRUCTURE OF THE EXPERIMENT

We selected 16 fan wikis based on television shows for our experiment. Table 9 shows some of the details for each fan wiki. Each television show selected has had at least two seasons and a currently active wiki. *House of Cards* was chosen because an entire season is released on Netflix in a single day, making it different from networked television shows. *Lost* was chosen because its wiki, *Lostpedia*, has actually undergone some academic study [69], and is the oldest and largest fan wiki under consideration.

The articles of each wiki were analyzed and processed to find spoiler areas using a process simplified in Algorithm 7. The XML dumps for each wiki were acquired by automating the submission to each wiki’s export page, an example of which is shown in Figure 60. Utilizing this method, we computed additional statistics based on the revisions, mementos, the memento-revision mapping, and the spoiler areas.

Out of the 40,868 wiki pages processed for this experiment, we discovered that many of them were wiki redirects, which are a way to “forward users from one page name to another” [35]. Redirects are often used to deal with articles that can be referred to by multiple names. Sometimes wiki editors may not know the real name of an introduced fictional character until much later, and will use a redirect from the old name to the new. Sometimes wiki editors will create pages not knowing that one already exists, leaving future editors to create a redirect now that they know that a new page title was desired. Because of the number of redirects that contained only a single revision and only a single memento, we removed the redirects from consideration for calculation of spoiler areas and other statistics. This removed 16,394 pages from consideration, leaving us with 24,474 pages to process.

TABLE 9: Fan wikis used in the spoiler areas experiment

Television Show (Network)	Wiki URI .wikia.com	# of Pages	t_{r_1}	t_{e_1}	% of pages in Internet Archive
the Big Bang Theory (CBS)	bigbangtheory	1120	2007-12-14	2007-09-24	68.8%
Boardwalk Empire (HBO)	boardwalkempire	2091	2010-03-18	2010-08-23	80.6%
Breaking Bad (A&E)	breakingbad	998	2009-04-27	2008-01-20	76.0%
Continuum (Showcase)	continuum	258	2012-11-13	2012-05-27	86.8%
Downton Abbey (BBC)	downtonabbey	784	2010-10-04	2010-09-26	53.1%
Game of Thrones (HBO)	gameofthrones	3144	2010-06-24	2011-04-17	75.8%
Grimm (NBC)	grimm	1581	2010-04-14	2011-10-28	57.5%
House of Cards (Netflix)	house-of-cards	251	2013-01-11	2013-02-01	97.2%
How I Met Your Mother (CBS)	how-i-met-your-mother	1709	2008-07-21	2005-09-19	58.7%
Lost (ABC)	lostpedia	18790	2005-09-22	2004-09-22	39.1%
Mad Men (AMC)	madmen	652	2009-07-25	2007-06-03	85.0%
NCIS (CBS)	ncis	5345	2006-09-25	2003-09-23	93.2%
Once Upon A Time (ABC)	onceuponatime	1470	2011-08-09	2011-10-23	79.9%
Scandal (ABC)	scandal	331	2011-06-07	2012-04-05	82.8%
True Blood (HBO)	trueblood	1838	2008-10-06	2008-09-07	74.1%
White Collar (USA)	whitecollar	506	2009-10-30	2009-10-23	79.1%

The wiki XML exports were downloaded at a different time than the TimeMaps for those wiki pages. To overcome this inconsistency, any mementos in TimeMaps that existed after the wiki page was downloaded were discarded. In the next section, we discuss the results of this data gathering and analysis.

6.2 RESULTS

Of the 24,474 pages processed, only 15,119 pages actually had TimeMaps at the Internet Archive at the time the wiki exports were extracted. This means that **roughly 38% of the pages under consideration were not available in the Internet Archive**. This presents a problem for episodic fiction fans trying to use the Wayback Machine, or the Internet Archive through Memento, to avoid spoilers. This further demonstrates that using Memento directly on wikis is better for avoiding spoilers. The results have been broken up into three sections: data timelines, spoiler areas and probabilities, and missed updates and redundant mementos.

¹<http://lostpedia.wikia.com/wiki/Special:Export>

TABLE 10: Information required to determine if spoilers can be encountered if mindist is used

Required Information	Source of Information	Part of $\mathcal{S}(e)$ equation met
Memento datetimes of mementos for article	Memento TimeMap	$r_j \stackrel{?}{=} m_k$ $t_h = \frac{t_{m_{k-1}} + t_{m_k}}{2}$ t_{m_k}
Revision datetimes for the wiki article	XML Dump of Wiki Article	$r_j \stackrel{?}{=} m_k$ t_{r_j}
Episode datetimes	List of episodes from http://epguides.com	t_e

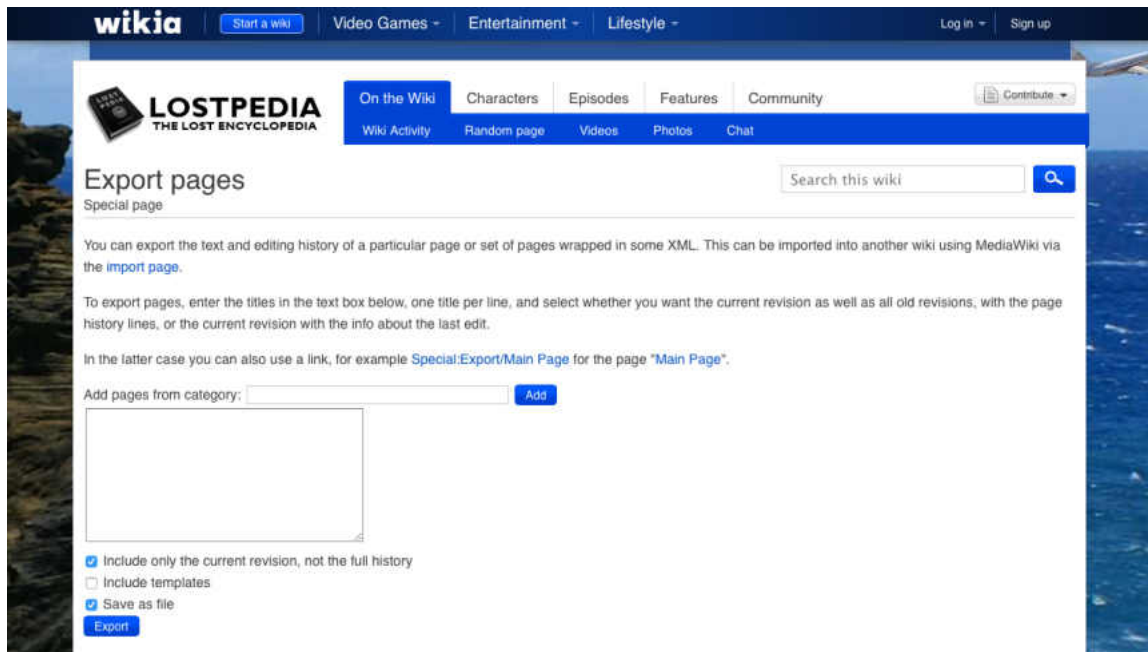


FIG. 60: Example export page for a MediaWiki installation¹

```

FINDSPOILERAREASINWIKIS(episodeList, wikiURI)
1  episodeTimes = GETEPISODETIMES(EPISEODELIST)
2  wikiTitles = GETPAGETITLES(wikiURI)
3  for each title  $\in$  wikiTitles
4      wikidump = FETCHXMLDUMP(title, wikiURI)
5      revisions = EXTRACTREVISIONTIMES(wikidump)
6      timemapURI = MAKETMURI(wikiURI, title)
7      timemap = FETCHTIMEMAP(timemapURI)
8      mementos = EXTRACTMEMENTOTIMES(timemap)
9      mementoRevisionMap =
          MAPREVSTOMEMS(revisions, mementos)
10     for each episode  $\in$  episodeTimes
11         paSpoilerArea =
             $S_a$ (episode, mementoRevisionMap)
12         aeSpoilerArea =
             $S_b$ (episode, mementoRevisionMap)
13         spoilerAreaList.append(paSpoilerArea)
14         spoilerAreaList.append(aeSpoilerArea)
15     MAPPAGETOSPOILERS(
        wikipageSpoilerMap, title, spoilerAreaList)
16 return wikipageSpoilerMap

```

Algorithm 7: Algorithm for spoiler probability experiment



(a) Big Bang Theory



(b) Boardwalk Empire



(c) Breaking Bad



(d) Continuum



(e) Downton Abbey



(f) Game of Thrones



(g) Grimm



(h) House of Cards

FIG. 61: Timelines for the wiki sites used in this experiment: top timeline represents the length of the episode run, middle timeline represents the life of the wiki, bottom timeline represents the span of time the Internet Archive has operated on the site



(i) How I Met Your Mother



(j) Lostpedia



(k) Mad Men



(l) NCIS



(m) Once Upon A Time



(n) Scandal



(o) True Blood



(p) White Collar

6.2.1 DATA TIMELINES

We were able to process the data and determine the timelines for episodes, wiki revisions, and mementos from the Internet Archive. Figure 61 shows the timelines for each wiki, showing graphically how they evolved as the time progressed.

As we see, the 16 wikis have somewhat different histories. The top line, corresponding to our *events* line from Chapter 5, shows the timeline of episodes from the premier of the television show to the latest episode. The middle line shows the life of the wiki up to the present, starting at the first revision of the first page. The bottom line shows the life of the Internet Archive’s interest in the wiki, starting with the first memento captured.

Even though the top line, representing episodes, stops while the wiki goes on, in most cases only the season has ended for the show, not the show itself. The only three shows in our data set that have truly gone off of the air by the time of this study are *Lost*, *How I Met Your Mother*, and *True Blood*, so it is interesting to see that these wikis continued to receive updates long after their shows were gone.

Most of the wikis follow the pattern discussed in section 2.6, where first the television show started, then the wiki came into existence, and the Internet Archive began to archive it. Others display a somewhat surprising characteristic, where the wiki actually exists prior to the television show!

Some networks create enough publicity that fans will create a wiki containing what little information has been released prior to the airing of the television show. HBO spent a lot of time and money advertising *Boardwalk Empire* (Figure 61b) in advance the series [20]. *Game of Thrones* (Figure 61f) already had a literary fan following who immediately sought out information on the television show. The writers of *Grimm* (Figure 61g) and *Scandal* (Figure 61n) already had fans [36, 94], so the wiki was created prior to the show.

If we were to add the press releases and news reports to the top line, we would see the line extend and reveal the expected pattern of event follows wiki revision follows archive. Of course, we encountered no instances where the archive starts prior to the site’s existence.

These timelines are useful because they demonstrate differences between these sites and show that not all fan wikis have the same expected history.

6.2.2 SPOILER AREAS AND PROBABILITIES

We determined that there are three categories of page behavior with regards to timelines:

1. **normal** - the television show started, then the wiki page was created, then the web archive recorded the page
2. **wiki-before-show** - the wiki page was created with some type of foreknowledge about the show before the show began to air; not spoilers per se, but based off of press releases, interviews, etc.
3. **season-in-a-day** - an entire season of a show is released in one day, effectively leading to n episodes all airing at the same time; the *House of Cards* show from Netflix fits into this category

Figure 62 shows our spoiler area graph for the most popular page using the normal behavior. In this case, a page on *Lostpedia* about a character named Kate Austen was created after the show had aired. Each spoiler area is shown in red using an alpha channel that gives it some degree of transparency. When these transparent red areas stack up, of course the red gets darker, so we cannot reliably see all of the 86 pre-archive spoiler areas that exist prior to the first memento. Because this page only has 4 mementos around 2009 and then no archiving by the Internet Archive until 2011, there are 8 archive-extant spoiler areas, also shown in red. The probability of encountering a spoiler for Kate's page is 67%, calculated by Equation (14) from section 5.2.

Figure 63 shows the spoiler areas for the most popular page of the *Game of Thrones Wiki*, for the character Daenerys Targaryen. We see no pre-archive spoiler areas because the page is an example of the wiki-before-show behavior. Even though 29 mementos exist for this page, there are not enough to avoid the existence of the 24 archive-extant spoiler areas. *Game of Thrones* is a very popular series, and it is likely that a popular page linked to this page, and the Internet Archive crawler was led to this page earlier than many other pages we have seen. The probability of encountering a spoiler in Daenerys Targaryen's page is only 16%, even though there are 24 small spoiler areas, likely due to this aggressive archiving.

Frank Underwood's wiki page for *House of Cards*, seen in Figure 64 is an example of the season-in-a-day behavior. This series has two seasons, but all 13 episodes for

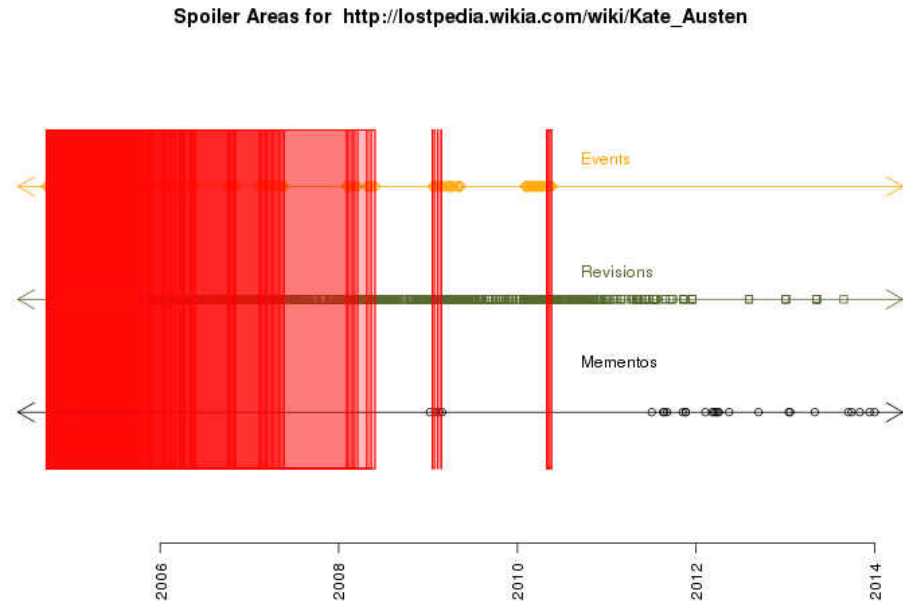


FIG. 62: Spoiler areas for the most popular page in *Lostpedia* (3,531 revisions)²

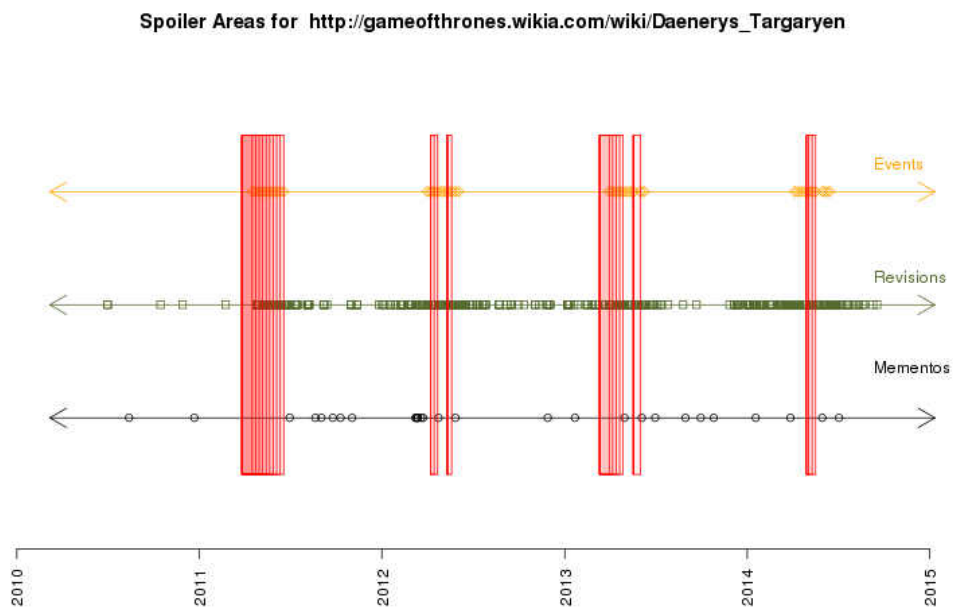


FIG. 63: Spoiler areas for the most popular page in the *Game of Thrones Wiki* (768 revisions)³

²http://lostpedia.wikia.com/wiki/Kate_Austen

³http://gameofthrones.wikia.com/wiki/Daenerys_Targaryen

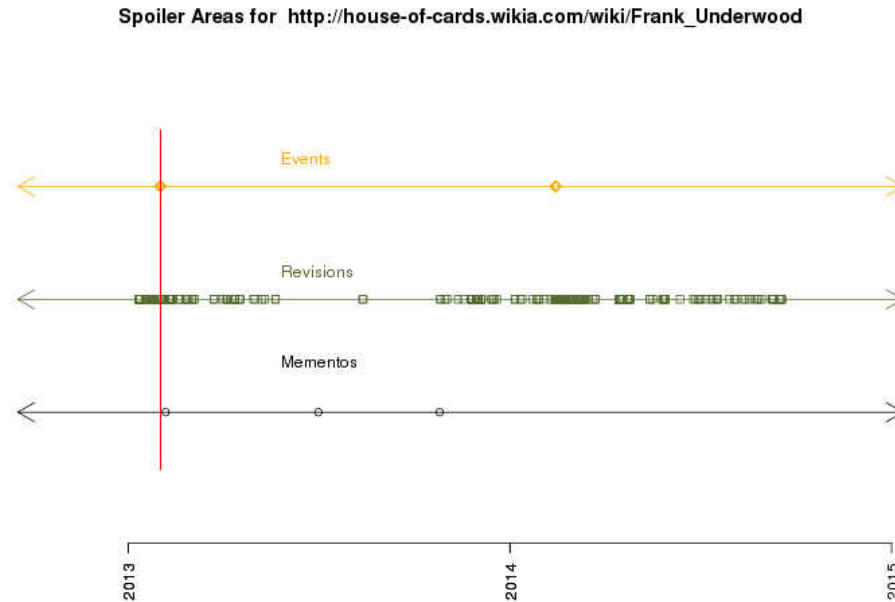


FIG. 64: Spoiler areas for the most popular page in the *House of Cards Wiki* (380 revisions)⁴

a single season are released on the same day, resulting in 13 stacked event points for each season. This series was added to this experiment because of the limited number of events and the unique release behavior for this show. This resulted in 13 pre-archive spoiler areas for this page, all at the beginning of the series. These pre-archive spoiler areas have no size due to the fact that all of them begin and end at the same time. This leads to a 0% chance of encountering a spoiler in this wiki page, seeing as each season is released like a 13-hour movie rather than on a weekly basis. In this case, time is not able to differentiate between individual episodes because $t_{e_1} = t_{e_2} = \dots t_{e_{13}}$. We require a new dimension to order otherwise simultaneous events. A different situation exists with another Netflix series, *Arrested Development* Season 4, in which all episodes for a season are released at once, but the episodes do not need to be viewed in any particular order, making it difficult to identify when spoilers would occur.

Table 11 contains statistics for the most popular page in each of the wikis surveyed, where popularity is determined by the number of page revisions generated.

⁴http://house-of-cards.wikia.com/wiki/Frank_Underwood

TABLE 11: Spoiler probabilities for most popular pages within each fan wiki

Wiki	Page Name	Probability of Spoiler	# of Spoiler Areas	# of Revisions	# of Mementos
bigbangtheory	Sheldon Cooper	0.31	69	1958	30
boardwalkempire	Nucky Thompson	0.15	31	290	15
breakingbad	Walter White	0.43	40	882	20
continuum	Keira Cameron	0.54	21	104	5
downtonabbey	Sybil Branson	0.42	23	580	3
gameofthrones	Daenerys Targaryen	0.16	24	768	29
grimm	Nick Burkhardt	0.39	30	795	5
house-of-cards	Frank Underwood	0.0	13	380	3
how-i-met-your-mother	Barney Stinson	0.55	120	588	13
lostpedia	Kate Austen	0.67	94	3531	27
madmen	Mad Men Wiki	0.22	36	250	85
ncis	Abigail Sciuto	0.67	182	404	11
onceuponatime	Emma Swan	0.36	34	1210	11
scandal	Main Page	0.60	31	250	14
trueblood	Eric Northman	0.28	47	931	14
whitecollar	Neal Caffrey	0.29	38	199	8

Seeing as these wikis are authored by fans, readers familiar with many of these television shows will not be surprised that most of the popular pages are main characters. The table also lists the number of spoiler areas, revisions, and mementos, showing how there is not a simple relationship between these values that indicate the probability of encountering a spoiler. Appendix A contains more visualizations of spoiler areas from the most popular pages in each wiki.

From this we see the spoiler areas and hence spoiler probabilities for individual pages, but what about entire sites? Again, let us consider our behavior categories, this time applying them to an entire wiki. We have created a histogram showing the spoiler probability for each wiki.

Figure 65 shows the histogram for the *Lostpedia* wiki with the normal episodes-revisions-mementos behavior. We see several peaks in the histogram around 0.65, 0.82, and 0.99, but nothing resembling a standard normal distribution. Note that no page has a spoiler probability of less than 0.26, because the wiki was created a year after the show first started airing, giving each page a one season long pre-archive spoiler area to begin with.

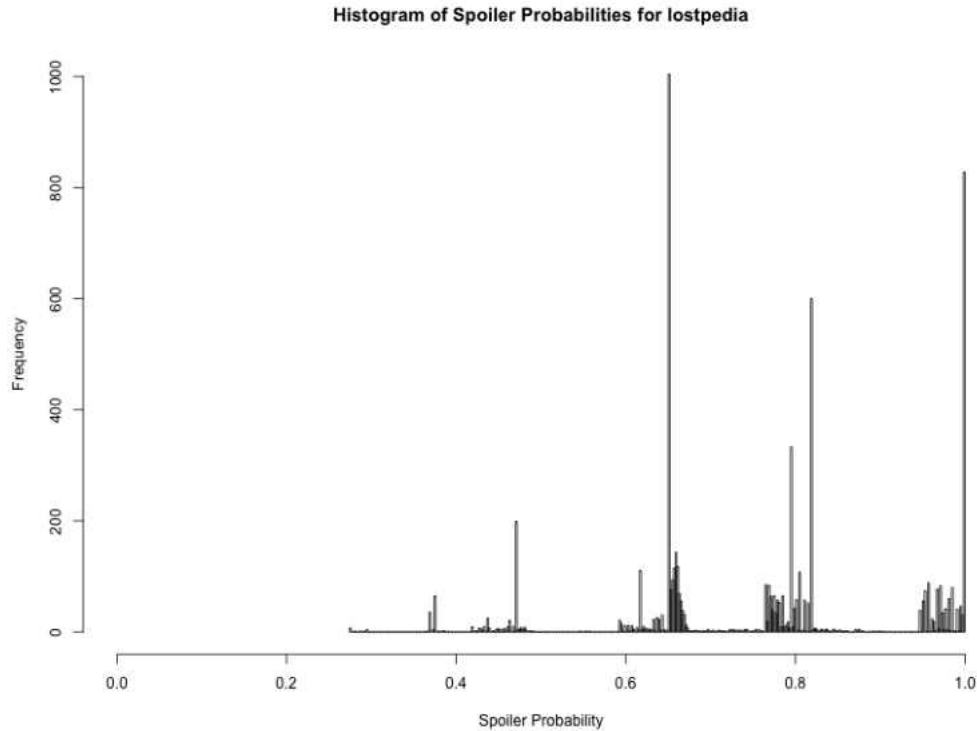


FIG. 65: Histogram of spoiler probabilities for *Lostpedia*

Figure 66 shows the histogram for the *Game of Thrones Wiki* observing a wiki-before-show behavior. We see large peaks around 0.35, 0.64, and 0.99. Because there are no pre-archive spoiler areas for any page, only archive-extant spoiler areas remain, giving us a much broader distribution of probabilities than we saw for *Lostpedia*.

Our final category of season-in-a-day is observed by the *House of Cards Wiki*, shown in Figure 67. There we see most pages have no spoiler probability at all and the maximum spoiler probability is around 0.27. Again, this is where our model breaks down, leading to a number of zero-length pre-archive spoiler areas because of the release process for the television show.

Figure 68 shows the histogram of spoiler probabilities for all pages in this study. In the histogram, we see the highest peak around 0.66, which corresponds to the mean shown in Table 12. Also in Table 12, we see the number of revisions per day is an order of magnitude lower than the number of mementos recorded per day. This is concerning for those trying to avoid spoiler using the Wayback Machine, even if it used the minpast heuristic, because there are many missed updates to these pages

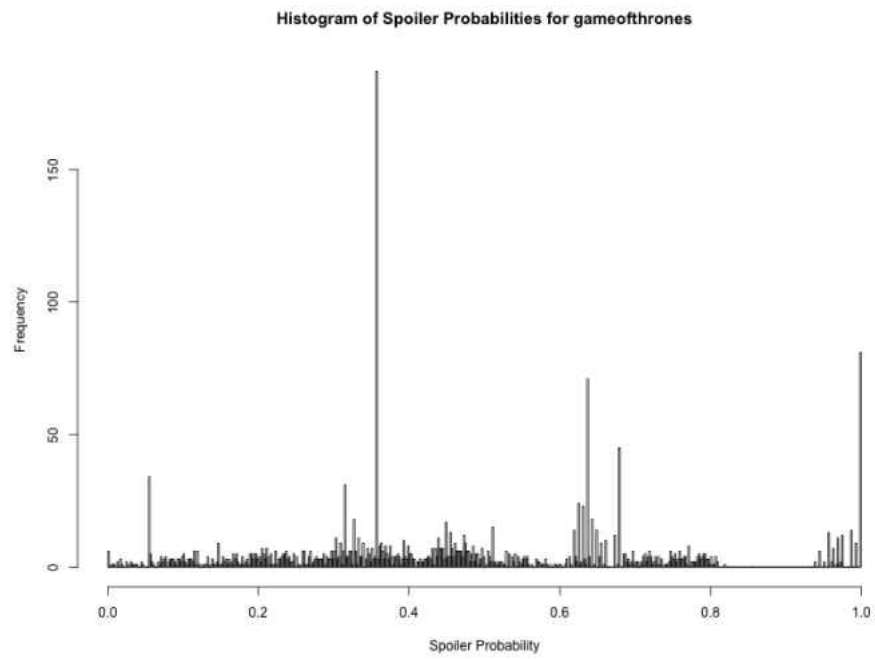


FIG. 66: Histogram of spoiler probabilities for *Game of Thrones Wiki*

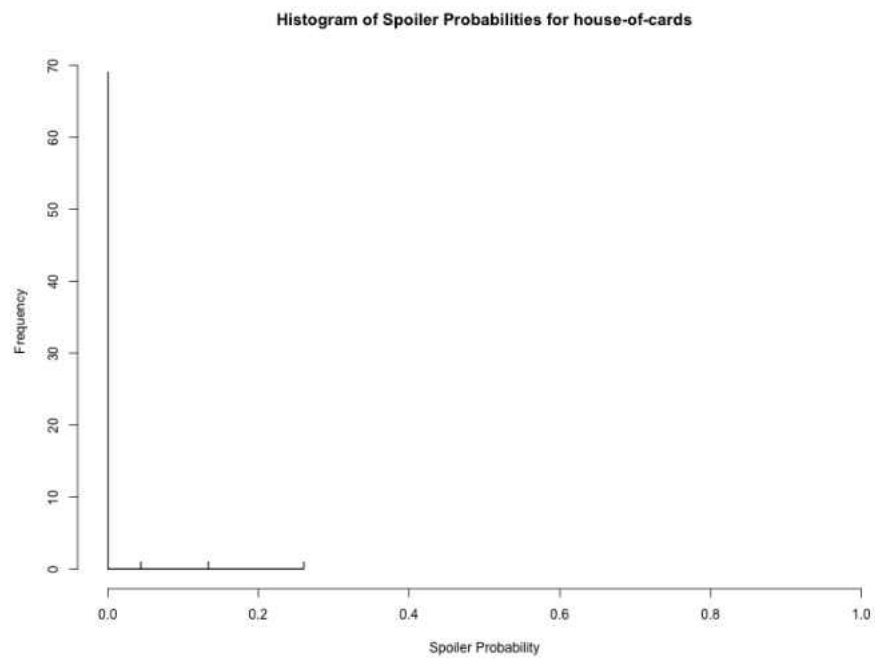


FIG. 67: Histogram of spoiler probabilities for *House of Cards Wiki*

TABLE 12: Statistics for each fan wiki

Wiki	Probability of Spoiler			Revisions/Day			Mementos/Day		
	Mean	std dev	Rel Err	Mean	std dev	Rel Err	Mean	std dev	Rel Err
bigbangtheory	0.667	0.160	0.0116	0.0506	0.0668	0.0639	0.0033	0.0034	0.0488
boardwalkempire	0.417	0.170	0.0160	0.0102	0.0185	0.0718	0.0022	0.0026	0.0452
breakingbad	0.746	0.205	0.0127	0.0185	0.0351	0.0872	0.0032	0.0032	0.0459
continuum	0.394	0.177	0.0471	0.0317	0.0250	0.0829	0.0051	0.0023	0.0479
downtonabbey	0.585	0.174	0.0196	0.0374	0.0636	0.1124	0.0020	0.0013	0.0419
gameofthrones	0.473	0.248	0.0122	0.0425	0.0652	0.0356	0.0041	0.0049	0.0279
grimm	0.479	0.175	0.0201	0.0700	0.0857	0.0672	0.0027	0.0015	0.0305
house-of-cards	0.006	0.035	0.6705	0.0772	0.1364	0.2082	0.0075	0.0044	0.0687
how-i-met-your-mother	0.741	0.100	0.0046	0.0163	0.0220	0.0463	0.0014	0.0010	0.0263
lostpedia	0.768	0.163	0.0027	0.0391	0.1083	0.0348	0.0040	0.0055	0.0173
madmen	0.530	0.144	0.0133	0.0049	0.0076	0.0764	0.0014	0.0021	0.0755
ncis	0.818	0.107	0.0041	0.0073	0.0097	0.0413	0.0009	0.0008	0.0279
onceuponatime	0.516	0.163	0.0132	0.1271	0.1327	0.0437	0.0037	0.0025	0.0281
scandal	0.591	0.165	0.0269	0.0418	0.0484	0.1120	0.0030	0.0019	0.0608
trueblood	0.517	0.162	0.0106	0.0210	0.0410	0.0658	0.0016	0.0016	0.0345
whitecollar	0.390	0.250	0.0500	0.0117	0.0147	0.0986	0.0019	0.0015	0.0609
Overall	0.659	0.226	0.0029	0.0362	0.0871	0.0200	0.0032	0.0044	0.0114

each day.

Figure 69 shows the cumulative distribution function for spoiler probabilities for all pages surveyed. From here we see the spoiler probability for pages rise at low percentages of the whole, indicating that most pages in the study have probability of encountering a spoiler.

Histograms for each wiki are available in Appendix B and cumulative distribution function graphs for each wiki are available in Appendix C.

6.2.3 MISSED UPDATES AND REDUNDANT MEMENTOS

Possessing every last revision of a page put us in the unique position of being able to study not only how often each page changed, but how often they get archived. We took the Memento-Datetimes of each wiki revision and compared them to the Memento-Datetimes of each archived memento in the Internet Archive. From here we were able to determine how well each page was being archived at a given time

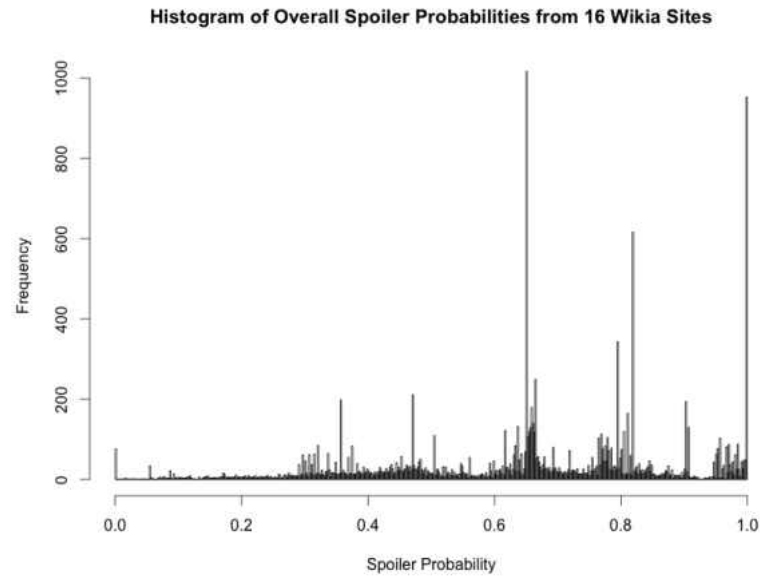


FIG. 68: Histogram of spoiler probabilities for all pages in study

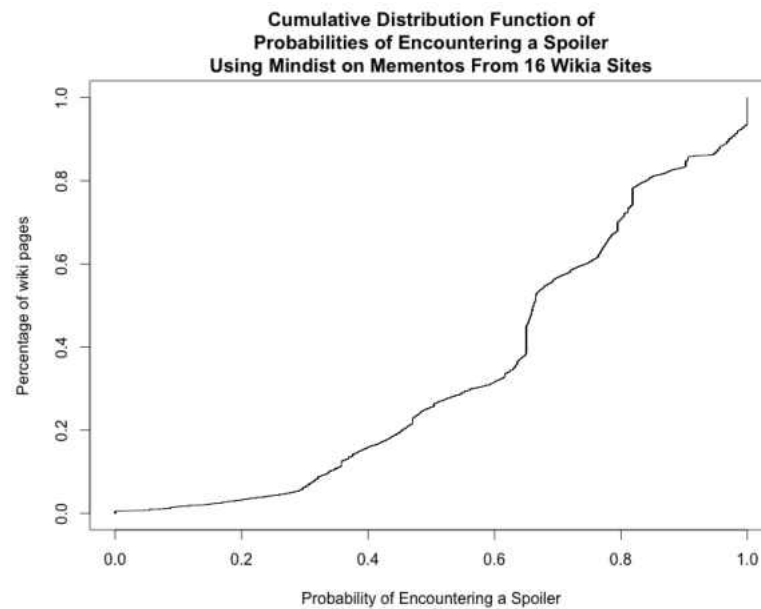


FIG. 69: Graph of the cumulative distribution function of spoiler probabilities for all 16 wiki sites

period.

In figure 70 we see the number of missed updates encountered over the history of all pages in the study. The Y-axis represents each URI in the dataset. The X-axis is time. Lighter colors indicate fewer missed updates on that day. Of interest are the vertical lines seen throughout the visualization. The datetimes for these lines correspond to changes in policy at the Internet Archive. In 2009 and in late 2011, the Internet Archive reduced its quarantine period for archiving of new pages. In October of 2013, the Internet Archive published the *Save Page Now* feature, leading to fewer missed updates after that point. This graph shows that the more aggressive archiving performed by the Internet Archive in more recent years are resulting in fewer missed updates.

Alternatively, in Figure 71, we see the redundant mementos recorded by the web archive over the history of all pages in this study. Again, the Y-axis represents each URI in the dataset. The X-axis is time. Lighter colors indicate fewer redundant mementos on that day. We see the same vertical lines as in Figure 70, except, in this visualization, darker colors indicate that the Internet Archive is archiving when it should not. The same more aggressive archiving shown in Figure 70 is also archiving pages that have not updated. One could argue that this is a waste of resources, but we also must consider that possessing redundant copies of the same page is a goal of digital preservation, as seen in such projects as LOCKSS [86].

6.3 CONCLUSIONS

We have conducted a study showing that spoiler areas do indeed exist for fan wikis for users that browse past versions of these resources using the mindist heuristic. We have found that, for the wiki sites under consideration, that there is a mean 66% probability that one will end up with a spoiler if they use TimeGates supporting the mindist heuristic. We have also shown that wiki pages update at a rate that is an order of magnitude faster than the Internet Archive records them. Finally, because we have access to all revisions and all mementos we showed how many missed updates exist for each page and how the number of missed updates has gone down as the Internet Archive has gotten more aggressive.

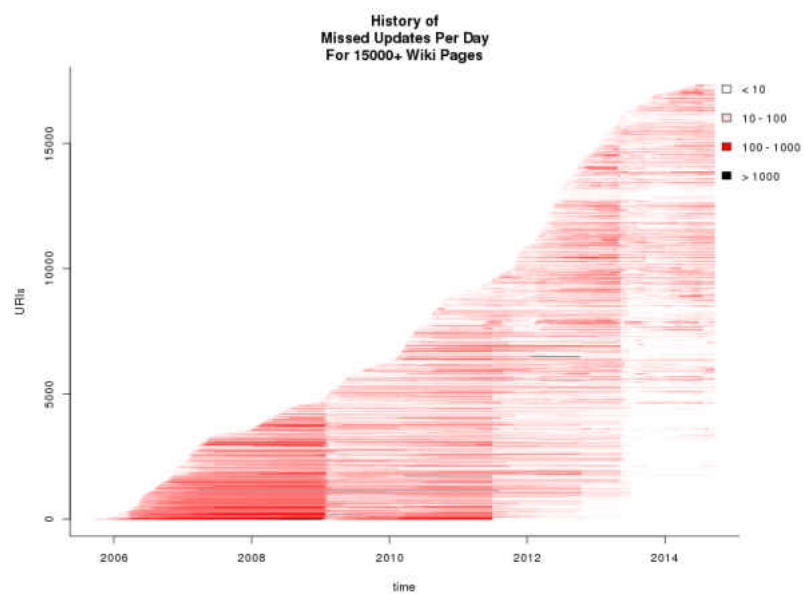


FIG. 70: Plot of missed updates for 16 wiki sites over time, lighter colors indicate few to no missed updates, darker colors indicate many missed updates

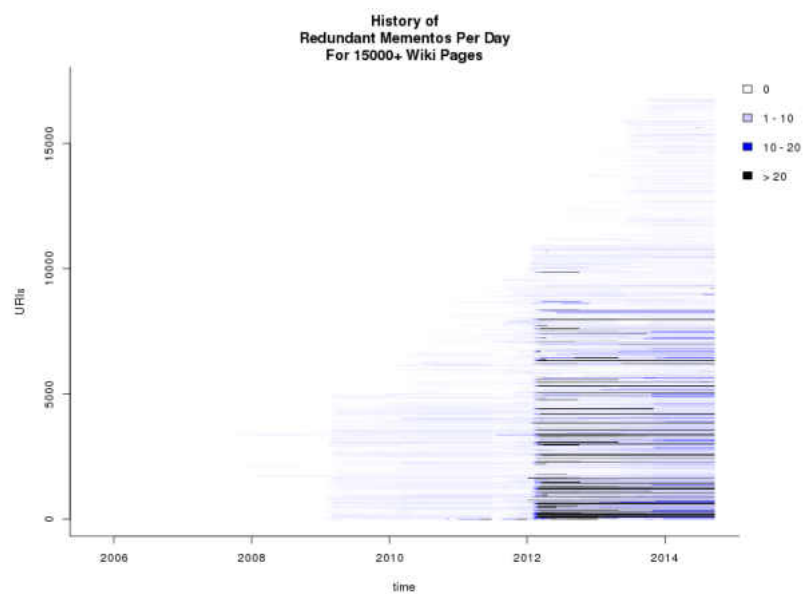


FIG. 71: Plot of redundant mementos for 16 wiki sites over time, lighter colors indicate few to no redundant mementos, darker colors indicate many redundant mementos

CHAPTER 7

MEASURING ACTUAL SPOILERS IN WAYBACK

MACHINE LOGS

The previous chapter addressed the probability of being redirected to spoilers if the mindist heuristic is used. In this chapter, we will show that users *are* being redirected to pages in the future. Fortunately, we have access to some of the logs from the Wayback Machine in 2011 and 2012. The data in these logs has been anonymized, but the URIs the users are attempting to visit and the URIs they came from are available. Using this information we can determine which datetime they intended to visit, and whether they ended up in the past or future.

Research has already been done by Ainsworth in how much drift exists within the web archive [3]. That study indicates that the Wayback Machine uses a **sliding target policy**. This means that each request is in some way based on the datetime of the last request, resulting in a user ending up in a much different datetime than they had originally started. The Wayback Machine still uses the mindist heuristic to determine which memento to deliver to a user, but it changes the desired datetime t_a based on the datetime of the memento from the last request.

Contrary to this, Memento uses a **sticky target policy**, allowing a user to fix the datetime t_a throughout their browsing session. The study found that there is still some small drift with the sticky target policy, but that it is constrained by the datetime remaining constant in each request. That drift is introduced only by the mindist heuristic, and the available mementos in the archive, rather than the sliding behavior of the Wayback Machine.

We are concerned about whether or not the user ended up in the future of where they intended. We want to know if they encountered a spoiler when using the Wayback Machine.

7.1 OUR METHOD FOR ANALYZING THE LOGS

As shown in Listing 7.1, the logs from the Wayback Machine are a standard Apache format. Each log entry corresponds to a single HTTP response to a request. Each



Memento from archive.org for URI-R
***http://www.example.com* on April 4, 2002 at 2:02:24 AM GMT**

FIG. 72: URI-M pattern for the Wayback Machine and Internet Archive

line is also separated by spaces into several fields. As noted by the caption, the datetime of the request is shown in **bold black**. The visited URI is in *red italics*. The status code of the request is shown in **bold orange**. The **referrer** is in *blue italics*.

The referrer is the URI that the user clicked on that brought them to the visited URI. Using this, we can track where the user came from and determine where they ended up. Fortunately for us, we can infer the desired datetime (referred to as t_a in previous chapters) and the memento-datetime from the URIs themselves. The Internet Archive allows access to all mementos using a standard URI format. Figure 72 shows the URI pattern used by the Internet Archive and the Wayback Machine to identify mementos. As we can see, the datetime is embedded in the URI. For the URI visited by the user, this datetime indicates the memento-datetime. For the referrer URI, this datetime indicates the desired datetime for the user.

Why do we say that we can *infer* the desired datetime? Without interviewing the visitors to the Wayback Machine, it is impossible to determine intent. The fact that the logs are anonymized makes this completely impossible. **We are making the assumption that some of the users receiving these responses intended to receive responses on the date that they started at, not the date delivered by the drift caused by the mindist heuristic.**

Using Listing 7.1 as an example, on line 1 the anonymized IP address 0.247.222.86 directly visited the April 4, 2002 02:02:24 GMT memento of the URI-R `http://www.example.com/page1.html` on February 2, 2012 at 07:03:55 GMT and

Listing 7.1: Example log entries from the Wayback Machine
 datetime of the request is shown in **bold black**

visited URIs are shown in *red italics*,

status code is shown in **bold orange**,

referrers are shown in *blue italics*

```

1 0.247.222.86 - - [02/Feb/2012:07:03:55 +0000] "GET
  http://web.archive.org/web/20020404020224/http://www.example.com/page1.html
  HTTP/1.1" 200 18875 "
  http://wayback.archive.org/web/*/http://www.example.com/page1.html " "
  Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/535.7 (KHTML,
  like Gecko) Chrome/16.0.912.77 Safari/535.7"
2 0.247.222.86 - - [02/Feb/2012:07:10:02 +0000] "GET
  http://web.archive.org/web/20020405015622/http://www.example.com/page2.html
  HTTP/1.1" 200 18875 "
  http://wayback.archive.org/web/20020404020224/http://www.example.com/page1.html"
  "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/535.7 (KHTML,
  like Gecko) Chrome/16.0.912.77 Safari/535.7"

```

received a status code of 200 (meaning successful). Note the time of the memento embedded in the URI-M. Also note that the datetime of the memento is April 4, 2002 and the datetime of the visit is February 2, 2012. That user arrived at that page by clicking on a link from `http://wayback.archive.org/web/*/http://www.example.com/page1.html`, which is a Wayback Machine listing of all available mementos for a given page, denoted by the `*` in the datetime part of the URI.

Further on line 2, the anonymized IP address 0.247.222.86 continued their journey by visiting the April 5, 2002 01:56:22 GMT memento of the URI-R `http://www.example.com/page2.html` on February 2, 2012 at 07:10:02 GMT and received a status code of 200 (meaning successful). That user arrived at that page by clicking on a link from `http://wayback.archive.org/web/20020404020224/http://www.example.com/page1.html`.

From these logs we can determine the inferred desired datetime from the referrer URI and the memento-datetime from the visited URI. Using this information, we can see how many requests end up in the future, meaning that those visitors are being redirected to spoilers via the Wayback Machine. Remember that the Wayback Machine uses the mindist TimeGate heuristic.

```

FINDSPOILERSINLOGFILE(logfile)
1  for each visitorID, visitedURI, referrer  $\in$  logfile
2       $t_m = \text{GETDATE}(\textit{visitedURI})$ 
3       $t_a = \text{GETDATE}(\textit{referrer})$ 
4       $\textit{wikidump} = \text{FETCHXMLDUMP}(\textit{title}, \textit{wikiURI})$ 
5       $\textit{revisions} = \text{EXTRACTREVISIONTIMES}(\textit{wikidump})$ 
6       $t_r = \text{GETREVMATCHINGMEMENTO}(t_m, \textit{revisions})$ 
7       $\textit{spoiler} = \text{INDETERMINATE}$ 
8      if rev is not NULL
9           $\textit{spoiler} = (t_a < t_r)$ 
10     PRINT(visitorID + " , " + spoiler)

```

Algorithm 8: Algorithm for Detecting spoilers in Internet Archive Logs

All requests for archived pages from wikia.com were extracted from the logs, resulting in 1,180,759 requests. Of those requests, we removed all requests for images, JavaScript, style sheets, supporting wiki pages (such as Template, Category, and Special pages), and advertisements. This left us with 62,227 requests to review.

For those remaining wikia.com pages, we downloaded the wiki export files, as done in the previous experiment, mapped the visited URI to the request that it had archived, and compared the datetime of that revision with the inferred desired datetime. We use t_a to represent the inferred desired datetime, and t_r to represent the datetime of the wiki revision matching the visited URI in the Wayback Machine.

Each response can be split into three categories in terms of spoilers: (1) **spoiler** - $t_a < t_r$; (2) **safe** - $t_a \geq t_r$; (3) **indeterminate** - either the datetime for the revision or the referrer was not able to be determined, likely because the article or whole wiki was moved or no longer exists, or because of 503 HTTP status codes due to the size of the export file.

This process, shown in Algorithm 8 determines how many requests are either spoiler, safe, or indeterminate for each log file. Indeterminate entries make up the bulk of the data collected, but offer no meaningful insight into the spoiler problem, and are thus discarded.

TABLE 13: Specifications of the Test Machine Used to Process the Wayback Machine Logs

CPU Number	2
CPU Clock Speed	2.4 GHz
CPU Type	Intel Xeon E7330
RAM	2 GB
Operating System	Red Hat Enterprise Linux 6.5

From this study we found that roughly 19% of these requests to the Wayback Machine result in spoilers.

7.2 SUMMARY

Using this simple study of URIs, we have shown that spoilers have been encountered by users of the Wayback Machine.

Why are these results around 19% rather than 50% if the mindist heuristic is being used? One would think that each request has an equal chance of being redirected to the past or the future. The Wayback Machine, as it turns out, in an effort to save space, engages in *URL agnostic reduplication* whereby the first item of content with a particular URI-R is saved [83]. If another crawl to the same URI-R finds the exact same content, then the Wayback Machine directs users to the earliest one encountered. Due to this behavior, the Wayback Machine seems to favor the past, but not enough for reliably protecting a user from spoilers.

CHAPTER 8

PREVENTING SPOILERS WITH THE MEMENTO

MEDIAWIKI EXTENSION

In Chapter 5 we showed that it is probable that people are getting spoilers through mindist. In Chapter 6 we used data from actual wikis and the Internet Archive to determine what the probabilities are for getting spoilers. In Chapter 7 we showed that people are actually encountering spoilers via the Wayback Machine. Now we document a solution to the issue.

We could change the TimeGates used for the web archives to all use the minpast heuristic, thus allowing everyone to avoid spoilers, but not all users are trying to avoid spoilers. Some users just want a memento that is close enough to their desired datetime, and because web archives are sparse, we do not have confidence that minpast will produce a memento that is a good representation of what the page looked like at their desired datetime.

Alternatively, realizing that wikis have access to all revisions, we produced an extension to MediaWiki that uses the minpast heuristic. Fortunately, Memento was created with the intent of being used by both web archives and content management systems (CMS) [103]. We chose MediaWiki because it is the wiki software used by Wikipedia and Wikia, where most fans encounter spoilers. In this section, we will discuss the design of this extension, performance testing that we conducted to indicate that it has a minimal impact on existing MediaWiki installations, and we also show how it can make some headway in solving the problem of temporal coherence [4]. We care about temporal coherence because embedded images may contain spoilers in their own right.

With this extension installed, a fan of fiction can avoid spoilers in the fan-based wiki of their choice.

8.1 DESIGN AND ARCHITECTURE

We took great care in creating a MediaWiki extension that might actually be

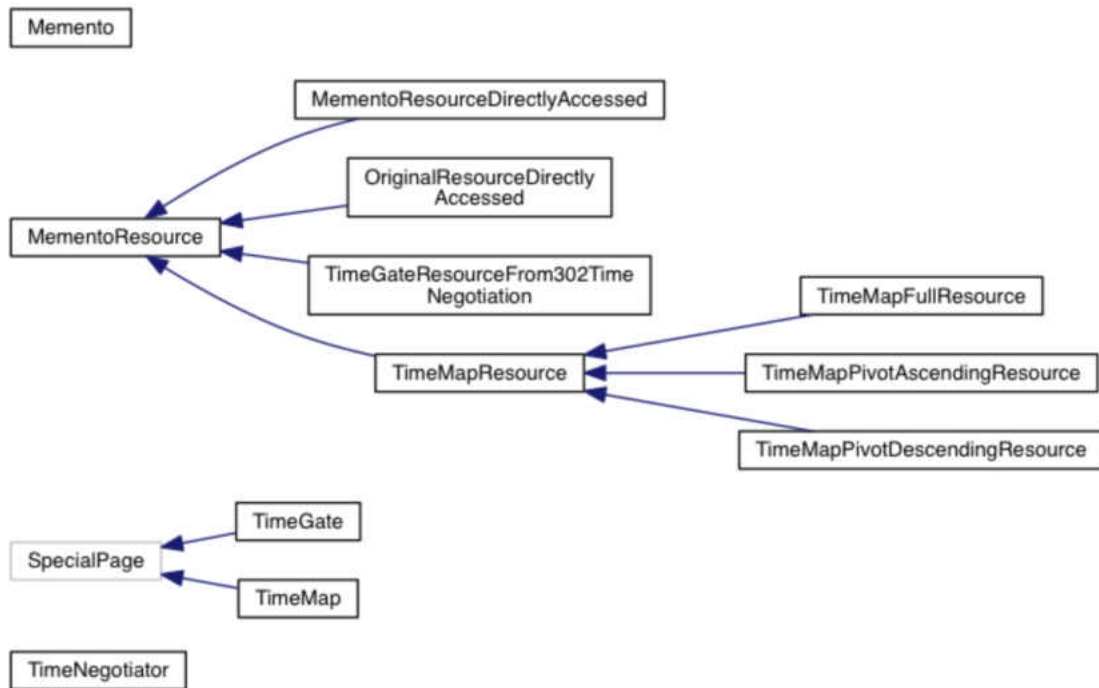


FIG. 73: Memento MediaWiki Extension Class Hierarchy Diagram

desirable to MediaWiki [48]. We sought to ensure that the extension followed MediaWiki’s coding conventions [58, 59, 112], followed MediaWiki’s security checklist [91], and did not require changes to core MediaWiki code [87].

Figure 73 shows the architecture of the Memento MediaWiki extension as created to address these concerns.

The Memento MediaWiki Extension partitions functionality into individual classes so that MediaWiki’s objects and functions could be consumed and utilized more efficiently, increasing performance while also allowing for extensibility.

The `Memento` class is the extension entry point for URI-R and URI-M work, implementing a *Mediator* design pattern [29]. It uses the `BeforeParserFetchTemplateAndtitle` hook [62] to ensure that the revision of an embedded article template matches the revision of the wiki article. It uses the `ImageBeforeProduceHTML` hook [63] to ensure that the revision of an embedded image matches the revision of the wiki article. It uses the `ArticleViewHeader` hook [60] to insert Memento headers into the responses.

Listing 8.1: Memento MediaWiki Extension Example Response for step 1 (URI-R) of Memento Pattern 2.1 (Memento headers in red)

```

HTTP/1.1 200 OK
Date: Sun, 25 May 2014 21:39:02 GMT
Server: Apache
X-Content-Type-Options: nosniff
Link: <http://ws-dl-05.cs.odu.edu/demo/index.php/Daenerys.Targaryen>;
      rel="original latest-version",
      <http://ws-dl-05.cs.odu.edu/demo/index.php/Special:TimeGate/Daenerys.Targaryen>;
      rel="timegate",
      <http://ws-dl-05.cs.odu.edu/demo/index.php/Special:TimeMap/Daenerys.Targaryen>;
      rel="timemap"; type="application/link-format"
Content-language: en
Vary: Accept-Encoding, Cookie
Cache-Control: s-maxage=18000, must-revalidate, max-age=0
Last-Modified: Sat, 17 May 2014 16:48:28 GMT
Connection: close
Content-Type: text/html; charset=UTF-8

```

MediaWiki provides a utility called a `SpecialPage` to perform specific functions not covered otherwise. When creating an extension, one may use these `SpecialPages` to centralize additional functionality, if necessary. Our extension created new `SpecialPages` as entry points for clients looking for `TimeGates` and `TimeMaps`.

Global variables are controlled using the `Memento` class. This way all extension configuration options (controlled as globals, as is the MediaWiki convention) are read and stored in one place in a controlled fashion. All other use of global variables have been removed from the code by using MediaWiki's native functions as much as possible.

As shown in Table 14 the `MementoResource` family of classes implement the different resource types used in the Memento framework. This architecture was chosen to improve code quality, while also supporting code extension and reusability. These classes, with the exception of `TimeGateResourceFrom302TimeNegotiation`, are selected based on the HTTP request using a *Factory Method*. This *Factory Method*, combined with a *Strategy* pattern, and utilizing *Template Methods*, makes sure the framework is easily extendable to include additional future patterns and resource types.

`TimeMaps` can be paged, allowing a machine client to follow one `TimeMap` to

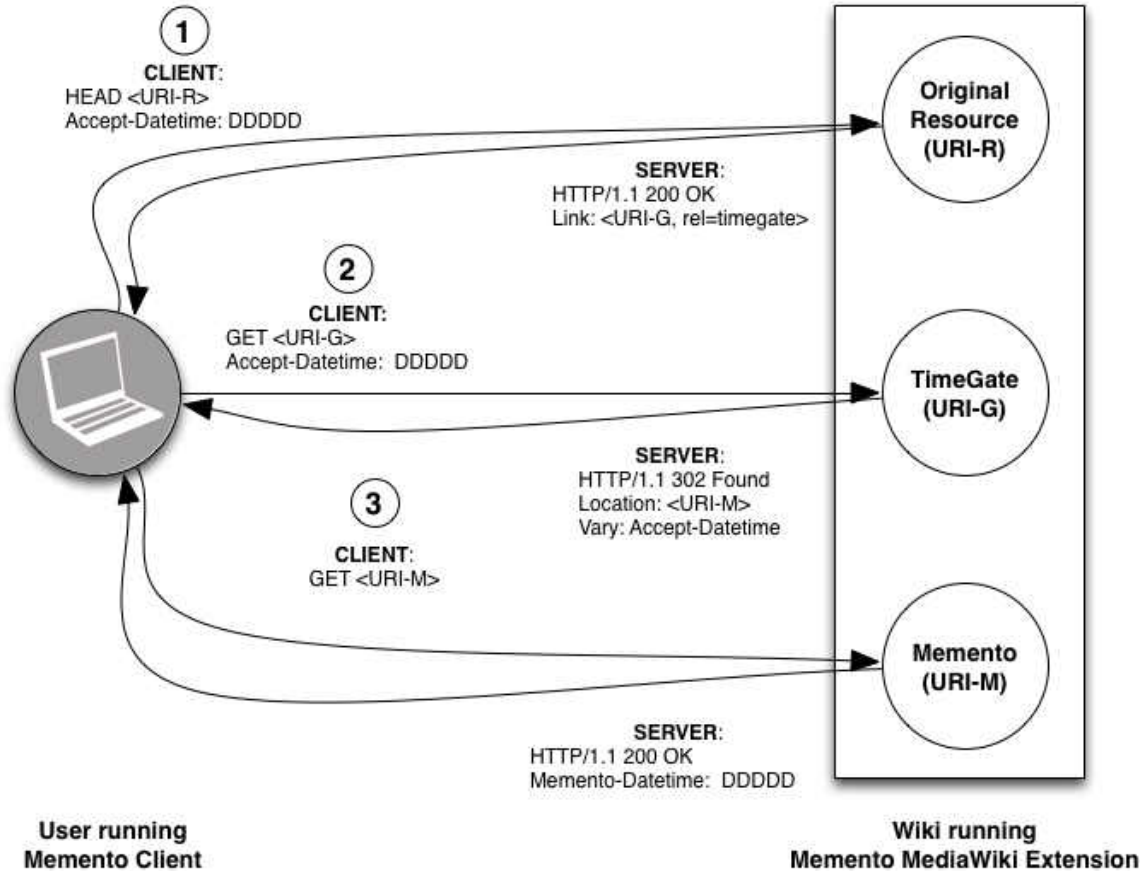


FIG. 74: Memento Pattern 2.1 Overview with Only Salient Headers, Methods, and Responses

another and another using the “follow your nose” principle of REST [25]. TimeMap URIs are constructed by the Memento MediaWiki Extension as shown in the examples in Table 15. Arguments, specified as part of the URI, indicate which TimeMaps should be returned. A `/-1/` following a datetime in the URI indicates that a TimeMap containing mementos prior to that datetime should be returned. A `/1/` following a datetime in the URI indicates that a TimeMap containing mementos after that datetime should be returned. A URI containing no datetime returns the latest Mementos for the given wiki article and a link to the next TimeMap, if there are more than 500 Mementos.

The TimeMap SpecialPage class also uses this same combination of design patterns to act according to how it are called. For, example, if the TimeMap SpecialPage is called using a `/-1/` following a datetime in the URI, then a

Listing 8.2: Memento MediaWiki Extension Example Response for step 2 (URI-G) of Memento Pattern 2.1 (Memento headers in red)

```

HTTP/1.1 302 Found
Date: Sun, 25 May 2014 21:43:08 GMT
Server: Apache
X-Content-Type-Options: nosniff
Vary: Accept-Encoding, Accept-Datetime
Location:
    http://ws-dl-05.cs.odu.edu/demo/index.php?title=Daenerys.Targaryen&oldid=1499
Link:
    <http://ws-dl-05.cs.odu.edu/demo/index.php/Special:TimeMap/Daenerys.Targaryen>;
rel="timemap"; type="application/link-format",
<http://ws-dl-05.cs.odu.edu/demo/index.php/Daenerys.Targaryen>;
rel="original latest-version"
Connection: close
Content-Type: text/html; charset=UTF-8

```

TimeMapPivotDescendingResource object is instantiated to provide paged TimeMaps below the given datetime. Likewise a /1/ following a datetime in the URI instantiates a TimeMapPivotAscendingResource object, providing paged TimeMaps above the given datetime. If no pivot is given in the URI, then a TimeMapFullResource object is instantiated, giving the full first page of the TimeMap from the current date.

The TimeNegotiator centralizes all time negotiation functionality. This way time negotiation is performed using the same algorithm across the entire extension, even if new classes are added for additional Memento patterns.

Table 16 provides example URIs that correspond to each of these resource types once the Memento MediaWiki Extension is installed.

Once this architecture was in place, we were able to address lingering design decisions.

8.1.1 TIMEGATE DESIGN DECISION

Two possible TimeGate design options were reviewed to determine which would be best suited to be the default pattern in the Memento MediaWiki Extension [47].

We evaluated the use of Pattern 1.1 and Pattern 2.1 from RFC 7089. Both patterns require a Memento client to find the URI-G from header information in the URI-R response.

Listing 8.3: Memento MediaWiki Extension Example Response for step 3 (URI-M) of Memento Pattern 2.1 (Memento headers in red)

```

HTTP/1.1 200 OK
Date: Sun, 25 May 2014 21:46:12 GMT
Server: Apache
X-Content-Type-Options: nosniff
Memento-Datetime: Sun, 22 Apr 2007 15:01:20 GMT
Link: <http://ws-dl-05.cs.odu.edu/demo/index.php/Daenerys.Targaryen>;
      rel="original latest-version",
      <http://ws-dl-05.cs.odu.edu/demo/index.php/Special:TimeGate/Daenerys.Targaryen>;
      rel="timegate",
      <http://ws-dl-05.cs.odu.edu/demo/index.php/Special:TimeMap/Daenerys.Targaryen>;
      rel="timemap"; type="application/link-format"
Content-language: en
Vary: Accept-Encoding, Cookie
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Cache-Control: private, must-revalidate, max-age=0
Connection: close
Content-Type: text/html; charset=UTF-8

```

Pattern 2.1 uses distinct URIs for URI-R and URI-G. Figure 74 shows a simplified diagram of a Pattern 2.1 exchange.

Pattern 1.1 uses the same URI for both URI-R and URI-G, allowing a resource to function as its own TimeGate, meaning that the client can short-circuit the process by one request.

Version 1.0 of the Memento MediaWiki Extension utilized Pattern 2.1, but Pattern 1.1 was explored to save on network traffic and improve performance.

As can be seen in Figure 74, Pattern 2.1 requires three request-response pairs to retrieve a Memento.

$$d_{p2.1} = a + RTT_a + b + RTT_b + M + RTT_M \quad (15)$$

Equation 15 calculates the duration of using Pattern 2.1, where a is time the Memento MediaWiki Extension takes to generate the URI-R response in step 1, b is the time it takes to generate the URI-G response in step 2, and M is the time it takes to generate the URI-M response in step 3. RTT_a , RTT_b , and RTT_M is defined as *round-trip-time*, which is “the time it takes for a small packet to travel from client to server and then back to the client” [51], for transmitting the data computed during a , b , and M .

TABLE 14: Version 2.0 Memento MediaWiki Extension MementoResource Class Family Members Mapped To Their Memento Resource Type

Extension Class	Memento Resource Type
MementoResourceDirectlyAccessed	URI-M
OriginalResourceDirectlyAccessed	URI-R
TimeGateResourceFrom302TimeNegotiation	URI-G
TimeMapResource (class family): TimeMapFullResource TimeMapPivotAscendingResource TimeMapPivotDescendingResource	URI-T

Figure 75 shows a simplified diagram of Pattern 1.1, which requires two request-response pairs to retrieve a Memento.

$$d_{p1.1} = B + RTT_B + M + RTT_M \quad (16)$$

Equation 16 calculates the duration for using Pattern 1.1, where B is the time it takes to generate the URI-G response in step 1. Just like in Equation 15, M and RTT_M are the same. The term RTT_B is the round-trip time to receive and transmit the results of the calculation done during B .

Our intuition was that Pattern 1.1 should be faster. It has fewer round trips to make between the client and server.

For Pattern 1.1 to be the better choice for performance, $d_{p1.1} < d_{p1.2}$, which leads to Equation 17.

$$\begin{aligned}
 d_{p1.1} &< d_{p1.2} \\
 B + RTT_B + \cancel{M} + \cancel{RTT_M} &< a + RTT_a + \\
 &\quad b + RTT_b + \cancel{M} + \cancel{RTT_M} \\
 B + RTT_B &< a + RTT_a + b + RTT_b \quad (17)
 \end{aligned}$$

TimeGate responses consist of 302 status messages in response to a GET request. The difference between the number of bytes in a request and response conversation

TABLE 15: Examples of TimeMap URIs From the Memento MediaWiki Extension

Meaning	Relative TimeMap URI
Get TimeMap for the latest 500 Mementos for the wiki article “Daenerys Targaryen”	/index.php/Special:TimeMap/Daenerys_Targaryen
Get TimeMap for the 500 Mementos (or less) prior to June 30, 2011 at midnight	/index.php/Special:TimeMap/20110630000000/-1/Daenerys_Targaryen
Get TimeMap for the 500 Mementos (or less) after June 30, 2011 at midnight	/index.php/Special:TimeMap/20110630000000/1/Daenerys_Targaryen

should differ only by a few bytes at most between Pattern 1.1 and 2.1. If we consider that a TimeGate response will be equivalent regardless of pattern implemented, then $RTT_B \simeq RTT_b$. This brings us to Equation 18.

$$\begin{aligned}
 B + \cancel{RTT_B} &< a + RTT_a + b + \cancel{RTT_b} \\
 B &< a + RTT_a + b \\
 B &< a + b + RTT_a
 \end{aligned}
 \tag{18}$$

Thus, to determine if Pattern 1.1 is actually better, we need to find values for B (Pattern 1.1 duration for datetime negotiation), a (time to respond to the initial HEAD request in Pattern 2.1), b (Pattern 2.1 duration for datetime negotiation), and RTT_a (the round trip time for the HEAD request during the first step in Pattern 2.1).

Caching Concerns

After review of the Wikimedia architecture, it also became apparent that caching was an important aspect of our design and architecture plans. Because the initial

TABLE 16: Examples of Memento Resources From the Memento MediaWiki Extension

Memento Resource Type	Memento Resource Notation	Relative URI Example
Original Resource	URI-R	/index.php/Daenerys.Targaryen
Memento	URI-M	/index.php?title=Daenerys.Targaryen&oldid=27870
TimeGate	URI-G	/index.php/Special:TimeGate/Daenerys.Targaryen
TimeMap	URI-T	/index.php/Special:TimeMap/Daenerys.Targaryen

architecture implemented Pattern 2.1 and 302 responses are not supposed to be cached [21], caching was not of much concern. Now that we have decided to pursue Pattern 1.1, it becomes even more important.

Experiments with Varnish (the caching server used by Wikimedia [110]) indicate that the *Vary* header correctly indicates what representations of the resource are to be cached. If the URI-R contains a *Vary* header with the value *Accept-Datetime*, this indicates to Varnish that it should cache each URI-R representation in response to an *Accept-Datetime* in the request for that URI-R. Other values of the *Vary* header have a finite number of values, but *Accept-Datetime* can have a near-infinite number of values (i.e., all datetimes in the past), making caching near useless for Pattern 1.1.

Those visitors of a URI-R that do not use *Accept-Datetime* in the request header will be able to reap the benefits of caching readily. Memento users of system using Pattern 1.1 will scarcely reap this benefit, because Memento clients send an initial *Accept-Datetime* with every initial request.

Caching is important to our duration equations because a good caching server returns a cached URI-R in a matter of milliseconds, meaning our value of a in Equation 18 is incredibly small, on the order of 0.1 seconds on average from our test server.

Pattern 1.1 vs. Pattern 2.1 URI-G Performance

The next step was to get a good set of values for b , URI-G performance for Pattern 2.1, and B , URI-G performance for Pattern 1.1.

To get a good range of values, we conducted testing using the benchmarking tool

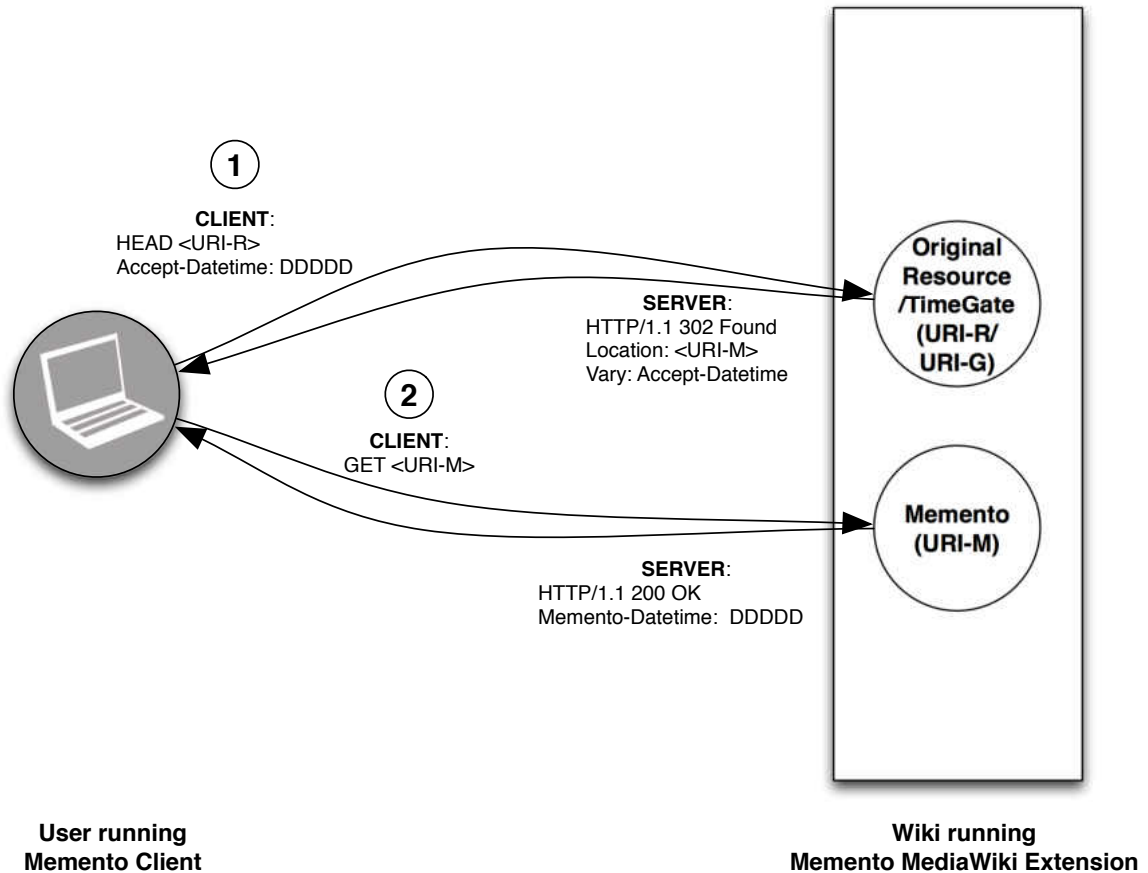


FIG. 75: Memento Pattern 1.1 Overview with Only Salient Headers, Methods, and Responses

Siege [28] on our demonstration wiki. The test machine was a virtual machine with the specifications listed in Table 17. The test machine consists of two installs of MediaWiki containing the Memento MediaWiki Extension: one utilizing Pattern 2.1 and the second implemented using Pattern 1.1. The data used in the test wikis came from *A Wiki of Ice and Fire*, consisting on many articles about the popular *A Song of Ice and Fire* book series.

Both TimeGate implementations use the same `TimeNegotiator` class, as shown in the architecture from Figure 73. They only differ in where this class is called. The Pattern 1.1 implementation uses the `ArticleViewHeader` hook [60] to instantiate this class and perform datetime negotiation. The Pattern 2.1 implementation utilizes a MediaWiki `SpecialPage` [66] at a separate URI to instantiate this class and perform datetime negotiation.

TABLE 17: Specifications of the Test Machine Used to Compare Pattern 1.1 vs. Pattern 2.1 URI-G Performance

CPU Number	2
CPU Clock Speed	2.4 GHz
CPU Type	Intel Xeon E7330
RAM	2 GB
Operating System	Red Hat Enterprise Linux 6.5
Apache HTTP Server Version	2.2.15
PHP Version	5.3.3

Listing 8.4: Example of Siege output

```

HTTP/1.1 302 0.60 secs: 0 bytes ==> GET /demo-special/index.php/
Special:TimeGate/Daenerys
HTTP/1.1 200 3.10 secs: 95662 bytes ==> GET /demo-special/index.php?title=
Daenerys&oldid=27870
HTTP/1.1 302 3.41 secs: 0 bytes ==> GET /demo/index.php/Daenerys
HTTP/1.1 200 1.86 secs: 94558 bytes ==> GET /demo/index.php?title=Daenerys
&oldid=27870

```

Tests were performed against *localhost* to avoid the benefits of using the installed Varnish caching server. By doing this, we see the true processing times from MediaWiki for TimeGate response generation. Also, caching was disabled in MediaWiki to avoid skewing the results.

Siege was run against 6304 different articles in the demonstration wiki. The date of Mon, 30 Jun 2011 00:00:00 GMT was used for datetime negotiation. This date corresponds to the release of the book *A Dance With Dragons* which came out after the wiki had an established base of users. A flurry of activity should occur around and after that date. All previous books in the *A Song of Ice and Fire* series were released prior to the wiki's creation.

Listing 8.4 gives an example of the output from Siege. This output was further

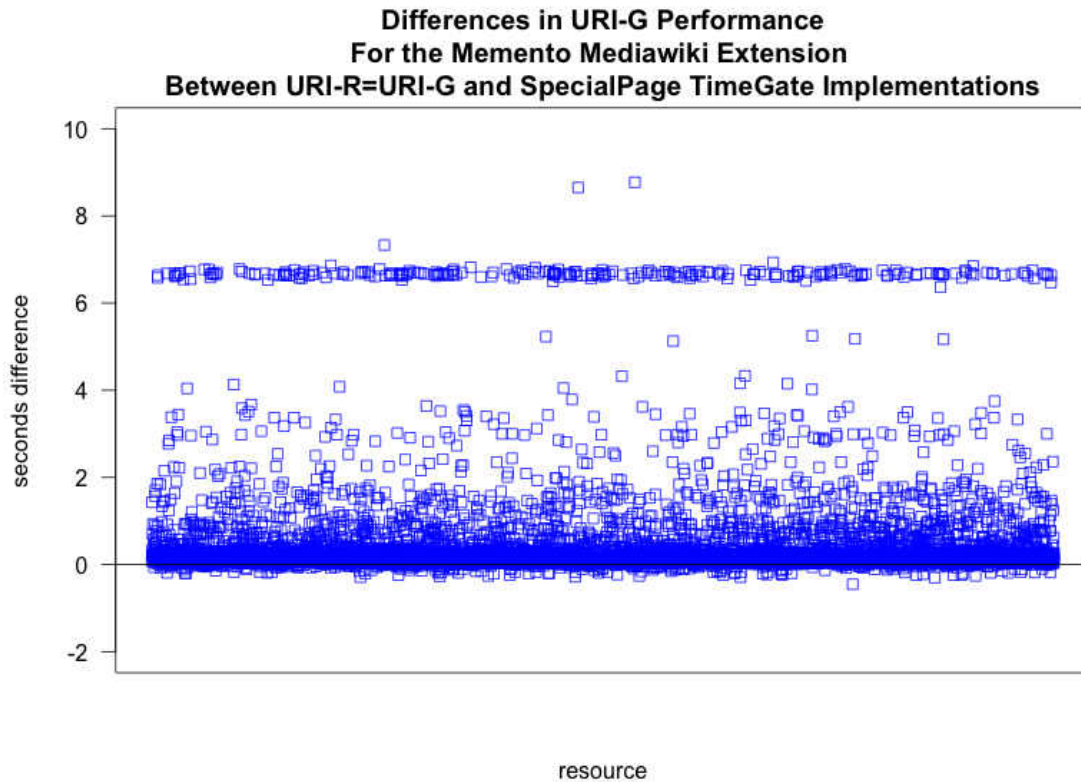


FIG. 76: Differences in URI-G performance between Pattern 1.1 and 2.1

processed using a Python script which extracted all of the 302 responses, which correspond to those instances of datetime negotiation (the 200 responses are just Siege dutifully following the 302 redirect). The URI then indicates which edition of the Memento MediaWiki Extension is installed, differing only in their TimeGate implementation. URIs beginning with `/demo-special` use Pattern 2.1. URIs beginning with `/demo` use Pattern 1.1. From these lines we can compare the amount of time it takes to perform datetime negotiation using each design option.

Figure 76 shows the results of this analysis. The plot shows the difference between the Pattern 1.1 and Pattern 2.1 processing times. Seeing as most values are above 0, it appears that there is a marked benefit to using Pattern 2.1. The string of values around 7 seconds difference are all Wiki redirect pages, leading one to infer that redirection is especially expensive with Pattern 1.1.

Figure 77 contains a histogram with 12 buckets containing the range of processing time values for Pattern 1.1.

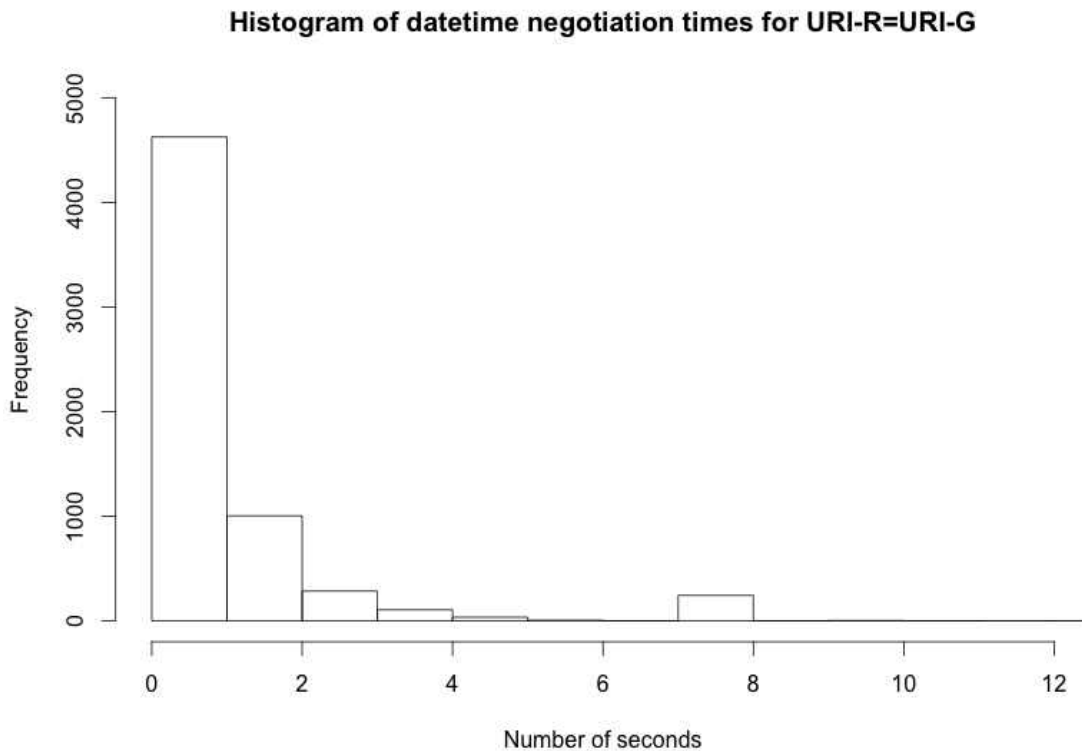


FIG. 77: Histogram showing Pattern 1.1 values

Figure 78 contains another histogram with 12 buckets for comparison, showing the range of processing time values for Pattern 2.1.

Table 18 shows the statistics from the testing. We now have values for b and B , so $0.22 \leq b \leq 1.75$ and $0.56 \leq B \leq 12.06$ for Equation 18. Of course, the processing time varies based on page size, number of revisions, and other factors.

The high side of the range of values shown for Pattern 1.1 from Table 18 and Figure 77 exceed those acceptable to the MediaWiki performance guidelines [80]. This also leads one to infer that the cost of using Pattern 1.1 may not be acceptable to the Wikimedia team.

Round Trip Time

Our final missing term from Equation 18 is RTT_a . RTT is a combination of *transmission delay* (d_t), *propagation delay* (d_p), queuing delay, and processing delay [51]. For the purposes of this paper, we are ignoring queuing delay and processing

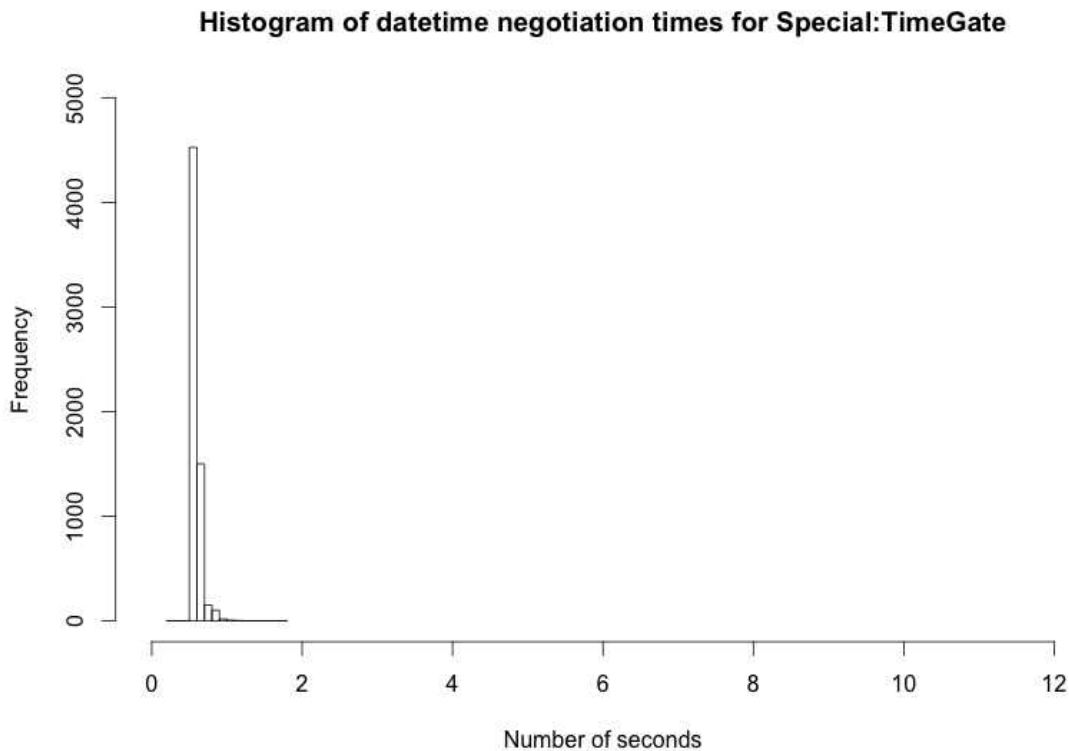


FIG. 78: Histogram showing Pattern 2.1 values

delay, as those are dependent of the router infrastructure of the Internet and are typically negligible, thus we are reduced to Equation 19.

$$RTT = d_t + d_p \quad (19)$$

And transmission delay is a function of the number of bits (N) divided by the rate of transmission (R) [51], shown in Equation 20.

$$d_t = \frac{N}{R} \quad (20)$$

Listing 8.5 shows an example Pattern 2.1 HEAD request. Considering cookies and other additional data, the average initial Pattern 2.1 HEAD request consists of the 700 Byte HTTP request + a 20 Byte TCP header [96] + a 20 Byte IP header [96]. This gives a total payload of 740 Bytes or 5920 bits. Thus our request transmission delay is $d_{trq} = 5920 \text{ b}/R$.

Listing 8.6 shows an example Pattern 2.1 200 status code reply. Considering variability within the Link header relation entries, the average initial Pattern 2.1

Listing 8.5: Example HTTP Request for RTT_a

```

HEAD /demo/index.php/Daenerys_Targaryen HTTP/1.1
Host: ws-dl-05.cs.odu.edu
Accept: */*
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
Cookie: __utma=99999999.999999999.999999999.999999999.999999999.99;
__utms=99999999.999999999.9.9.utmcsr=example.com|utmccn=(referral)|utmcmd=
referral|utmctt=/; __utma
=999999999.999999999.999999999.999999999.999999999.9; __utmz
=999999999.999999999.9.9.utmcsr=example|utmccn=(organic)|utmcmd=organic|
utmctr=(not%20provided); __atuv=99%7C99%2C99%7C99%2C99%7C99%2C99%7C99%7
C99
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (
KHTML, like Gecko) Chrome/34.0.1847.131 Safari/537.36

```

Listing 8.6: Example HTTP Response for RTT_a

```

HTTP/1.1 200 OK
Age: 0
Cache-Control: s-maxage=18000, must-revalidate, max-age=0
Connection: keep-alive
Content-language: en
Content-Type: text/html; charset=UTF-8
Date: Tue, 06 May 2014 02:57:35 GMT
Last-Modified: Sat, 22 Mar 2014 02:47:30 GMT
Link: <http://ws-dl-05.cs.odu.edu/demo/index.php/Daenerys_Targaryen>; rel="
original latest-version",<http://ws-dl-05.cs.odu.edu/demo/index.php/Special
:TimeGate/Daenerys_Targaryen>; rel="timegate",<http://ws-dl-05.cs.odu.edu/
demo/index.php/Special:TimeMap/Daenerys_Targaryen>; rel="timemap"; type="
application/link-format"
Server: Apache/2.2.15 (Red Hat)
Vary: Accept-Encoding,Accept-Datetime,Cookie
Via: 1.1 varnish
X-Content-Type-Options: nosniff
X-Powered-By: PHP/5.3.3
X-Varnish: 2138031585

```


TABLE 18: Statistics on Pattern 1.1 vs. Pattern 2.1 TimeGate testing results

	Pattern 1.1	Pattern 2.1
Min	0.56	0.22
Max	12.06	1.75
Mean	1.24	0.6
Median	0.77	0.59

response consists of a 700 Byte HTTP response + a 20 Byte TCP header + a 20 Byte IP header. This gives a total payload of 740 Bytes or 5920 bits. Thus our response transmission delay $d_{trs} = 5920 b/R$.

Seeing as both share the same denominator, our total transmission delay $d_t = d_{trq} + d_{trs} = 5920 b/R + 5920 b/R = 11840 b/R$.

Assuming an average-to-worst case of 1G wireless telephony (28,800 bps), the end user would experience a transmission delay of $d_t = 11840 b/28800 bps = 0.41 s$. Combining this with our average case for both TimeGate patterns from the previous section, $b = 0.6 s$ and $B = 1.24 s$, and using $a = 0.1$ from the caching results, we get Equation 21.

$$\begin{aligned}
B &< RTT_a + a + b \text{ From (18)} \\
B &< d_p + d_t + a + b \text{ From (19)} \\
1.24 s &< d_p + d_t + 0.1 s + 0.6 s \\
1.24 s &< d_p + 0.41 s + 0.1 s + 0.6 s \\
1.24 &< d_p + 1.11 s \tag{21}
\end{aligned}$$

So, an end user with 1G wireless telephony would need to experience an additional 0.13 s of propagation delay in order for Pattern 1.1 to be comparable to Pattern 2.1.

Propagation delay is a function of distance and propagation speed, as shown in Equation 22.

$$d_p = \frac{d}{s_p} \tag{22}$$

Seeing as wireless telephony travels at the speed of light, the distance one would need to transmit a signal to make Pattern 1.1 viable becomes $80944 km = 50296.3 miles$

as shown in Equation 23.

$$\begin{aligned}
 0.13 \text{ s} &= \frac{d}{299792458 \text{ m/s}} \\
 (0.13 \text{ s})(299792458 \text{ m/s}) &= d \\
 d &= 38973019.54 \text{ m} = 38973 \text{ km} = 24216.7 \text{ miles}
 \end{aligned} \tag{23}$$

This is almost the circumference of the Earth [92]. Even if we used copper wire (which has a worse propagation delay) rather than radio waves, the order of magnitude is the same. Considering the amount of redundancy on the Internet, the probability of hitting this distance is quite low, meaning that propagation delay will likely be so small that we will ignore it for the rest of this discussion.

That brings us back to transmission delay. At what transmission delay, and essentially what bandwidth, does Pattern 1.1 win out over Pattern 2.1 using our average values for b and B ?

$$\begin{aligned}
 B &< d_t + a + b \text{ From (18) and (19), removing } d_p \\
 1.24 \text{ s} &< d_t + 0.1 \text{ s} + 0.6 \text{ s} \\
 1.24 \text{ s} &< d_t + 0.7 \text{ s} \\
 0.54 \text{ s} &< d_t \\
 d_t &= \frac{N}{R} \text{ From (20)} \\
 0.54 \text{ s} &= \frac{11840 \text{ b}}{R} \\
 (0.54 \text{ s})(R) &= 11840 \text{ b} \\
 R &= \frac{11840 \text{ b}}{0.54 \text{ s}} = 21926 \text{ bps}
 \end{aligned} \tag{24}$$

Thus, the bandwidth for which Pattern 1.1 would begin to be useful would be anything at the speed less than 1G telephony, but would become produce increasingly poorer performance for bandwidths higher than that.

TimeGate Design Conclusion

From the data gathered and the experiments run, used in Equations 18, 19, and 20, Pattern 1.1 takes too much processing time to be viable, in spite of the saved *RTT*. It comes down to the values of b (processing time for Pattern 2.1) vs. B (processing time for Pattern 1.1), and B is greater in many cases.

Why the big difference? It turns out that the `ArticleViewHeader` hook used in the Pattern 1.1 implementation runs after MediaWiki has loaded all of the page data. The Pattern 2.1 implementation extends a `SpecialPage`, which has loaded nothing, and can start processing immediately.

Why not use a hook that is run before all of the page data is loaded? We need a hook that provides MediaWiki's `WebRequest` object for processing the *Accept-Datetime* request header. It also needs to provide MediaWiki's `WebResponse` object for producing the 302 response. Hooks earlier in the processing chain do not appear to provide this capability. We prototyped an implementation using the `BeforeInitialize` hook [61] and it did not preserve the needed response headers, nor did it perform better. Attempts to find earlier hooks by asking the MediaWiki development team have met with no success [45].

If a MediaWiki hook were available that gave the same performance for Pattern 1.1 as for Pattern 2.1 then transmission delay would no longer matter, and Pattern 1.1 would clearly be the best choice, as we see from Equation 25, because transmission delay would always be greater.

$$\begin{aligned}
 B &< d_t + a + b && \text{From (18) and (19), removing } d_p \\
 b &< d_t + 0.1 s + b && \text{Replacing } B \text{ with mean of } b \\
 b - b &< d_t + 0.1 s + b - b \\
 0 &< d_t + 0.1 s && (25)
 \end{aligned}$$

Of course, the processing time is not the only issue here; the use of Pattern 1.1 would make caching useless for Memento users of URI-Rs, considering Memento clients send an *Accept-Datetime* with each request, and there are a near infinite number of values for *Accept-Datetime*.

8.2 PERFORMANCE IMPACT ON MEDIAWIKI INSTALLATIONS

Once we completed initial development on the Memento MediaWiki Extension, we turned our focus to its impact on performance. We used Siege again, as in the TimeGate design experiment. The same machine as shown in Table 17 was used to run these performance tests, and the same demonstration wiki provided the test data.

As URI-Gs were tested during the TimeGate design experiment, we focused our attention on the other Memento resource types.

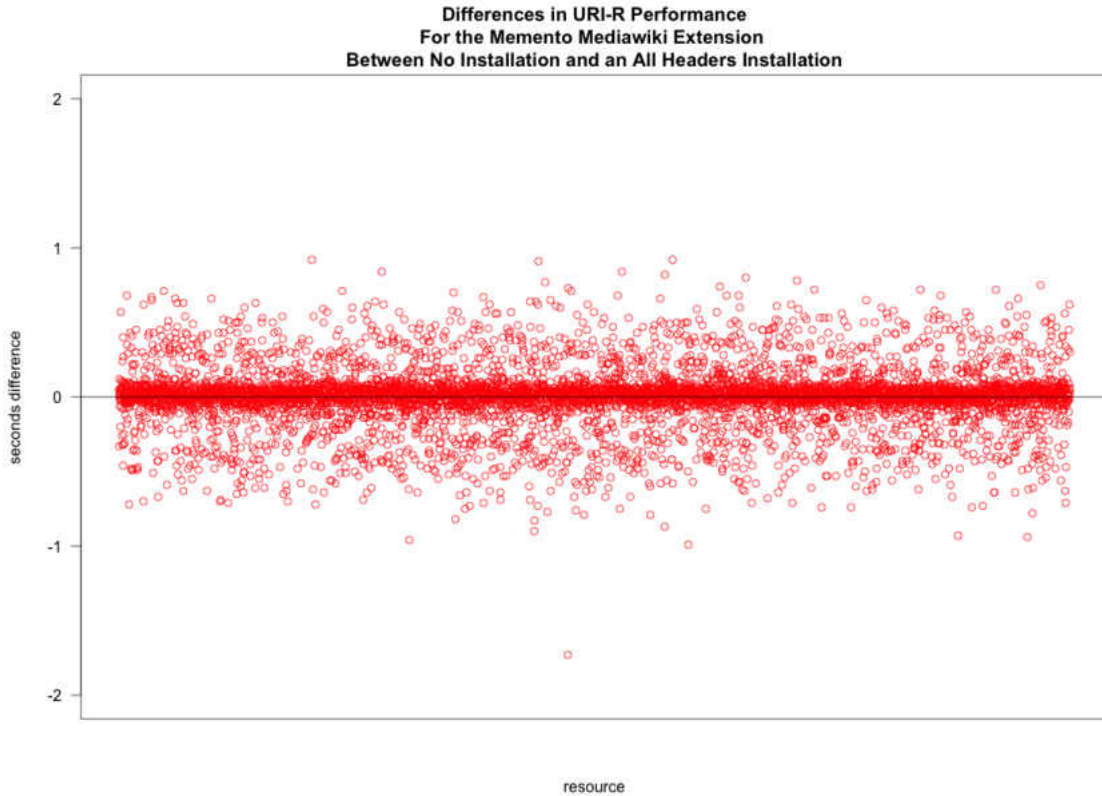


FIG. 79: Plot showing the difference in times for URI-Rs between a MediaWiki installation containing our extension vs one without it installed

8.2.1 URI-R PERFORMANCE

First, we look at the results for URI-Rs. These are the base wiki article pages. All the Memento MediaWiki Extension does is add Memento headers to these pages for a Memento client's benefit, informing the client of the URI for the TimeGate and TimeMap, and, in the case where all headers are enabled, first and last mementos.

Figure 79 shows the difference in seconds between accessing a wiki page's URI-R with the Memento MediaWiki installed and accessing the same wiki page without the extension loaded. Each point on the plot is one of 6480 different pages from the test wiki, and again they are evenly arranged around the 0 mark. The plots are evenly arranged around the 0 mark, with most of the points between 0.7 and -0.7. This means that installing the extension has a negligible impact on performance of URI-Rs. If the extension seriously impacted performance, then most of the plots should be above the 0 mark.

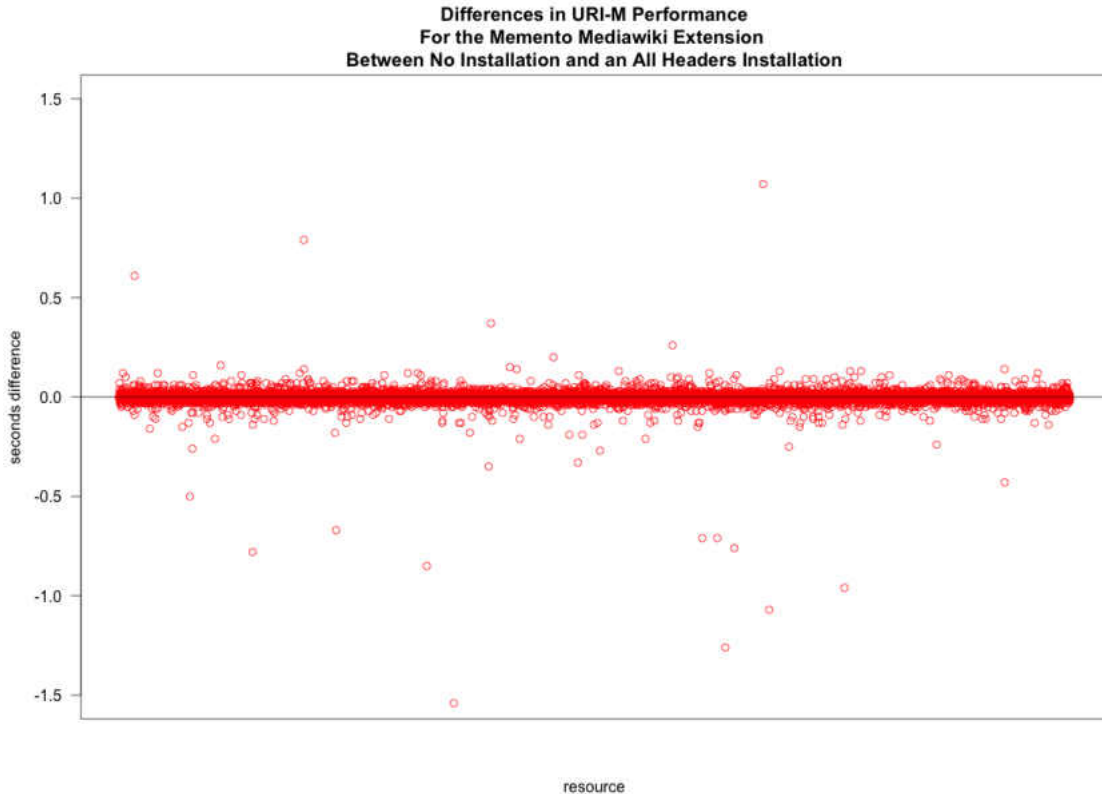


FIG. 80: Plot showing the difference in times for URI-Ms between a MediaWiki installation containing our extension vs one without it installed

8.2.2 URI-M PERFORMANCE

Secondly, we look at the results for URI-Ms, or *oldid pages*. This is the other Memento resource type that MediaWiki natively implemented already. Just like with URI-Rs, the Memento MediaWiki Extension adds Memento headers to these pages for a Memento client’s benefit, informing the client of the URI for the TimeGate and TimeMap, and, in the case where all headers are enabled, first and last mementos.

Figure 80 shows the difference in seconds between accessing a URI-M (or oldid page in MediaWiki parlance) with only mandatory Memento headers enabled and accessing the same page without the extension installed. Each point on the plot is one of 10257 different oldid pages from the test wiki. These plots are also arranged around the 0 mark, with most of the points between -0.25 and 0.25. This means that installing the extension has a negligible impact on URI-Ms. Again, if the extension seriously impacted performance, then most of the plots should be above the 0 mark.

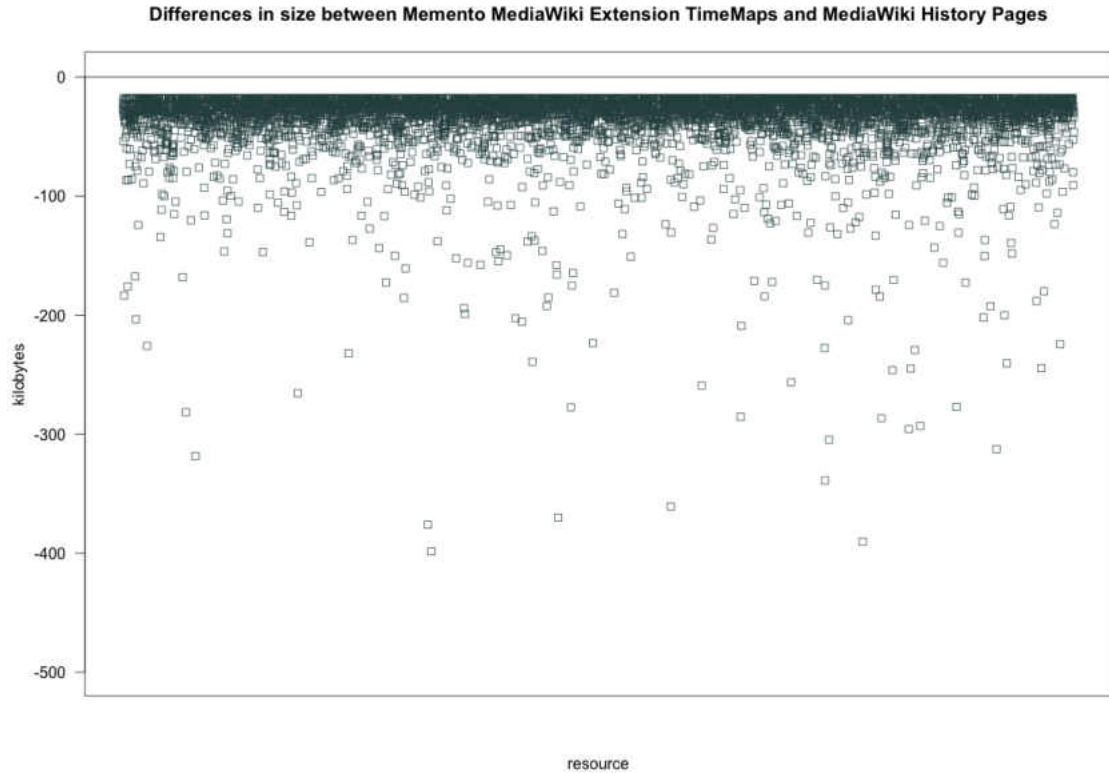


FIG. 81: Plot showing the difference in size between MediaWiki history pages and TimeMaps for the same article

8.2.3 URI-T PERFORMANCE

The closest thing to a Memento TimeMap (URI-T) in MediaWiki is a history page, but they are not really the same thing. The audience for history pages are humans, whereas the audience for TimeMaps are machine clients. Seeing as 80.8% of requests for TimeMaps come from machine clients [6], and 95% of machine clients download TimeMaps exclusively [7], there is interest in providing a machine readable format of the history page. To use a history page, a machine client would need to parse the HTML, performing unnecessary computation in order to get the same data provided much more succinctly by a TimeMap.

Again, we used Siege to download 6252 sample history pages and TimeMaps from our demonstration wiki. Figure 81 shows the difference in size between a MediaWiki history page and the corresponding TimeMap for the same article across 6252 sample pages. The mean in this sample is -34.7 kilobytes. This means, that if one were to

TABLE 19: Status of full temporal coherence among MediaWiki Entities

MediaWiki Entity	Status of Solution for Memento
Wiki Article	Complete in Extension
Template Page	Complete in Extension
Embedded Images	Complete as an Experimental Feature That Can Be Enabled
Embedded JavaScript	Requires change to MediaWiki source
Embedded StyleSheets	Requires change to MediaWiki source

solely rely upon a MediaWiki history page to acquire TimeMap data, they would need to parse through an additional unnecessary 35 kilobytes. In addition, there would be extra processing time given to stripping out the HTML and generating the TimeMap, which is a waste when a standard format TimeMap exists already.

Of course, one could also use the MediaWiki API to generate the information for TimeMaps, but TimeMaps provide URIs, whereas the MediaWiki API provides revision identifiers, which would require one to construct URIs in addition to parsing the API output in order to produce a TimeMap.

8.3 ATTEMPTS AT TEMPORAL COHERENCE

The Memento MediaWiki Extension is in a unique place to attempt to address the concept of temporal coherence. Web archives process a web page and retrieve the embedded resources at some point thereafter, which creates all kinds of problems when attempting to reconstruct the page to resemble its past revision [1]. MediaWiki has access to every revision of its embedded resources, therefore true temporal coherence should, in theory, be achievable for wiki revisions. To realize this, each MediaWiki URI-M must contain all of the correct revisions of those embedded images, JavaScript, and stylesheets that existed at the time the URI-M was saved. Table 19 shows the status of this work.

Temporal coherence is important to our study of spoilers because embedded images and other content can sometimes contain spoilers themselves, even though the article containing them does not.

As we show below, the temporal coherence of all Mementos served by MediaWiki



FIG. 82: Example Wikipedia page¹ with an embedded image that has been changed as the page content changes

is potentially a condition called *prima facie violative*, specifically the pattern *Right Newer Last-Modified*. This means that past revisions of a MediaWiki page contain the current revision of embedded resources.

The following sections highlight the issues of MediaWiki's temporal coherence in more detail.

8.3.1 EMBEDDED IMAGES

One of the problems we seek to address is the issue of embedded images [46]. MediaWiki allows one to store multiple versions of an embedded image under a single page name in the *File* namespace.

Figure 82 shows a screenshot of a Wikipedia page containing a map showing the legal status of Same-sex marriage law in the United States. The article content is changed as this issue unfolds, and the map is updated also to reflect the article content.


¹https://en.wikipedia.org/w/index.php?title=Same-sex_marriage_law_in_the_United_States_by_state&oldid=604205801


Same-sex marriage law in the United States by state

From Wikipedia, the free encyclopedia

This is an **old** revision of this page, as edited by 203.45.63.200 (talk) at 05:36, 5 June 2013. It may differ significantly from the current revision.

(diff) ← Previous revision | Latest revision (diff) | Newer revision → (diff)

 This article is in a list format that may be better presented using prose. You can help by converting this article to prose, if appropriate. Editing help is available. (April 2013)

 This article may require cleanup to meet Wikipedia's quality standards. The specific problem is: **Formatting and content layout in many sections does not follow WP:MOS.** Please help improve this article if you can. (April 2013)

This article summarizes the legal and political actions taken by the individual states of the United States regarding same-sex marriage. The texts are following.

Contents [hide]

- 1 Alabama
- 2 Alaska
- 3 Arizona
- 4 Arkansas
- 5 California
- 6 Colorado
- 7 Connecticut
- 8 Delaware
- 9 District of Columbia
- 10 Florida
- 11 Georgia
- 12 Hawaii
- 13 Idaho
- 14 Illinois
- 15 Indiana
- 16 Iowa
- 17 Kansas
- 18 Kentucky
- 19 Louisiana
- 20 Maine
 - 20.1 2012 initiative
- 21 Maryland
- 22 Massachusetts
- 23 Michigan



State laws regarding same-sex partnerships in the United States¹

- Same-sex marriage allowed¹
- Domestic partnerships or civil unions granting privileges similar to marriage for same-sex domestic partners²
- Limited/enumerated privileges granted by state
- Same-sex marriage performed elsewhere recognized
- No prohibition or recognition of same-sex marriage or unions in territory law
- Judicial ruling against a same-sex marriage ban stayed pending appeal³
- Statute bans same-sex marriage
- Constitution and statute ban same-sex marriage⁴
- Constitution and statute ban same-sex marriage and some or all other same-sex unions

FIG. 83: June 5, 2013 version of the example MediaWiki page² with an embedded image that is changed as the page content changes (note that the map is the same as in Figure 82, which does not match the article text)

If we access previous revisions of the MediaWiki page now, then it displays the *current* revision of the map, *not the one that goes with that revision of the article*.

What should be shown is the image shown in Figure 84 because it accurately reflects the content of the July 5, 2013 revision of the article.

²https://en.wikipedia.org/w/index.php?title=Same-sex_marriage_law_in_the_United_States_by_state&oldid=558400004

File history

Click on a date/time to view the file as it appeared at that time.
 (newest | oldest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)










	Date/Time	Thumbnail	Dimensions	User	Comment
current	21:48, 22 March 2014		959 × 593 (105 KB)	Bigdaddybrabantio	Michigan: https://twitter.com/AP/status/447475865589653504 , http://www.detroitnews.com/article/20140322/METRO06/303220039
	21:46, 22 March 2014		959 × 593 (105 KB)	Dralwik	Roll back Michigan
	15:55, 22 March 2014		959 × 593 (104 KB)	Dralwik	Implement beige stripes per en:File talk:Same-sex marriage in USA.svg#Proposal 3 (Change black to beige/gold)
	21:37, 21 March 2014		959 × 593 (104 KB)	Dralwik	Michigan struck down, no stay yet
	01:03, 20 March 2014		959 × 593 (105 KB)	Dralwik	Moving Kentucky to stayed ruling category
	01:24, 4 March 2014		959 × 593 (105 KB)	Dralwik	Implement stripes for states with stayed rulings per consensus at en:File talk:Same-sex marriage in USA.svg
	15:36, 1 March 2014		959 × 593 (105 KB)	SPQRobin	re-add Kentucky recognition, since it is equivalent to a law that has been signed but not taken effect yet
	08:40, 1 March 2014		959 × 593 (105 KB)	Plumber	revision until consensus is found
	08:39, 1 March 2014		959 × 593 (105 KB)	Plumber	proposed changes to the territories; switching up limited/foreign

FIG. 85: MediaWiki Page⁴ showing the map’s file history

Figure 85 shows that Wikipedia (and transitively, MediaWiki) has access to all of the previous revisions of the map. The data is present in the system, but MediaWiki does not present the previous version of the image with the previous version of the page.

MediaWiki provides the `ImageBeforeProduceHTML` hook, which provides a `$file` argument, giving access to the `LocalFile` object for the embedded image. It also provides a `$time` argument that signifies the “Timestamp of file in ‘YYYYMMDDHHIISS’ string form, or false for current” [63].

We wanted to use the `$time` argument, but were perplexed when the hook did not perform as expected, so we examined the source of MediaWiki version 1.22.5. Listing 8.7 shows the hook being called within the MediaWiki file `Linker.php`.

Listing 8.8 shows that the `$time` variable that we would set is passed to the `makeThumbLink2` function, also in the same file.

But, as shown in Listing 8.9, the value of `$time` is not really used. Instead, it is used to create a boolean value before being passed on to `makeBrokenLinkObj`.

⁴https://en.wikipedia.org/wiki/File:Same-sex_marriage_in_USA.svg

Listing 8.7: Where the ImageBeforeProduceHTML hook is called in Linker.php

```

569  if ( !wfRunHooks( 'ImageBeforeProduceHTML', array( &$dummy, &$title,&$file,
570      &$frameParams, &$handlerParams, &$time, &$res ) ) ) {
571      return $res;
    }

```

Listing 8.8: Where the variable \$time is passed after the hook is called

```

639  if ( isset( $fp['thumbnail'] ) || isset( $fp['manualthumb'] ) || isset( $fp[
        'framed'] ) ) {
640      # Create a thumbnail. Alignment depends on the writing direction of
641      # the page content language (right-aligned for LTR languages,
642      # left-aligned for RTL languages)
643      #
644      # If a thumbnail width has not been provided, it is set
645      # to the default user option as specified in Language*.php
646      if ( $fp['align'] == '' ) {
647          if ( $parser instanceof Parser ) {
648              $fp['align'] = $parser->getTargetLanguage()->alignEnd();
649          } else {
650              # backwards compatibility, remove with makeImageLink2()
651              global $wgContLang;
652              $fp['align'] = $wgContLang->alignEnd();
653          }
654      }
655      return $prefix . self::makeThumbLink2( $title, $file, $fp, $hp, $time,
        $query ) . $postfix;
656  }

```

Listing 8.9: Where the variable \$time is used to create a boolean value

```

861  if ( !$exists ) {
862      $s .= self::makeBrokenImageLinkObj( $title, $fp['title'], '', '', '', $time
        == true );
863      $zoomIcon = '';
864  } elseif ( !$thumb ) {

```

Listing 8.10: Where the variable \$time is again used to create a boolean value

```

674  if ( !$thumb ) {
675      $s = self::makeBrokenImageLinkObj( $title, $fp['title'], '', '', '', $time ==
        true );
676  } else {

```

Back inside the `makeImageLink` function, we see a second use of the `$time` value, as shown in Listing 8.10, but it is again used to create a boolean argument to the same function as seen in Listing 8.9.

Note that its timestamp value of `$time` in ‘YYYYMMDDHHIISS’ string form is never actually used as described. So, the documentation for the `ImageBeforeProduceHTML` hook is incorrect on the use of this `$time` argument. In fact, the hook was introduced in MediaWiki version 1.13.0 and this code does not appear to have changed much since that time. It is possible that the `$time` functionality is intended to be implemented in a future version.

Finally, we discovered a possible solution by instead using the `$file` object’s `getHistory()` function [68]. This function returns an array of the `File` objects representing each revision of an image. Even better, it takes `$start` and `$end` arguments, meaning that this function can do the datetime negotiation itself. Seeing as the `$file` argument is passed in by reference to the `ImageBeforeProduceHTML`, we can reassign the `File` object to the one in the array with the desired datetime, thus loading the correct image.

Our final solution requires more review, as one needs to purge the MediaWiki cache in order to view the correct revision of the image. We also need to determine how to retrieve the correct datetime for the URI-M base page that loads the image. For these reasons, images are not currently supported by the extension, but as noted in Table 19, this capability has been prototyped for the next version of the Memento MediaWiki Extension.

8.3.2 EMBEDDED JAVASCRIPT AND CSS

JavaScript and StyleSheets are the other embedded resources necessary to satisfy temporal coherence. MediaWiki natively stores all versions of stylesheets for use [65], as shown in Figure 86. MediaWiki also natively stores all versions of JavaScript to use [64], as shown in Figure 87.

Revision history of "MediaWiki:Common.css"

[View logs for this page](#)

Browse history

From year (and earlier): From month (and earlier): Deleted only

Diff selection: mark the radio boxes of the revisions to compare and hit enter or the button at the bottom.

Legend: **(cur)** = difference with latest revision, **(prev)** = difference with preceding revision, **m** = minor edit.

- (cur | prev) 20:49, 25 April 2012 Ran (Talk | contribs) (16,536 bytes)
- (cur | prev) 21:07, 10 February 2012 Ran (Talk | contribs) (16,286 bytes)
- (cur | prev) 21:06, 10 February 2012 Ran (Talk | contribs) (16,358 bytes)
- (cur | prev) 20:44, 10 February 2012 Ran (Talk | contribs) (15,941 bytes)
- (cur | prev) 13:53, 9 October 2011 Ran (Talk | contribs) (15,595 bytes)
- (cur | prev) 13:52, 9 October 2011 Ran (Talk | contribs) (15,143 bytes)
- (cur | prev) 11:06, 9 October 2011 Ran (Talk | contribs) (15,093 bytes)
- (cur | prev) 19:30, 8 October 2011 Ran (Talk | contribs) (15,211 bytes)
- (cur | prev) 19:29, 8 October 2011 Ran (Talk | contribs) (15,243 bytes)
- (cur | prev) 19:27, 8 October 2011 Ran (Talk | contribs) (15,242 bytes)
- (cur | prev) 19:26, 8 October 2011 Ran (Talk | contribs) (15,242 bytes)
- (cur | prev) 19:24, 8 October 2011 Ran (Talk | contribs) (15,244 bytes)
- (cur | prev) 19:23, 8 October 2011 Ran (Talk | contribs) (15,244 bytes)
- (cur | prev) 19:23, 8 October 2011 Ran (Talk | contribs) (15,356 bytes)
- (cur | prev) 19:22, 8 October 2011 Ran (Talk | contribs) (15,243 bytes)
- (cur | prev) 19:21, 8 October 2011 Ran (Talk | contribs) (15,638 bytes)
- (cur | prev) 19:02, 8 October 2011 Ran (Talk | contribs) (15,244 bytes)
- (cur | prev) 19:02, 8 October 2011 Ran (Talk | contribs) (15,242 bytes)
- (cur | prev) 19:01, 8 October 2011 Ran (Talk | contribs) (15,242 bytes)
- (cur | prev) 18:58, 8 October 2011 Ran (Talk | contribs) (15,360 bytes)
- (cur | prev) 18:58, 8 October 2011 Ran (Talk | contribs) (15,417 bytes)
- (cur | prev) 18:57, 8 October 2011 Ran (Talk | contribs) (15,417 bytes)
- (cur | prev) 23:08, 7 October 2011 Ran (Talk | contribs) (15,211 bytes)
- (cur | prev) 09:47, 12 September 2011 Ran (Talk | contribs) (13,514 bytes)
- (cur | prev) 23:43, 12 November 2010 Ran (Talk | contribs) (13,105 bytes) (*Undo revision 20641 by Ran (talk)*)

FIG. 86: Example of CSS history⁵ in MediaWiki

⁵<http://awoiaf.westeros.org/index.php?title=MediaWiki:Common.css&action=history>

Revision history of "MediaWiki:Common.js"

[View logs for this page](#)

Browse history

From year (and earlier): From month (and earlier): Deleted only

Diff selection: mark the radio boxes of the revisions to compare and hit enter or the button at the bottom.
 Legend: **(cur)** = difference with latest revision, **(prev)** = difference with preceding revision, **m** = minor edit.

Compare selected revisions

- (cur | prev) 16:20, 29 September 2011 Ran (Talk | contribs) (4,503 bytes)
- (cur | prev) 16:17, 29 September 2011 Ran (Talk | contribs) (4,933 bytes)
- (cur | prev) 16:11, 29 September 2011 Ran (Talk | contribs) (4,549 bytes)
- (cur | prev) 15:58, 29 September 2011 Ran (Talk | contribs) (4,536 bytes)
- (cur | prev) 09:48, 12 September 2011 Ran (Talk | contribs) (4,505 bytes) *(Created page with "- →Any JavaScript here will be loaded for all users on every page load.: /** Collapsible tables * * Description: A...")*

FIG. 87: Example of JavaScript history in MediaWiki⁶

Figure 88 shows an example where the CSS matters. The previous version of this page is using the current CSS, which does not render the same way. As a result, the shield image appears over the text on the left side of the page.

Unfortunately, we could find no hooks that allowed the MediaWiki Extension to access these resources and change how the page is rendered. This is an item that will require us to work with the MediaWiki Development team.

Once this is achieved, it could be made an optional setting. Some sites may not want their present content displayed with previous styles or JavaScript code.

8.4 SUMMARY

In this chapter, we discussed the Memento MediaWiki Extension, a server-side solution to the spoiler problem.

We have also experimented with the use of Memento Pattern 1.1 in an attempt to improve performance, and have found that it would actually have a negative impact on performance, due to idiosyncrasies in how it would need to be implemented within MediaWiki. Thus, the Memento MediaWiki extension uses Memento Pattern 2.1 as described in the rest of this thesis.

⁶<http://awoiaf.westeros.org/index.php?title=MediaWiki:Common.js&action=history>

Revision as of 14:39, 27 May 2012 by [Dimadick](#) (Talk | contribs)
 (diff) | [Older revision](#) | [Latest revision](#) (diff) | [Newer revision](#) → (diff) Log in

[Template](#) [Discussion](#) [Read](#) [View source](#) [View history](#)

Content is available under [Creative Commons Attribution Share Alike](#).

[Privacy policy](#) [About A Wiki of Ice and Fire](#) [Disclaimers](#)




Kings of Westeros [hide]

<p>Navigation</p> <ul style="list-style-type: none"> Main page Community portal Current events Recent changes Random page Help Targaryens Toolbox What links here Related changes Special pages Printable version Permanent link Page information 	<pre> </div> </td></tr><tr style="height:2px;"><td></td></tr><tr><td class="navbox-group" style="width:100px; background-color:#ddd; white-space:nowrap; text-align: right;"><td style="text-align:left;border-left-width:2px;border-left-style:solid;width:100%;padding:0px;" colspan="1" class="navbox-list">Moat Cailin · Deepwood Motte · Stony Shore · Torrhen's Square · Winterfell (Theon) · Sack of Winterfell (Bolton) · Retaking of Torrhen's Square · Shield Islands</td></tr><tr style="height:2px;"><td></td></tr><tr><td class="navbox-group" style="width:100px; background-color:#ddd; white-space:nowrap; text-align: right;"><td style="text-align:left;border-left-width:2px;border-left-style:solid;width:100%;padding:0px;" colspan="1" class="navbox-list">Storm's End · Battle of the Blackwater · Battle of Castle Black · Siege of Dragonstone · Siege of Winterfell</td></tr></table> </pre>
--	--

Holders of the iron Throne
 Aegon I (1-37) · Aerys I (37-42) · Maegor I (42-48) · Jaehaerys I (48-103) · Viserys I (103-129) · Aegon II (129-131) · Aegon III (131-157) · Daeron I (157-161) · Baelor I (161-171) · Viserys II (171-172) · Aegon IV (172-184) · Daeron II (184-209) · Aerys I (209-221) · Maekar I (221-233) · Aegon V (233-259) · Jaehaerys II (259-262).

After Robb's Crowning
 Oxcross · Ashemark · Crag · The Fords · Sack of Harrenhal · Duskenale · Ruby Ford · Harrenhal 2 · Red Wedding · Siege of Riverrun · Siege of Raventree

FIG. 88: Example of the current CSS not agreeing with an previous revision of a MediaWiki page

We have also shown how merely installing the Memento MediaWiki Extension has a negligible impact on performance for accessing MediaWiki pages, both current and oldid.

Unfortunately, until work is done with the MediaWiki development team to address embedded stylesheets and JavaScript, temporal coherence cannot be fully achieved. Even though we have not achieved full temporal coherence, we believe we have addressed the issue of spoilers. Because JavaScript is mostly used for producing browser based effects and software functionality, it is unlikely to produce spoilers. Also, CSS is used to control the rendering of web pages, not their content.

CHAPTER 9

FUTURE WORK

Our solution provides relief from spoilers for fan wikis, but there are still other directions to take the discoveries from this research.

Research can be done on how to avoid spoilers for situations where entire series are released at once. Now that we have identified a special case where Memento is not the complete solution for avoiding spoilers in this case, what additional tools can be used?

For resources without access to all page revisions, such as sparse web archives, one can compare past and future mementos to determine if spoilers have been revealed. This may require an actual review of the returned content to see if it contains spoilers, but other factors, such as the amount of text different between mementos, may be the key to providing some measure of this capability for sparse web archives.

It is possible to use our model not just for avoiding spoilers in television shows, but potentially sporting events and news articles. Web archives are known to have abundant mementos for news sites, meaning that minpast can conceivably be used to avoid spoilers for sporting and current events. A study can be conducted as to how well this will work for this use case.

Wikipedia itself may be a fascinating topic of study for those trying to avoid information on emerging topics. Minpast can be used to discover historical data on topics with evolving information such as same sex marriage or the United States relations with Cuba. It is also possible to use minpast programmatically to extract information from wiki edits, determining not only when a specific event occurred, but what change happened.

One can look at the content of specific revisions to determine if specific semantic classes of revisions may contain spoilers. Metrics, such as length of content or number of revisions per hour, may be used to indicate that some revisions have a higher probability of containing spoilers. Alternatively, one can use tools such as Amazon's Mechanical Turk to determine the effectiveness of avoiding spoilers using these metrics.

There is the potential to create a system that would allow a user to select a television show and episode, then present them only with web pages from the time period prior to that episode. It would be a domain-specific front-end to the existing Memento infrastructure.

The digital preservation community may benefit from further study on additional TimeGate heuristics and their use cases, such as `minfuttr`, where one provides a datetime lower bound for the mementos returned. Using `minfuttr`, adventurous fans may be able to discover when spoilers were revealed on a given web page. It may also be possible to use `minfuttr` to find the first reported event after a given date.

One can use our wiki experiment on Wikipedia at large to determine the number of missed updates and redundant mementos, possibly arriving at a statistically significant sample that can be used to rate the effectiveness of web archiving.

CHAPTER 10

CONCLUSIONS

In this thesis we have examined the use of Memento to avoid spoilers. We have reviewed the work of others, discovering that, though work has been done to avoid spoilers in social media, little has been done for fan wikis. Memento provides an opportunity to view past versions of a web page, and wikis contain every revision of a page, allowing a user to visit a revision of a wiki page that was created prior to spoiler knowledge being released by episodic fiction.

We have examined different TimeGate heuristics used to return mementos to a user. We have discussed how the minpast heuristic would be best for avoiding spoilers, but is problematic when used in archives that are sparse. Because most web archives are sparse, the mindist heuristic is used instead. We indicated how the mindist heuristic can return spoilers because it finds the closest memento, and may return future versions of a page. When using mindist, we defined the term spoiler area to be that set of datetimes that will direct a user to a future revision, even though they chose a date in the past. We determined how one could calculate the probability of encountering a spoiler for a given page.

We then conducted a study on 16 fan wikis, contrasting their revision datetimes with Internet Archive memento datetimes and episode datetimes. We discovered that only 38% of the pages selected for our study actually existed in the Internet Archive. From this study, we showed that spoiler areas do exist and as pages get archived, they will produce spoilers. We also calculated a mean 0.66 probability of encountering a spoiler across all wikis in the study. Our study provided a unique opportunity to highlight the problem of missed updates, where some updates to pages are not recorded by web archives. We were able to show that the problem of missed updates is getting better, for some of the pages in the study.

We also studied anonymized logs from the Wayback Machine, showing that users are indeed being redirected to newer versions of pages, even though they came from a date in the past. We found that 17% of requests from our logs end up in the future, indicating that users of the Wayback Machine are experiencing spoilers.

Identifying that this problem can be solved for wikis by using minpast, we demonstrated the development and analysis of the Memento MediaWiki Extension, which can be used to avoid spoilers in fan wikis. We showed how the extension has a negligible impact on performance. We also lightly touched on the topic of temporal coherence, showing that parts of MediaWiki can help with the problem, even though it is not structured in a way to completely solve it.

From these studies, we have shown that the spoiler problem can exist in fan wikis and is being experienced by Wayback Machine users. We have also produced a solution in the Memento MediaWiki Extension. With this solution, the 0.66 probability of encountering a spoiler can go to 0 and Memento users can be directed to the actual version of a wiki page that existed at the time they desired, avoiding spoilers.

REFERENCES

- [1] AINSWORTH, S., AND NELSON, M. L. Evaluating sliding and sticky target policies by measuring temporal drift in acyclic walks through a web archive. In *Proceedings of the international ACM/IEEE Joint Conference on Digital libraries (JCDL)* (July 2013), pp. 39 – 48. (Also available as arXiv:1309:5503).
- [2] AINSWORTH, S. G., ALSUM, A., SALAHELDEEN, H., WEIGLE, M. C., AND NELSON, M. L. How much of the web is archived? In *Proceedings of the international ACM/IEEE Joint Conference on Digital libraries (JCDL)* (New York, New York, USA, 2011), ACM Press, pp. 133–136.
- [3] AINSWORTH, S. G., AND NELSON, M. L. Evaluating sliding and sticky target policies by measuring temporal drift in acyclic walks through a web archive. In *Proceedings of the international ACM/IEEE Joint Conference on Digital libraries (JCDL)* (2013), pp. 39 – 48.
- [4] AINSWORTH, S. G., NELSON, M. L., AND VAN DE SOMPEL, H. A Framework for Evaluation of Composite Memento Temporal Coherence. Tech. Rep. arXiv:1402.0928, Old Dominion University, Feb. 2014.
- [5] ALMEIDA, R., MOZAFARI, B., AND CHO, J. On the Evolution of Wikipedia. In *International Conference on Weblogs and Social Media* (2007).
- [6] ALNOAMANY, Y., ALSUM, A., WEIGLE, M. C., AND NELSON, M. L. Who and what links to the Internet Archive. In *Proceedings of the Theory and Practice of Digital Libraries (TPDL2013)* (Sept. 2013), pp. 346 – 357.
- [7] ALNOAMANY, Y. A., WEIGLE, M. C., AND NELSON, M. L. Access patterns for robots and humans in web archives. In *Proceedings of the international ACM/IEEE Joint Conference on Digital libraries (JCDL)* (2013), pp. 339 – 348.
- [8] BERJON, R., FAULKNER, S., LEITHEAD, T., O’CONNOR, E., PFEIFFER, S., AND HICKSON, I. HTML 5: A vocabulary and associated APIs for HTML and XHTML. <http://www.w3.org/TR/html5/>, 2014.

- [9] BERNERS-LEE, T. Web Architecture: Generic Resources. <http://www.w3.org/DesignIssues/Generic.html>, 1996.
- [10] BERNERS-LEE, T. ISSUE-14: What is the range of the HTTP dereference function? <http://www.w3.org/2001/tag/group/track/issues/14>, March 2002.
- [11] BERNERS-LEE, T., BRAY, T., CONNOLLY, D., COTTON, P., FIELDING, R., JECKLE, M., LILLEY, C., MENDELSON, N., ORCHARD, D., WALSH, N., AND WILLIAMS, S. Architecture of the World Wide Web, Volume One. <http://www.w3.org/TR/webarch/>, 2004.
- [12] BERNERS-LEE, T., FIELDING, R., AND MASINTER, L. RFC 3986: Uniform Resource Identifier (URI): Generic Syntax. <http://www.ietf.org/rfc/rfc3986.txt>, January 2005. Accessed: 2013-10-05.
- [13] BEYOU, S. Extension:BackwardsTimeTravel. <http://www.mediawiki.org/wiki/Extension:BackwardsTimeTravel>, Feb. 2014. Accessed: 2014-05-03.
- [14] BOYD-GRABER, J., GLASGOW, K., AND ZAJAC, J. S. Spoiler Alert : Machine Learning Approaches to Detect Social Media Posts with Revelatory Information. In *Proceedings of the American Society for Information Science and Technology* (Montreal, Quebec, Canada, 2013), vol. 50, Wiley Online Library, pp. 1–9.
- [15] COHEN, N. Spoiler Alert: Whodunit? Wikipedia Will Tell You. http://www.nytimes.com/2010/09/18/business/media/18spoiler.html?_r=2&adxnnl=1&adxnnlx=1284931453-Cougj2fpRsBoD+tJX2gG5g&, Sept. 2003.
- [16] CROFT, W. B., METZLER, D., AND STROHMAN, T. *Search Engines: Information Retrieval in Practice*. Pearson Education, Boston, Massachusetts, USA, 2010.
- [17] CUSLIDGE, T., AND WEISS, J. Potter fans' new foe? The Web. <http://www.popmatters.com/article/potter-fans-new-foe-the-web/>, July 2007. Accessed: 2014-09-16.

- [18] DENHAM, J. Netflix releases House of Cards ‘Spoiler Foiler’ for Twitter users. <http://www.independent.co.uk/arts-entertainment/tv/news/netflix-releases-house-of-cards-spoiler-foiler-for-twitter-users-9136324.html>, February 2014.
- [19] EBERT, R. Critics have no right to play spoiler. <http://www.rogerebert.com/rogers-journal/critics-have-no-right-to-play-spoiler>, 2005.
- [20] ELLIOTT, S. A Big Splash for a Prohibition Drama. *The New York Times* (Aug. 2010). Accessed: 2014-10-17.
- [21] FIELDING, R., GETTYS, J., MOGUL, J., FRYSTYK, H., MASINTER, L., LEACH, P., AND BERNERS-LEE, T. Hypertext Transfer Protocol – HTTP/1.1. <http://www.ietf.org/rfc/rfc2616.txt>, June 1999. Accessed: 2013-10-05.
- [22] FIELDING, R., AND RESCHKE, J. RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. <http://tools.ietf.org/html/rfc7230>, 2014.
- [23] FIELDING, R., AND RESCHKE, J. RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. <http://tools.ietf.org/html/rfc7231>, 2014.
- [24] FIELDING, R., AND RESCHKE, J. RFC 7232: Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests. <http://tools.ietf.org/html/rfc7232>, 2014.
- [25] FIELDING, R. T., AND TAYLOR, R. N. Principled design of the modern web architecture. *ACM Transactions on Internet Technology* 2, 2 (May 2002), 115–150.
- [26] FORRESTER, J., WICKE, G., ANANIAN, C. S., BREAUULT, A., I LLOPIS, M. O., AND SASTRY, S. Extension:Parsoid. <http://www.mediawiki.org/wiki/Parsoid>, May 2014.
- [27] FOX, R. Turning back 10 billion (web) pages of time. *Communications of the ACM* 44, 12 (2001), 10.

- [28] FULMER, J. JoeBlog Siege Home. <http://www.joedog.org/siege-home/>, Jan. 2012. Access: 2014-02-05.
- [29] GAMMA, E., HELM, R., JOHNSON, R., AND VLISSIDES, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.
- [30] GARCÍA, E. A Wiki of Ice and Fire. http://awoiaf.westeros.org/index.php/Main_Page, 2014.
- [31] GILES, J. Internet encyclopaedias go head to head. *Nature* 438, 7070 (Dec. 2005), 900–1.
- [32] GOLBECK, J. The Twitter Mute Button: A Web Filtering Challenge. In *SIGCHI Conference on Human Factors in Computing Systems* (2012), pp. 2755–2758.
- [33] GROSS, B. D. Spoiler alert! Negotiating social media in the DVR age. <http://www.cnn.com/2014/02/25/tech/social-media/spoilers-social-media/index.html>, Feb. 2014.
- [34] HART, H. Spoiler Wars Heat Up as Lost Returns. *New York Times* (Jan. 2009).
- [35] Help:Redirects. <http://www.mediawiki.org/wiki/Help:Redirects>, Oct. 2014. Accessed: 2014-10-05.
- [36] HIBBERD, J. ‘Buffy’ writers sell Grimm’s Fairy Tales pilot to NBC. *Entertainment Weekly* (Jan. 2011). Accessed: 2014-10-17.
- [37] HOLTMAN, K., AND MUTZ, A. Transparent Content Negotiation in HTTP. <http://tools.ietf.org/pdf/rfc2295.pdf>, 1998.
- [38] HU, M., LIM, E.-P., SUN, A., LAUW, H. W., AND VUONG, B.-Q. Measuring article quality in wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management - CIKM '07* (April 2007), ACM Press, p. 243.
- [39] HULL, J. SPOILER ALERT! <http://joshbuddy.github.io/spoiler-alert/>.

- [40] JAWORSKI, M. How to stop the Internet from ruining ' Game of Thrones ' for you. <http://www.dailydot.com/technology/how-to-avoid-spoilers-online/>, Apr. 2014.
- [41] JENKINS, H. *Convergence Culture: Where Old And New Media Collide*. New York University Press, 2008.
- [42] JEON, S., KIM, S., AND YU, H. Don't Be Spoiled by Your Friends: Spoiler Detection in TV Program Tweets. In *Seventh International AAAI Conference on Weblogs and Social Media* (2013), pp. 681–684.
- [43] JOHNS, M. D. Two Screen Viewing and Social Relationships: Exploring the invisible backchannel of TV viewing. In *Cultural Attitudes Towards Technology and Communication 2012* (Murdoch University, Australia, 2012), no. 1982, Murdoch University, pp. 333–343.
- [44] JOHNSON, S. The Big Bang Theory Comes Under Fire For Walking Dead Spoiler. <http://comicbook.com/blog/2013/02/08/the-big-bang-theory-comes-under-fire-for-walking-dead-spoiler/>, 2013.
- [45] JONES, S. [wikitech-l] Memento Extension for MediaWiki: Quick question about hooks. <http://lists.wikimedia.org/pipermail/wikitech-l/2014-March/075018.html>, Mar. 2014. Accessed: 2014-05-03.
- [46] JONES, S. M. 2014-04-01: Yesterday's (Wiki) Page, Today's Image? <http://ws-dl.blogspot.com/2014/04/2014-04-01-yesterdays-wiki-page-todays.html>, April 2014.
- [47] JONES, S. M. 2014-04-17: TimeGate Design Options For MediaWiki. <http://ws-dl.blogspot.com/2014/04/2014-04-18-timegate-design-options-for.html>, April 2014.
- [48] JONES, S. M., NELSON, M. L., SHANKAR, H., AND VAN DE SOMPEL, H. Bringing Web Time Travel to MediaWiki: An Assessment of the Memento MediaWiki Extension. Tech. Rep. avXiv:1406.3, Old Dominion University, June 2014.

- [49] KELLY, M., NELSON, M. L., AND WEIGLE, M. C. Mink: Integrating the Live and Archived Web Viewing Experience Using Web Browsers and Memento. In *Proceedings of the international ACM/IEEE Joint Conference on Digital libraries (JCDDL)* (London, England, September 2014), pp. 469–470.
- [50] KURAPOV, A. xrate. <http://microformats.org/wiki/xrate>, 2012.
- [51] KUROSE, J., AND ROSS, K. *Computer Networking: A Top Down Approach*, 6th ed. Pearson, 2013.
- [52] LAM, S., AND RIEDL, J. The past, present, and future of Wikipedia. *Computer* 44, 3 (March 2011), 87–90.
- [53] LEAVER, T. Watching Battlestar Galactica in Australia and the Tyranny of Digital Distance. *Media International Australia, Incorporating Culture & Policy*, 126 (2008), 145–154.
- [54] LEAVITT, J. D., AND CHRISTENFELD, N. J. S. Story spoilers don't spoil stories. *Psychological science* 22, 9 (Sept. 2011), 1152–4.
- [55] LEUF, B., AND CUNNINGHAM, W. *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley Longman Publishing Co., Inc., Boston, Massachusetts, USA, 2001.
- [56] LUCASSEN, T., AND SCHRAAGEN, J. M. Trust in Wikipedia: How Users Trust Information from an Unknown Source. In *Proceedings of the 4th Workshop on Information Credibility* (New York, NY, USA, 2010), WICOW '10, ACM, pp. 19–26.
- [57] MANLEY, S. 100,000 wikis on Wikia. http://community.wikia.com/wiki/User_blog:Sarah_Manley/100,000_wikis_on_Wikia, 2010.
- [58] Manual:Coding conventions. http://www.mediawiki.org/wiki/Manual:Coding_conventions, Oct. 2013. Accessed: 2013-10-15.
- [59] Manual:Coding conventions/PHP. http://www.mediawiki.org/wiki/Manual:Coding_conventions/PHP, Nov. 2013. Accessed: 2013-10-15.

- [60] Manual:Hooks/ArticleViewHeader. <http://www.mediawiki.org/wiki/Manual:Hooks/ArticleViewHeader>, Dec. 2012. Accessed: 2013-10-05.
- [61] Manual:hooks/BeforeInitialize. <http://www.mediawiki.org/wiki/Manual:Hooks/BeforeInitialize>, December 2013.
- [62] Manual:Hooks/BeforeParserFetchtemplateAndtitle. <http://www.mediawiki.org/w/index.php?title=Manual:Hooks/BeforeParserFetchTemplateAndtitle&action=history>, Oct. 2011. Accessed: 2013-10-05.
- [63] Manual:Hooks/ImageBeforeProduceHTML. <http://www.mediawiki.org/wiki/Manual:Hooks/ImageBeforeProduceHTML>, Mar. 2008. Accessed: 2013-04-01.
- [64] Manual:Interface/JavaScript. <http://www.mediawiki.org/wiki/Manual:Interface/JavaScript>, Mar. 2014. Accessed: 2013-04-01.
- [65] Manual:Interface/Stylesheets. <http://www.mediawiki.org/wiki/Manual:Interface/Stylesheets>, Sept. 2013. Accessed: 2013-04-01.
- [66] Manual:Special pages. http://www.mediawiki.org/wiki/Manual:Special_pages, Oct. 2013. Accessed: 2013-10-05.
- [67] MASANÈS, J. *Web Archiving*. Springer Berlin Heidelberg, Berlin, 2006.
- [68] Mediawiki: File class reference. <https://doc.wikimedia.org/mediawiki-core/master/php/html/classFile.html#a04bc50490d762a33a13169b1495d3361>, May 2014.
- [69] MITTELL, J. Sites of participation: Wiki fandom and the case of Lostpedia. In *Transformative Works and Cultures* (September 2009).
- [70] MITTELL, J. Wikis and Participatory Fandom. *The Participatory Cultures Handbook* (2012), 35.
- [71] MURPHY, J., HASHIM, N. H., AND OCONNOR, P. Take Me Back: Validating the Wayback Machine. *Journal of Computer-Mediated Communication* 13, 1 (Oct. 2007), 60–75.

- [72] NELSON, M. L. Memento-Datetime is not Last-Modified. <http://ws-dl.blogspot.com/2010/11/2010-11-05-memento-datetime-is-not-last.html>, 2010.
- [73] NELSON, M. L. A Plan For Curating “Obsolete Data or Resources”. In *UNC/NSF Workshop “Curating for Quality: Ensuring Data Quality to Enable New Science”* (Arlington, VA, Sept. 2012), vol. abs/1209.2664.
- [74] NELSON, M. L. 2013-07-15: Wayback Machine Upgrades Memento Support. <http://ws-dl.blogspot.com/2013/07/2013-07-15-wayback-machine-upgrades.html>, Jul 2013.
- [75] NELSON, M. L. 2013-10-14: Right-Click to the Past – Memento for Chrome. <http://ws-dl.blogspot.com/2013/10/2013-10-14-right-click-to-past-memento.html>, Oct 2013.
- [76] NETFLIX. Netflix spoiler foiler. <http://www.spoilerfoiler.com>.
- [77] NETFLIX. Netflix spoiler foiler. <http://breakingbad.spoilerfoiler.com>, 2014.
- [78] NOTESS, G. R. The Wayback Machine: The Web’s Archive. *ONLINE* 26, 2 (2002), 59 – 61.
- [79] PEDIAPRESS - Home. <http://pediapress.com/>.
- [80] Performance guidelines. https://www.mediawiki.org/wiki/Performance_guidelines, May 2014.
- [81] PICKING CARROTS LLC. Spoiler Shield. <http://www.spoilershield.com>, October 2014.
- [82] POPITSCH, N., MOSSER, R., AND PHILIPP, W. Urobe: A prototype for wiki preservation. In *International Conference on Preservation of Digital Objects* (September 2010).
- [83] REED, S., AND LACALLE, M. What is the difference between All and New Documents? <https://webarchive.jira.com/wiki/pages/viewpage.action?pageId=86573220>, Oct. 2014.

- [84] REITER, R. On closed world data bases. In *Readings in Nonmonotonic Reasoning*, M. L. Ginsberg, Ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987, pp. 300–310.
- [85] RESCORLA, E. RFC 2818: HTTP Over TLS. <https://tools.ietf.org/html/rfc2818>, May 2000.
- [86] ROSENTHAL, D. S., AND REICH, V. Permanent web publishing. In *USENIX Annual Technical Conference, FREENIX Track* (2000), pp. 129–140.
- [87] SANDERSON, R. Bug 34778: Deploy extension Memento on Wikipedia sites. https://bugzilla.wikimedia.org/show_bug.cgi?id=34778, Sept. 2013. Accessed: 2014-05-03.
- [88] SCHENONE, L. F. Extension:TimeMachine. <http://www.mediawiki.org/wiki/Extension:TimeMachine>, Feb. 2014. Accessed: 2014-05-03.
- [89] SCHIRRA, S., SUN, H., AND BENTLEY, F. Together alone. In *SIGCHI* (2014), pp. 2441–2450.
- [90] SCHMITT, R., HAAS, V., NOTHMAN, J., NIKOLAEV, A. V., BEIGEL, J., AMSÜSS, C., ZIBAROV, Z., WARD, N., CIPU, A., MÜLLER, T., AND CIEŚLAK, M. Welcome to mwlib’s documentation – mwlib 0.15 documentation. <http://mwlib.readthedocs.org/en/latest/index.html>, Dec. 2011.
- [91] Security checklist for developers. http://www.mediawiki.org/wiki/Security_checklist_for_developers, June 2012. Accessed: 2013-06-28.
- [92] SEEDS, M. A. *Foundations of Astronomy*, 3 ed. Wadsworth Publishing Company, Boston, Massachusetts, USA, 1992.
- [93] SHANKAR, H. Memento Time Travel - Chrome Web Store. <https://chrome.google.com/webstore/detail/memento-time-travel/jgbfpjledahoajcppakbgilmojkaghgm?hl=en>, 2014.
- [94] STANHOPE, K. ABC Picks Up Crisis Management Pilot From *Grey’s* Creator Shonda Rhimes. *TV Guide* (Dec. 2010). Accessed: 2014-10-17.

- [95] STEINER, T., VAN HOOLAND, S., AND SUMMERS, E. MJ No More: Using Concurrent Wikipedia Edit Spikes with Social Network Plausibility Checks for Breaking News Detection. In *Proceedings of the 22Nd International Conference on World Wide Web Companion* (Republic and Canton of Geneva, Switzerland, 2013), WWW '13 Companion, International World Wide Web Conferences Steering Committee, pp. 791–794.
- [96] STEVENS, W. R. *TCP/IP Illustrated, Volume 1*. Addison Wesley, Boston, Massachusetts, USA, 1994.
- [97] STROMBERG, B. Chrome Web Store - Tumblr Savior. <http://bit.ly/tumblr-savior>, July 2014.
- [98] STUVEN, R. Chrome Web Store - Open Tweet Filter. <http://bit.ly/open-tweet-filter>, July 2014.
- [99] TOYODA, M., AND KITSUREGAWA, M. The History of Web Archiving. *Proceedings of the IEEE 100*, Special Centennial Issue (May 2012), 1441–1443.
- [100] TSANG, A. S. L., AND YAN, D. Reducing the Spoiler Effect in Experiential Consumption. *Advances in Consumer Research*, 36 (2009), 708–709.
- [101] VAN DE SOMPEL, H., NELSON, M. L., AND SANDERSON, R. RFC 7089: HTTP Framework for Time-Based Access to Resource States – Memento. <http://tools.ietf.org/html/rfc7089>, 2013.
- [102] VAN DE SOMPEL, H., NELSON, M. L., SANDERSON, R., BALAKIREVA, L., AINSWORTH, S., AND SHANKAR, H. Memento: Time travel for the web. Tech. Rep. arXiv:0911.1112, Los Alamos National Laboratories and Old Dominion University, 2009.
- [103] VAN DE SOMPEL, H., SANDERSON, R., NELSON, M., BALAKIREVA, L., SHANKAR, H., AND AINSWORTH, S. An HTTP-based versioning mechanism for linked data. In *Proceedings of Linked Data on the Web Workshop* (April 2010).
- [104] VILAR, S. Chrome Web Store - Facebook Posts Filter. <http://bit.ly/facebook-posts-filter>, July 2014.

- [105] VUONG, B.-Q., LIM, E.-P., SUN, A., LE, M.-T., LAUW, H. W., AND CHANG, K. On Ranking Controversies in Wikipedia: Models and Evaluation. In *Proceedings of the 2008 International Conference on Web Search and Data Mining* (New York, NY, USA, 2008), WSDM '08, ACM, pp. 171–182.
- [106] WIKIA. About - Wikis from Wikia - Join the best wiki communities. http://www.wikia.com/About_Us, 2014.
- [107] WIKIA. Downton Abbey Wiki. http://downtonabbey.wikia.com/wiki/Downton_Abbey_Wiki, 2014.
- [108] WIKIA. Once Upon A Time Wiki. http://onceuponatime.wikia.com/wiki/Once_Upon_a_Time_Wiki, 2014.
- [109] WIKIA. Scandal Wiki. http://scandal.wikia.com/wiki/Main_Page, 2014.
- [110] Wikimedia servers. http://meta.wikimedia.org/wiki/Wikimedia_servers, Feb. 2014. Accessed: 2014-04-07.
- [111] WIKIPEDIA. Wikipedia:Spoiler. <http://en.wikipedia.org/wiki/Wikipedia:Spoiler>, 2014.
- [112] Writing an extension for deployment. https://www.mediawiki.org/wiki/Writing_an_extension_for_deployment, Oct. 2013. Accessed: 2013-10-15.
- [113] YASPAN, A. *Essentials of Probability*. Prindle, Weber & Schmidt, Incorporated, Boston, Massachusetts, USA, 1968.

APPENDIX A

SPOILER AREA VISUALIZATIONS

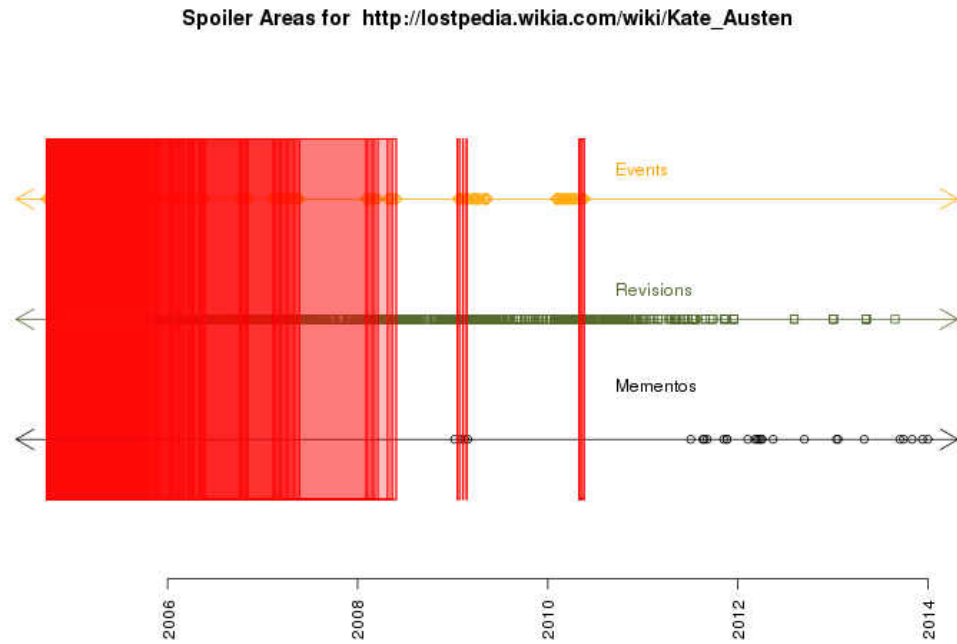


FIG. 89: Spoiler areas for the most popular page in *Lostpedia* (3,531 revisions)¹

¹http://lostpedia.wikia.com/wiki/Kate_Austen

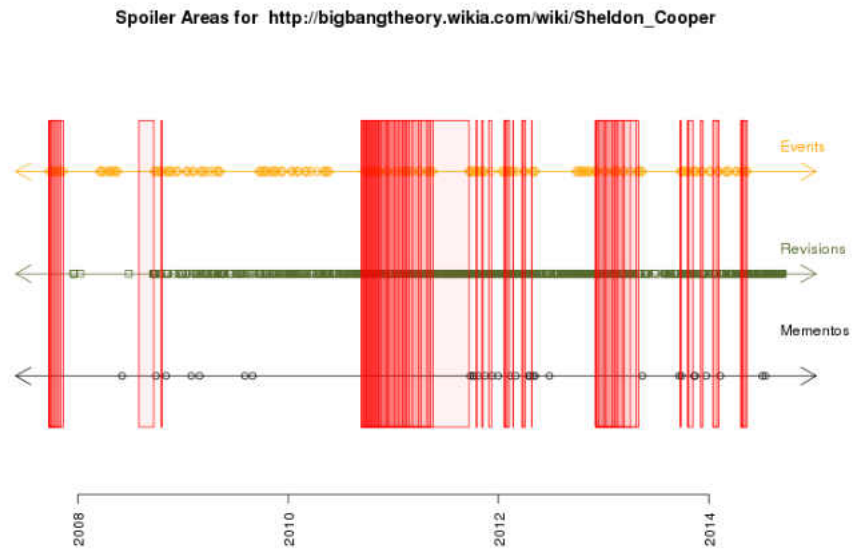


FIG. 90: Spoiler areas for the page in *the Big Bang Theory Wiki* that contains the most revisions²

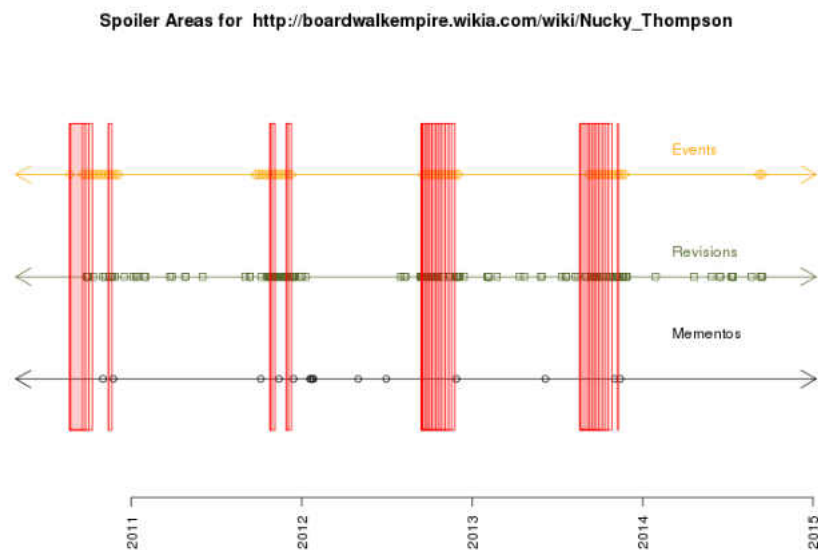


FIG. 91: Spoiler areas for the page in the *Boardwalk Empire Wiki* that contains the most revisions³

²http://bigbangtheory.wikia.com/wiki/Sheldon_Cooper

³http://boardwalkempire.wikia.com/wiki/Nucky_Thompson

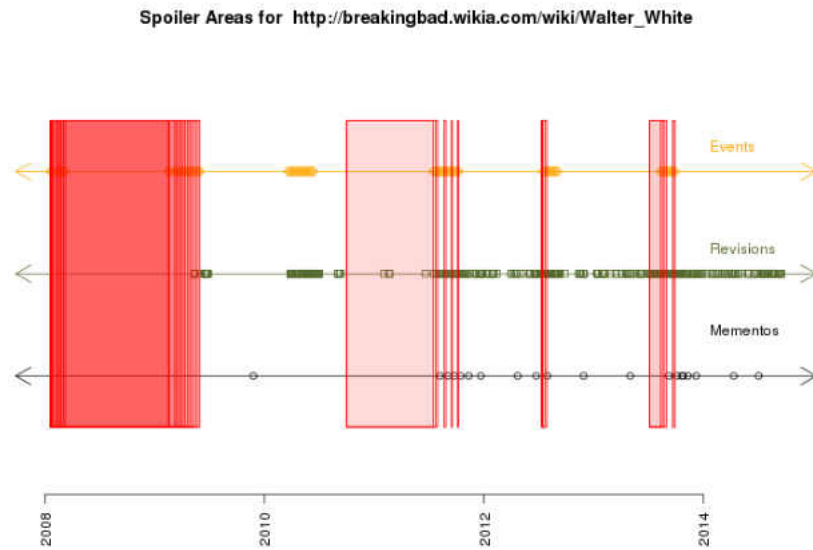


FIG. 92: Spoiler areas for the page in the *Breaking Bad Wiki* that contains the most revisions⁴

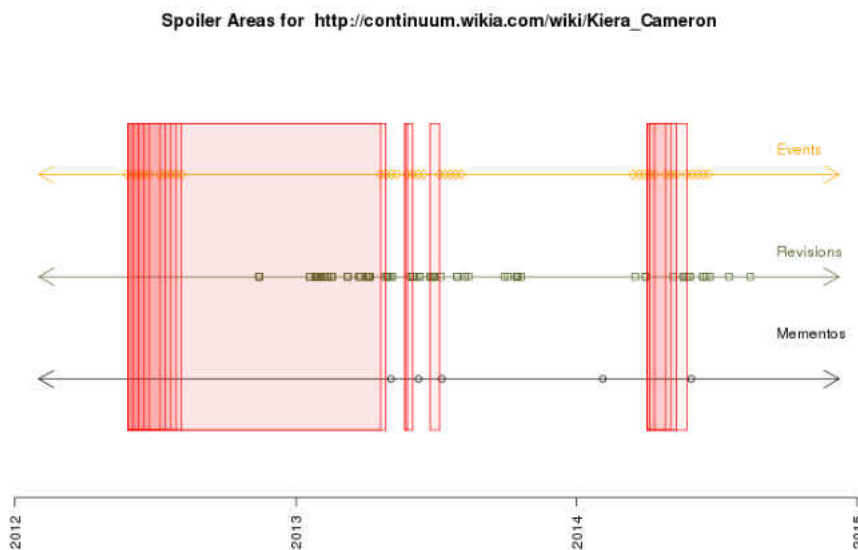


FIG. 93: Spoiler areas for the page in the *Continuum Wiki* that contains the most revisions⁵

⁴http://breakingbad.wikia.com/wiki/Walter_White

⁵http://continuum.wikia.com/wiki/Kiera_Cameron

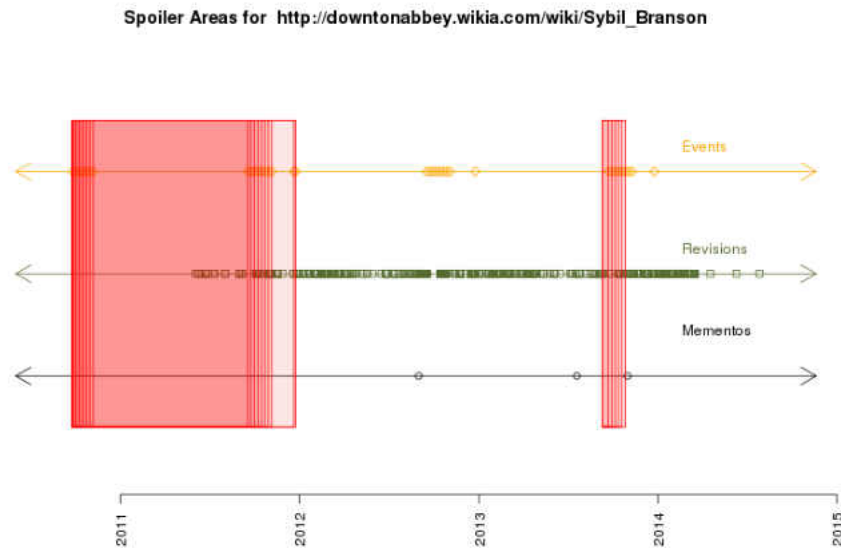


FIG. 94: Spoiler areas for the page in the *Downton Abbey Wiki* that contains the most revisions⁶

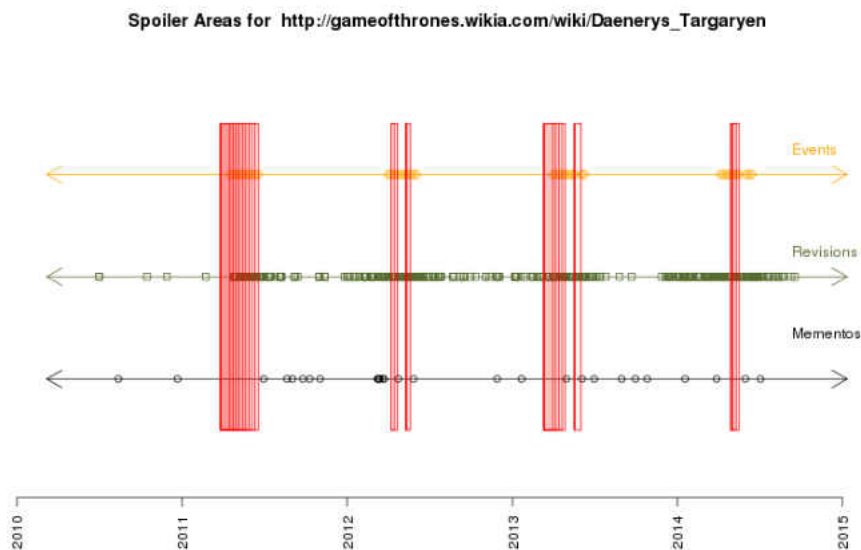


FIG. 95: Spoiler areas for the most popular page in the *Game of Thrones Wiki* (768 revisions)⁷

⁶http://downtonabbey.wikia.com/wiki/Sybil_Branson

⁷http://gameofthrones.wikia.com/wiki/Daenerys_Targaryen

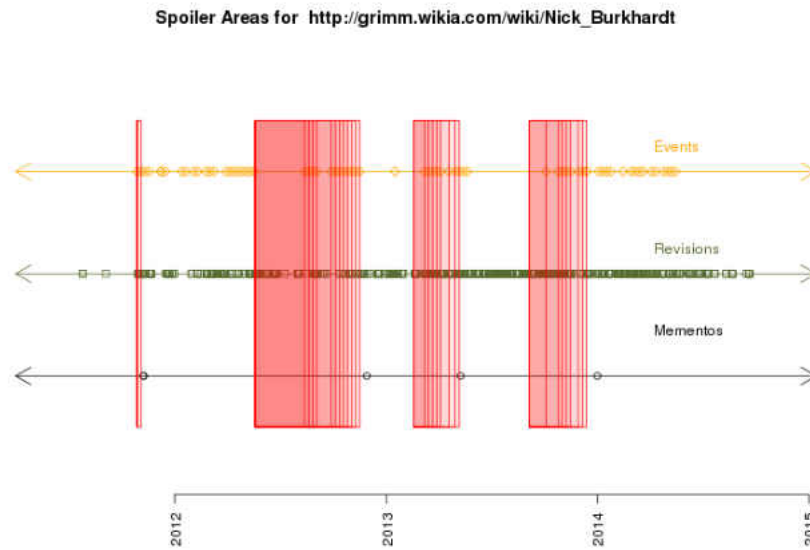


FIG. 96: Spoiler areas for the page in the *Grimm Wiki* that contains the most revisions⁸

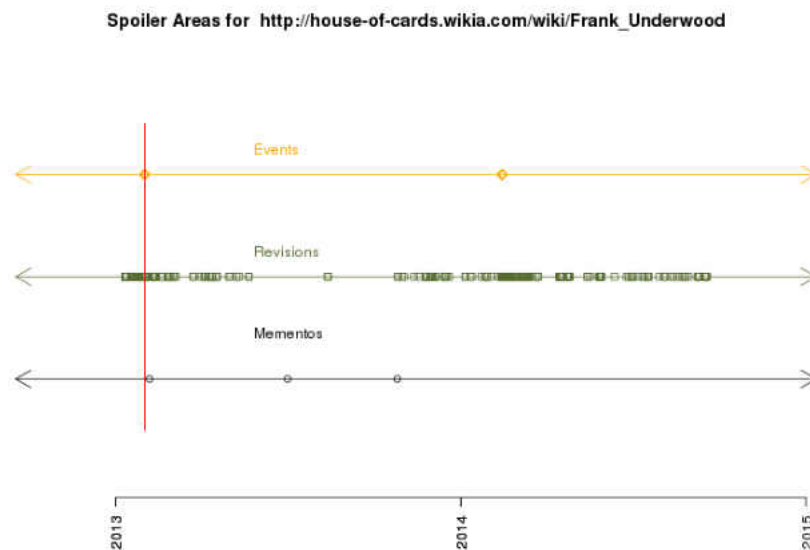


FIG. 97: Spoiler areas for the most popular page in the *House of Cards Wiki* (380 revisions)⁹

⁸http://grimm.wikia.com/wiki/Nick_Burkhardt

⁹http://house-of-cards.wikia.com/wiki/Frank_Underwood

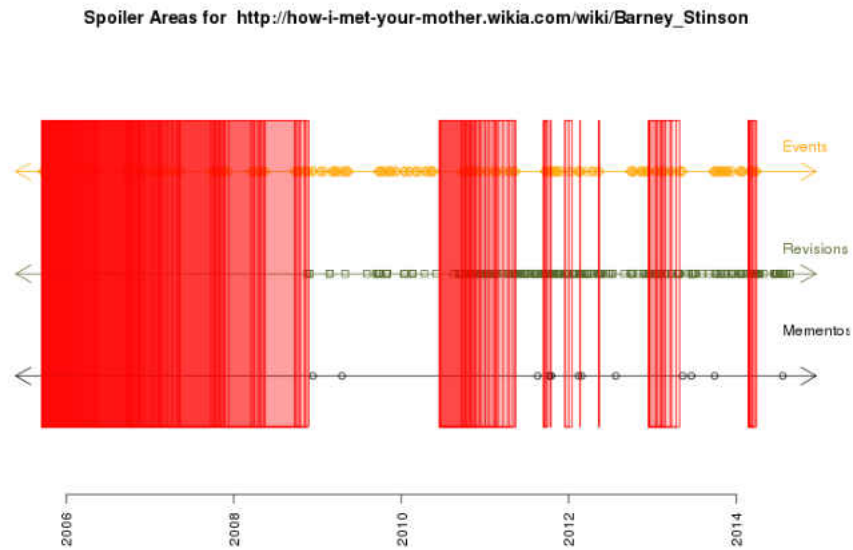


FIG. 98: Spoiler areas for the page in the *How I Met Your Mother Wiki* that contains the most revisions¹⁰

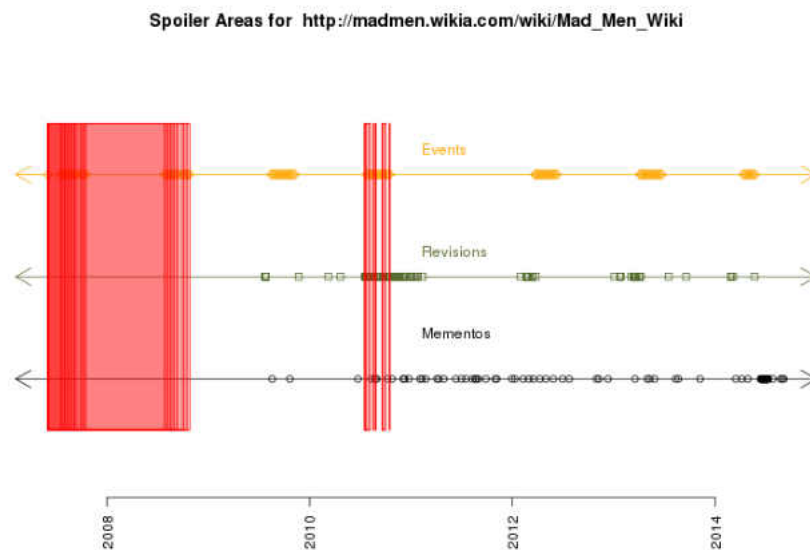


FIG. 99: Spoiler areas for the page in the *Mad Men Wiki* that contains the most revisions¹¹

¹⁰http://how-i-met-your-mother.wikia.com/wiki/Barney_Stinson

¹¹http://madmen.wikia.com/wiki/Mad_Men_Wiki

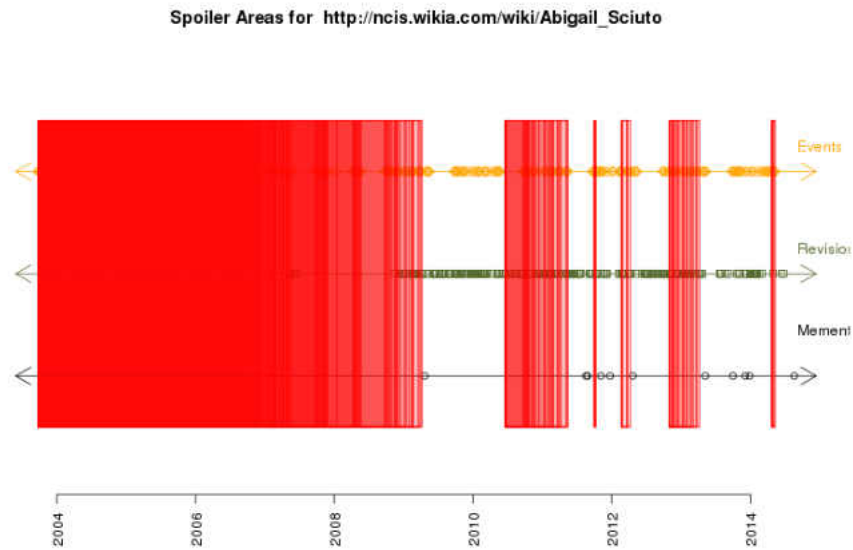


FIG. 100: Spoiler areas for the page in the *NCIS Database* that contains the most revisions¹²

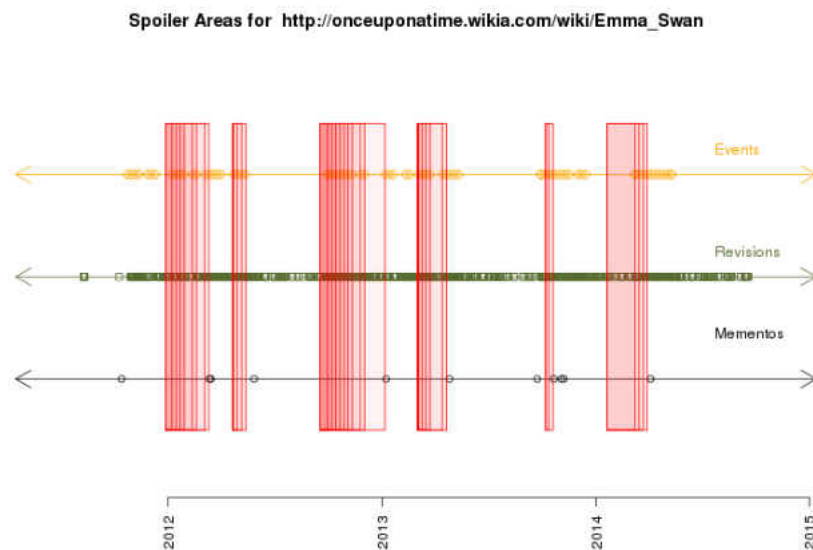


FIG. 101: Spoiler areas for the page in the *Once Upon A Time Wiki* that contains the most revisions¹³

¹²http://ncis.wikia.com/wiki/Abigail_Sciuto

¹³http://onceuponatime.wikia.com/wiki/Emma_Swan/Gallery

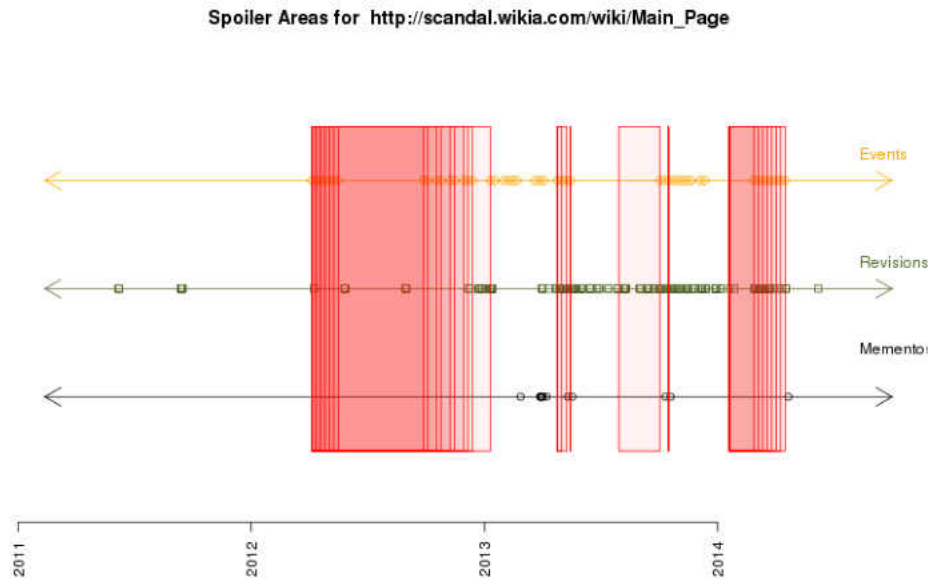


FIG. 102: Spoiler areas for the page in the *Scandal Wiki* that contains the most revisions¹⁴

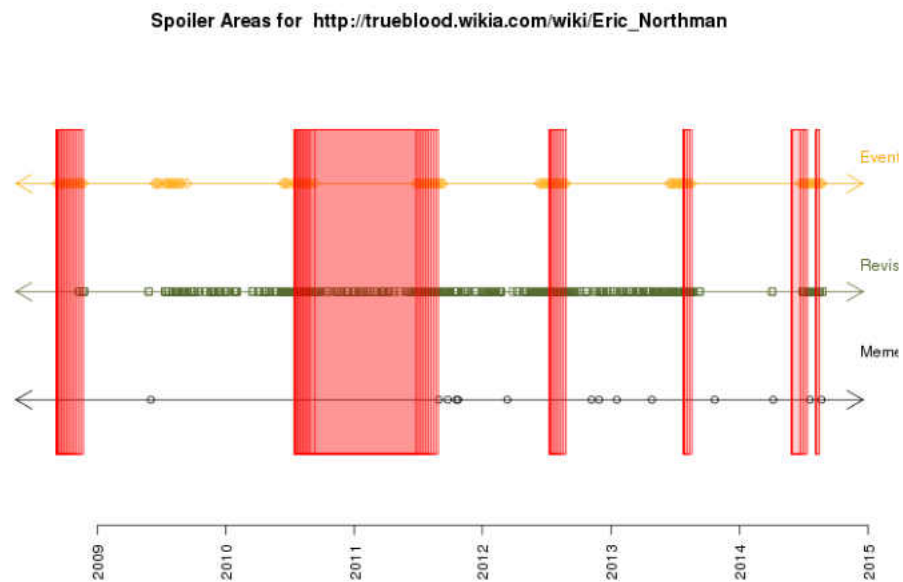


FIG. 103: Spoiler areas for the page in the *True Blood Wiki* that contains the most revisions¹⁵

¹⁴http://scandal.wikia.com/wiki/Main_Page

¹⁵http://trueblood.wikia.com/wiki/Eric_Northman

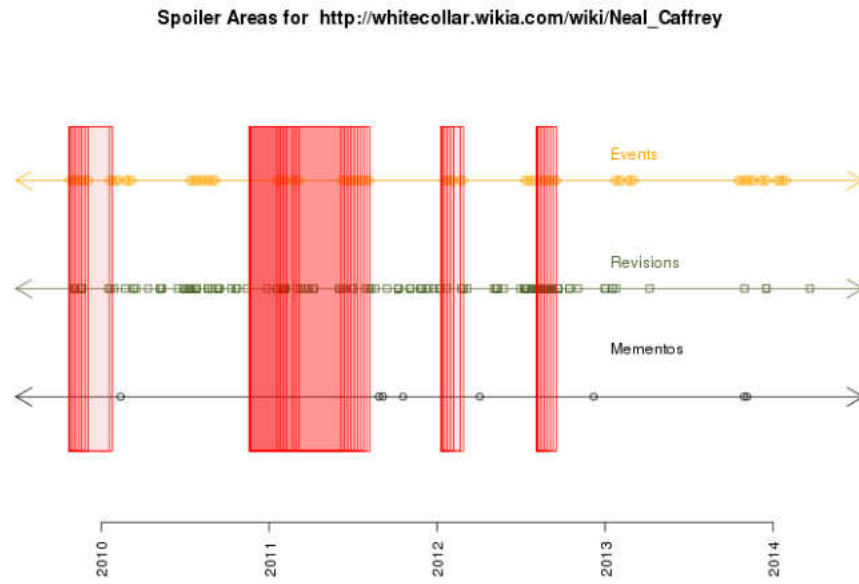


FIG. 104: Spoiler areas for the page in the *White Collar Wiki* that contains the most revisions¹⁶

¹⁶http://whitecollar.wikia.com/wiki/Neal_Caffrey

APPENDIX B

SPOILER PROBABILITY HISTOGRAMS

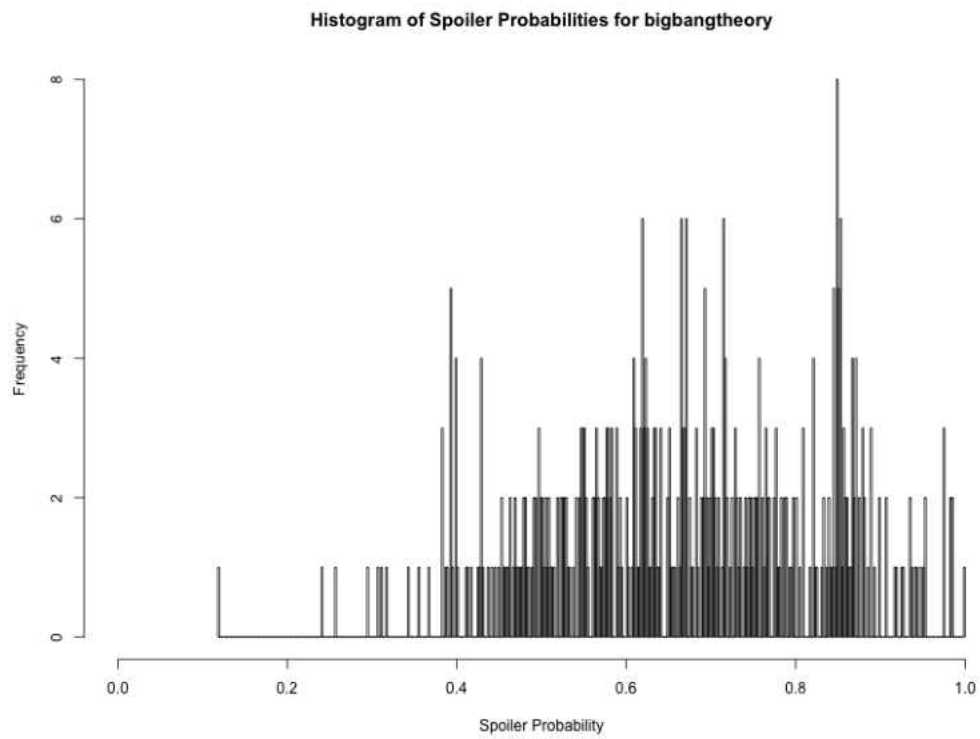


FIG. 105: Big Bang Theory Wiki

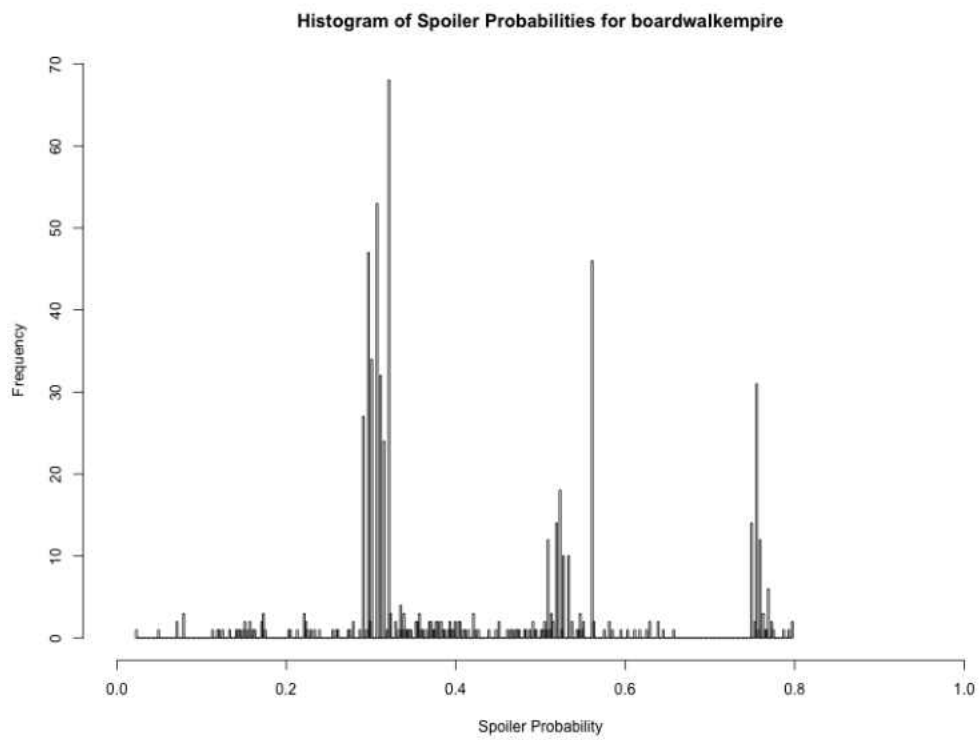


FIG. 106: Boardwalk Empire Wiki

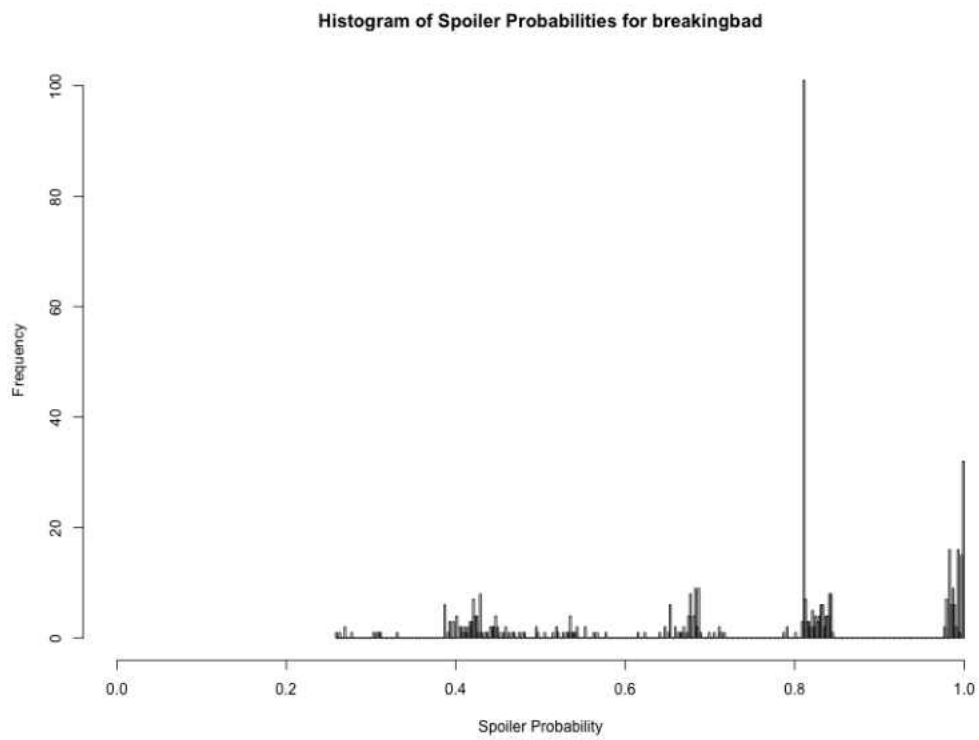


FIG. 107: Breaking Bad Wiki

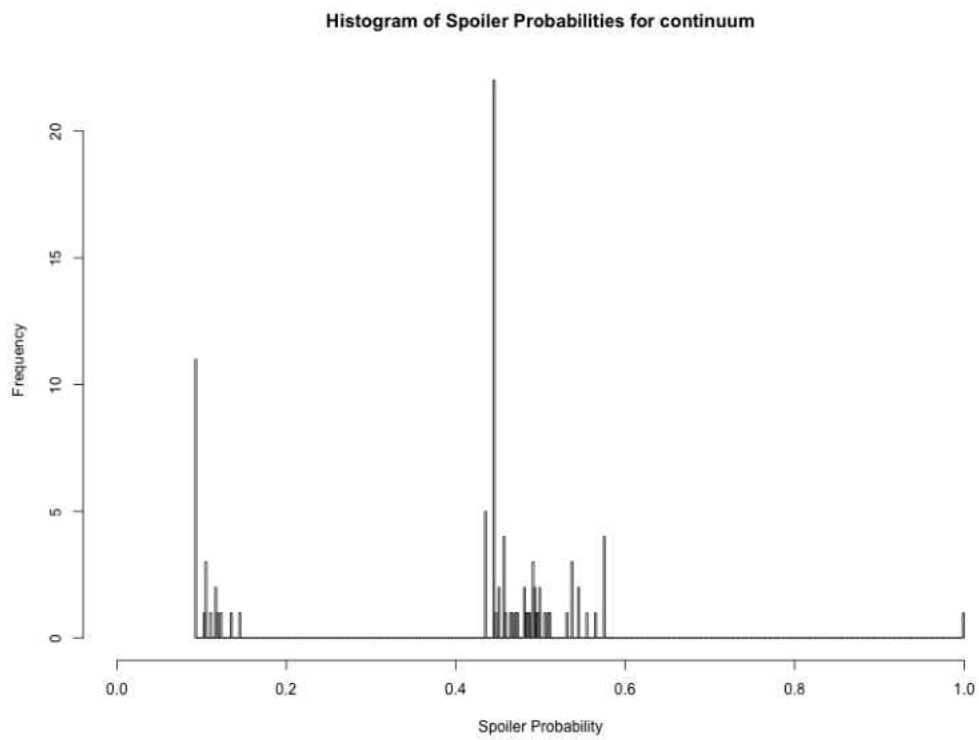


FIG. 108: Continuum Wiki

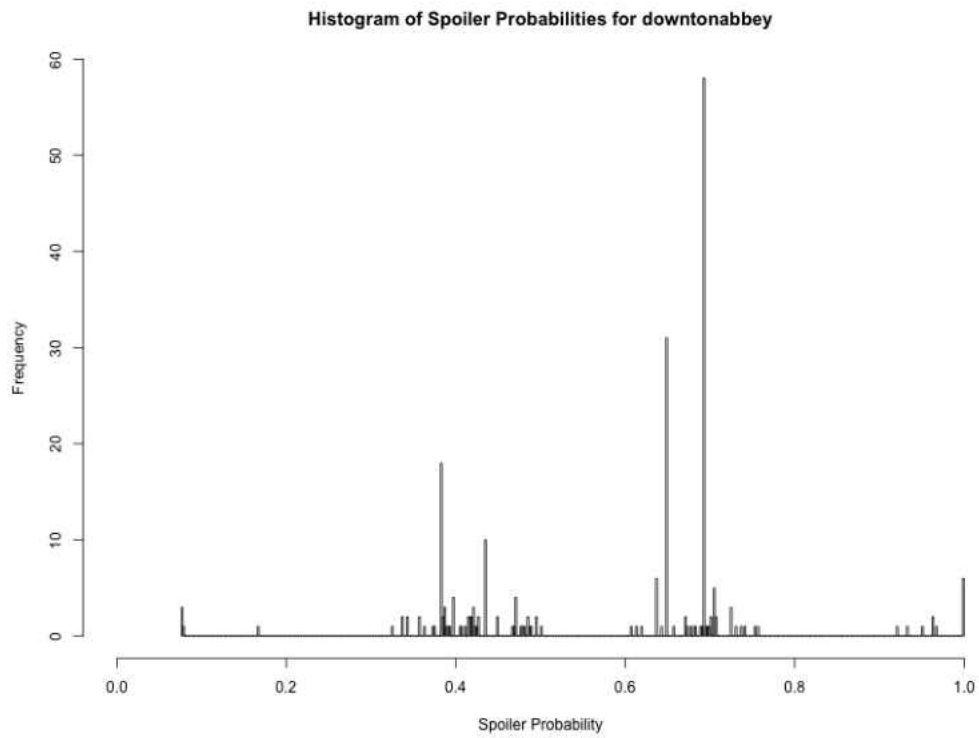


FIG. 109: Downton Abbey Wiki

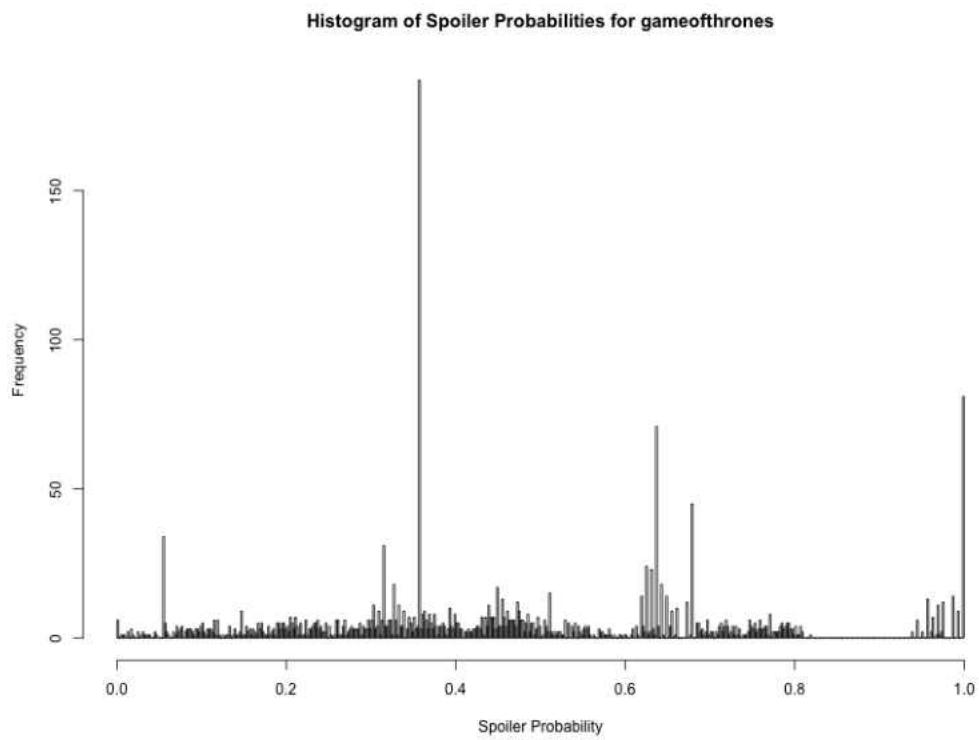


FIG. 110: Game of Thrones Wiki

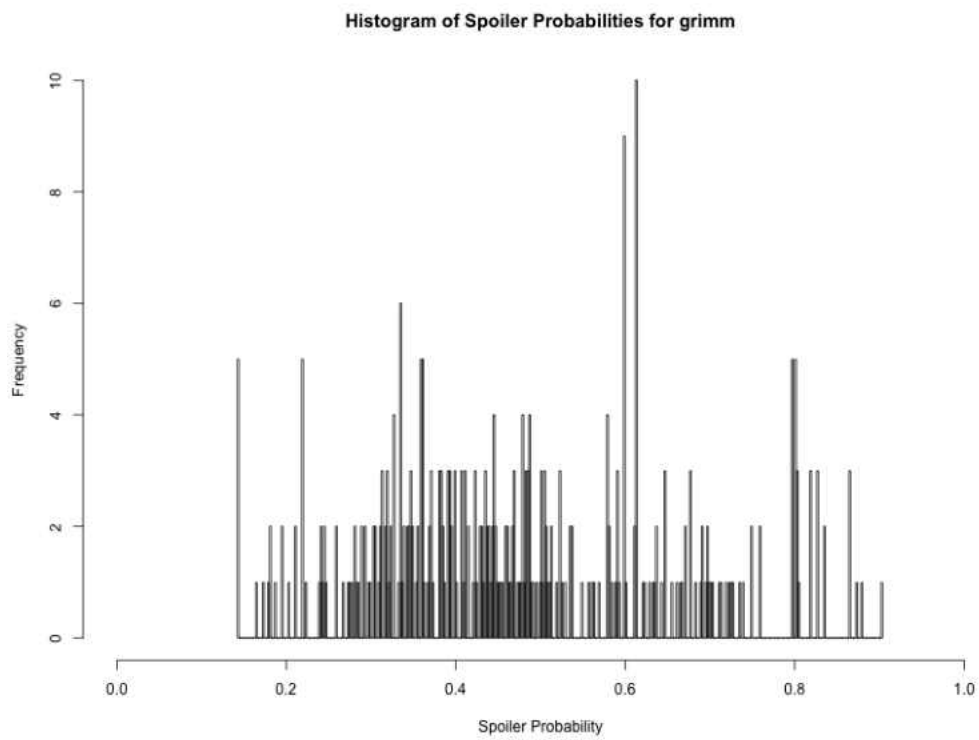


FIG. 111: Grimm Wiki

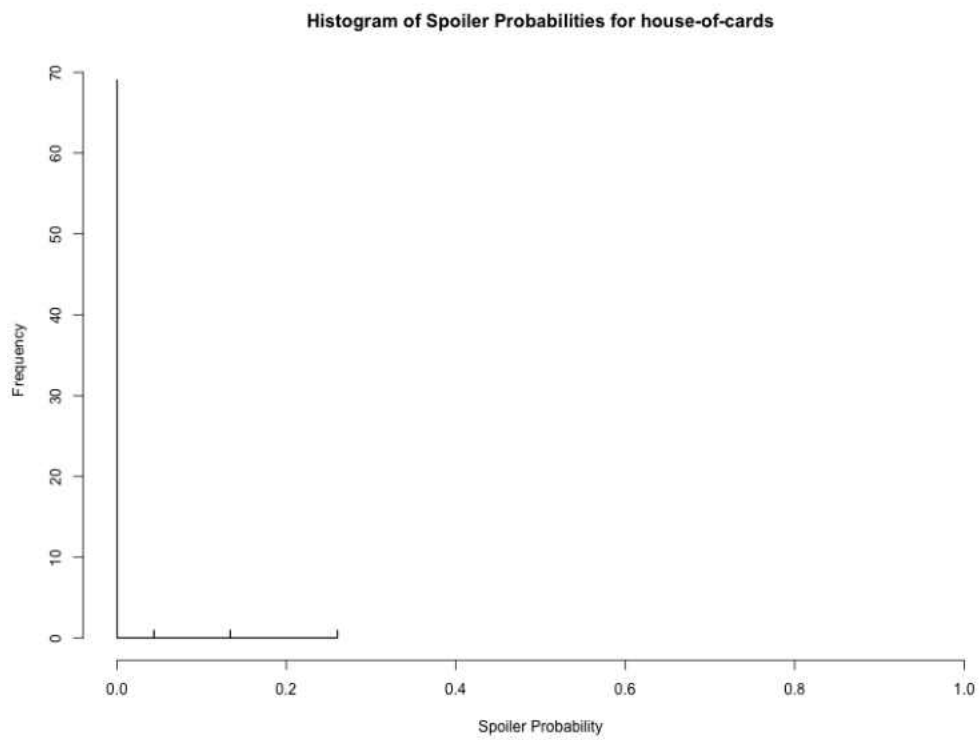


FIG. 112: House of Cards Wiki

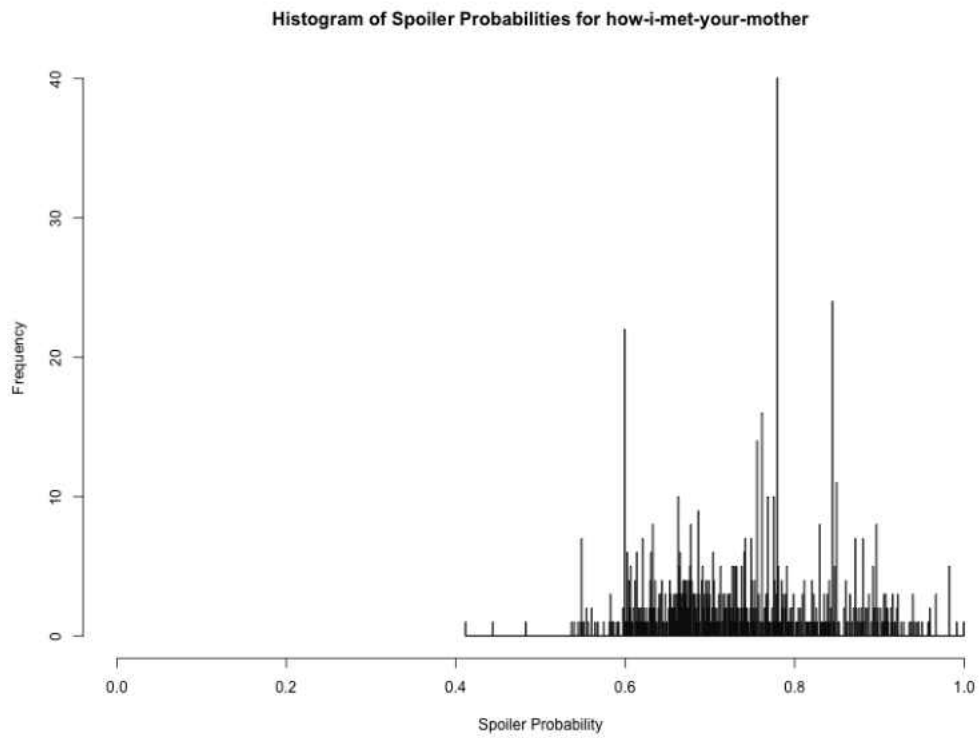


FIG. 113: How I Met Your Mother Wiki

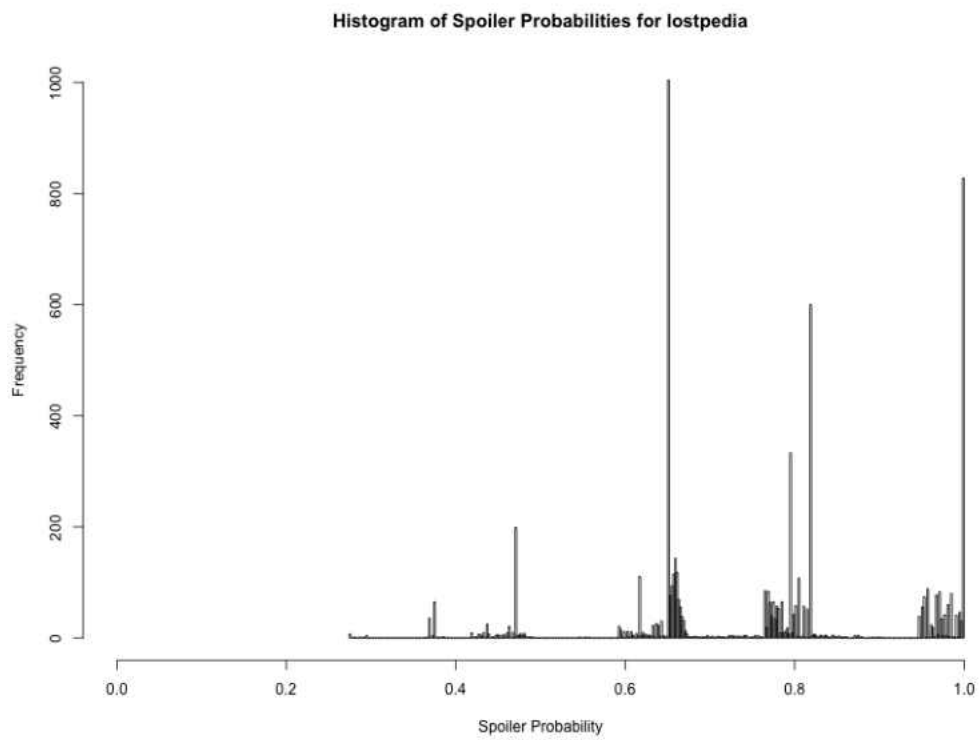


FIG. 114: Lostpedia

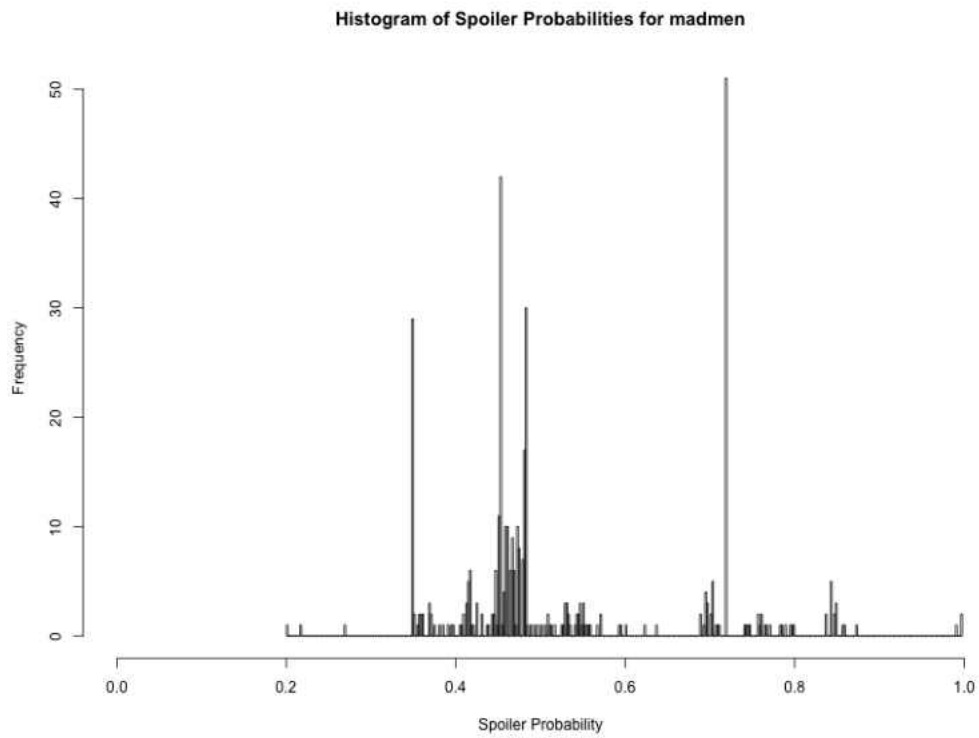


FIG. 115: Mad Men Wiki

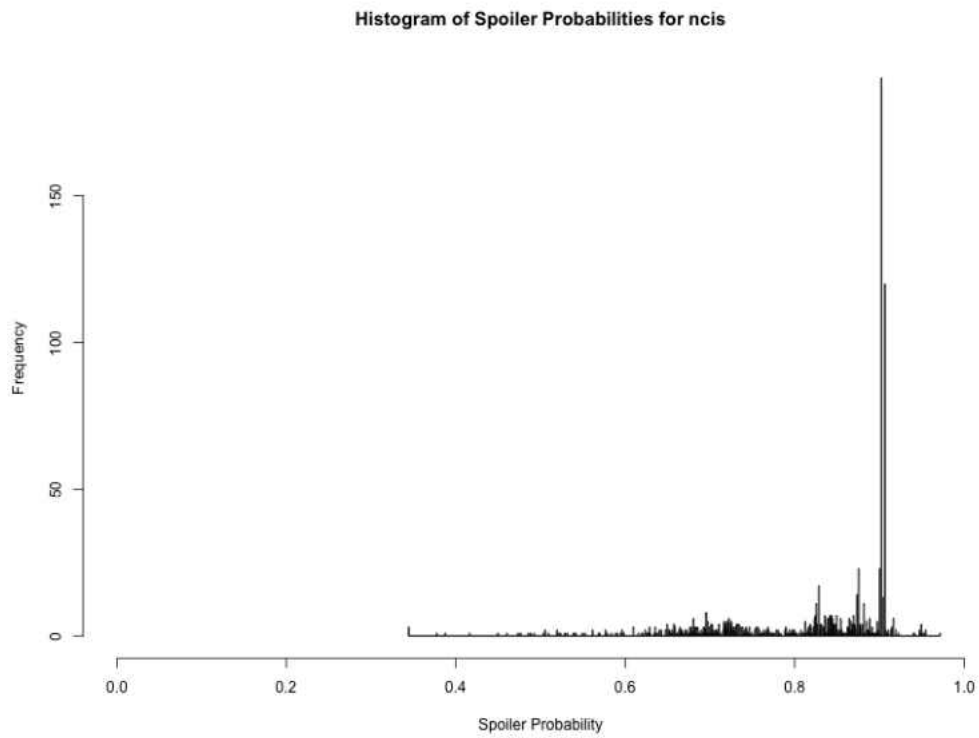


FIG. 116: NCIS Database

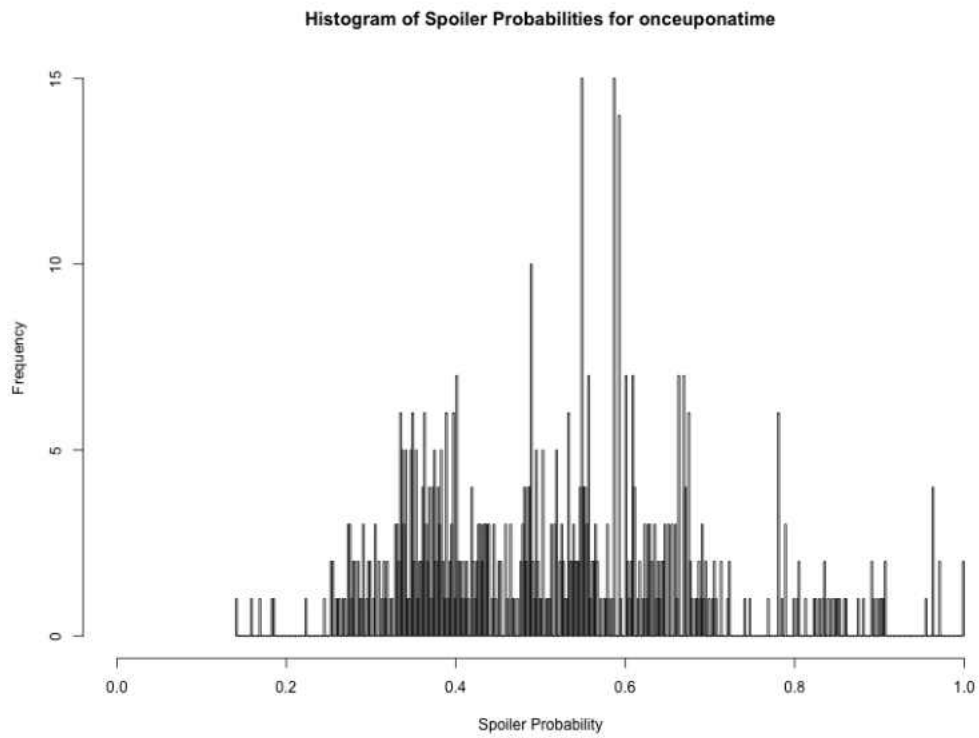


FIG. 117: Once Upon A Time Wiki

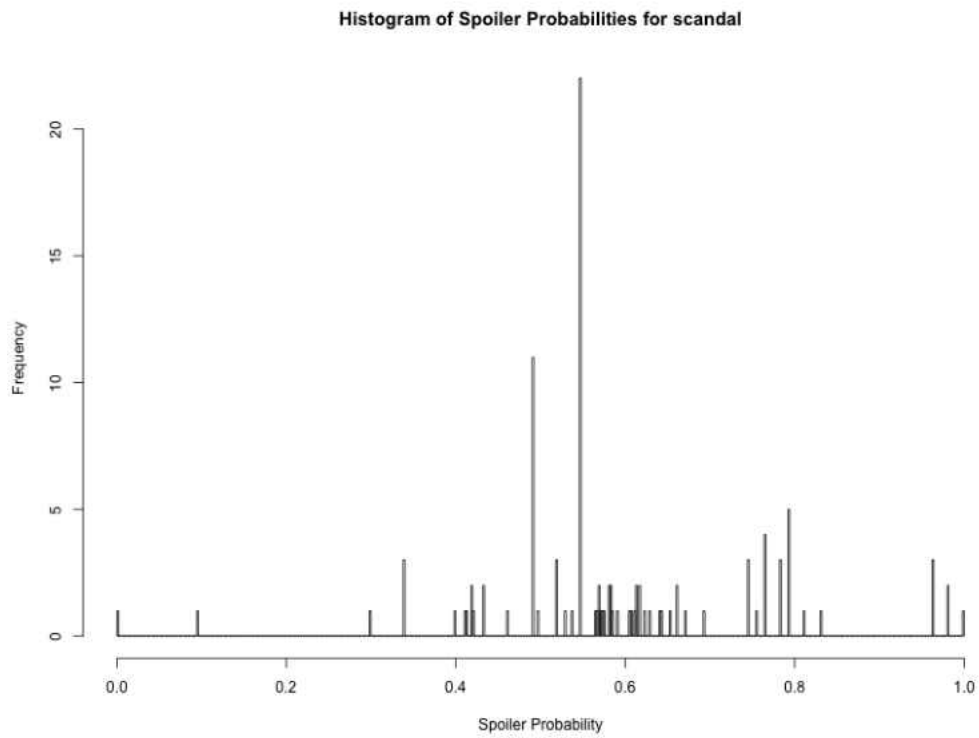


FIG. 118: Scandal Wiki

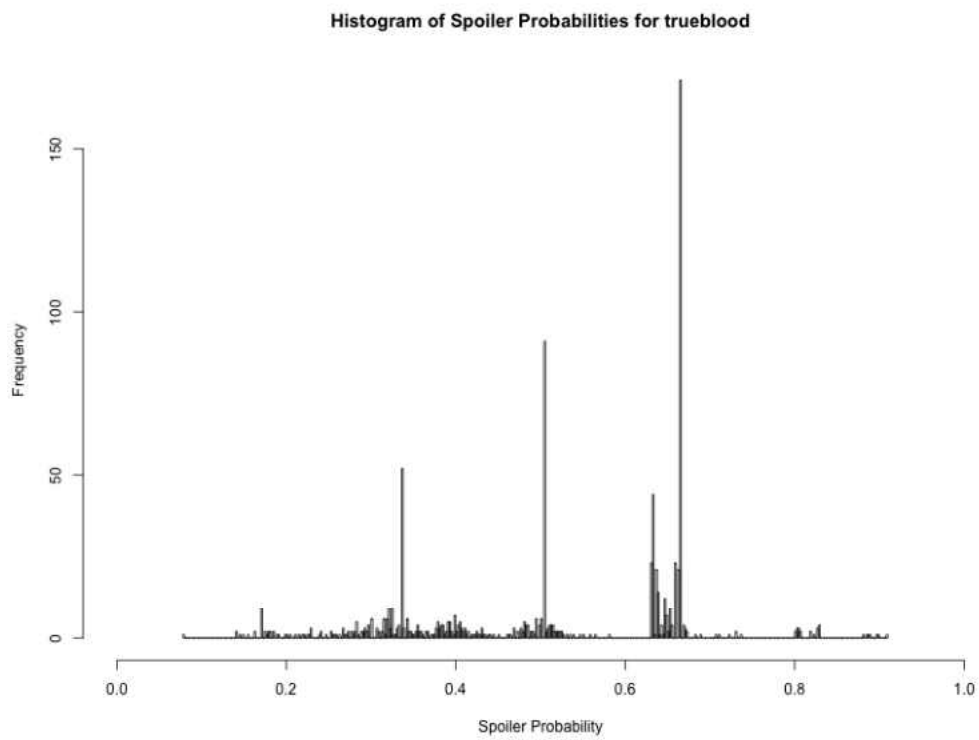


FIG. 119: True Blood Wiki

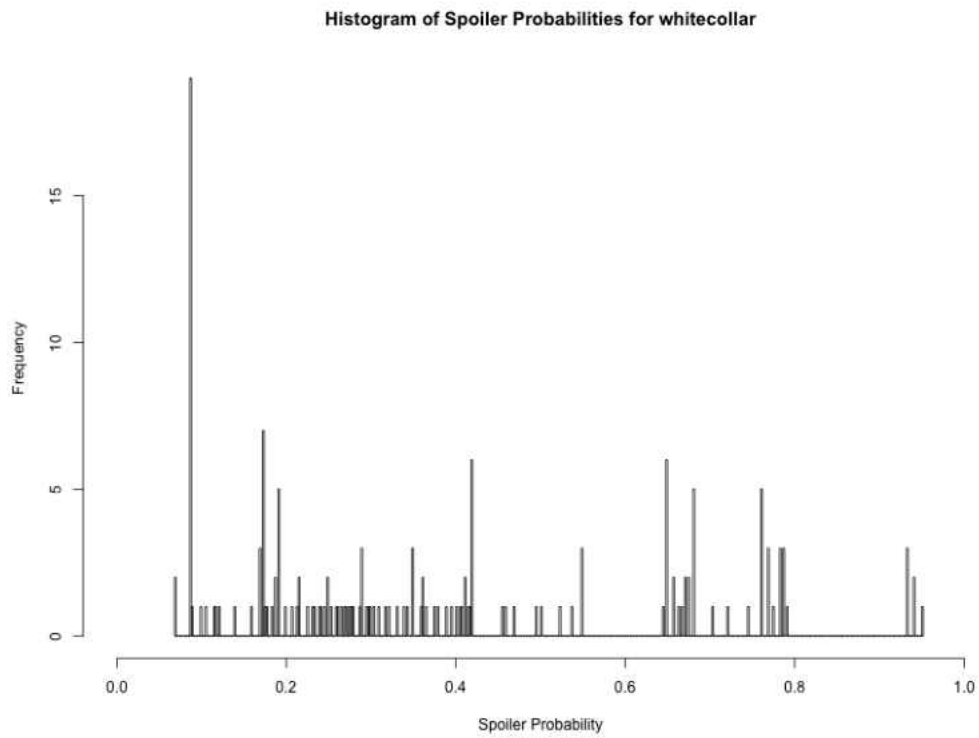


FIG. 120: White Collar Wiki

APPENDIX C

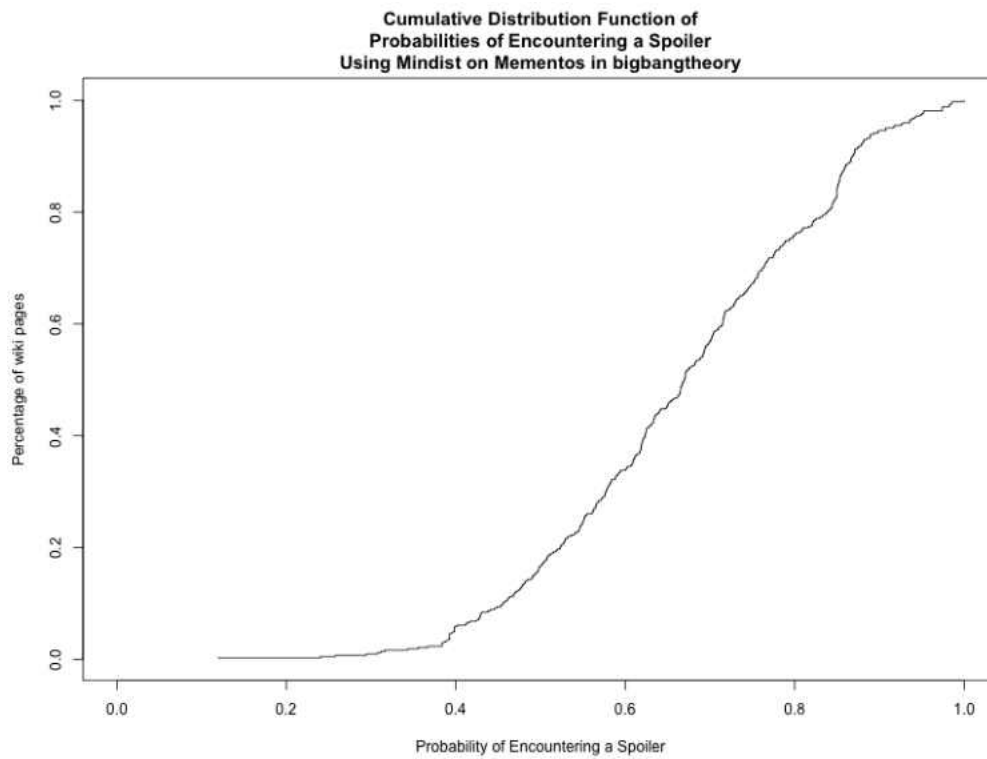
SPOILER PROBABILITY CUMULATIVE
DISTRIBUTION FUNCTION

FIG. 121: Big Bang Theory

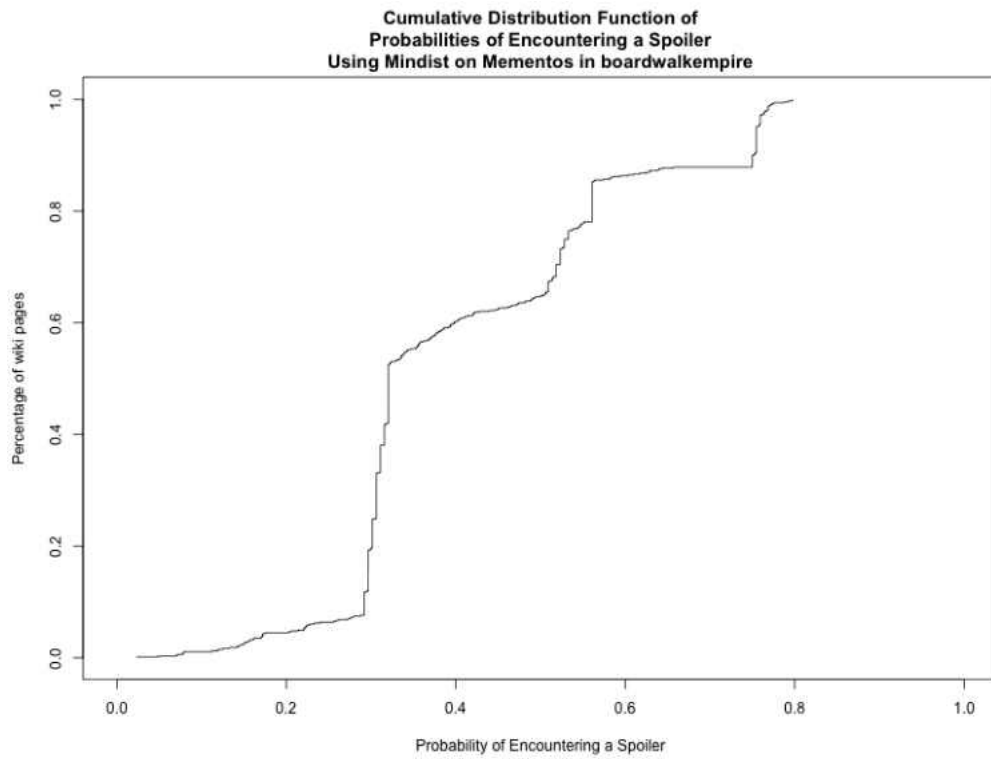


FIG. 122: Boardwalk Empire

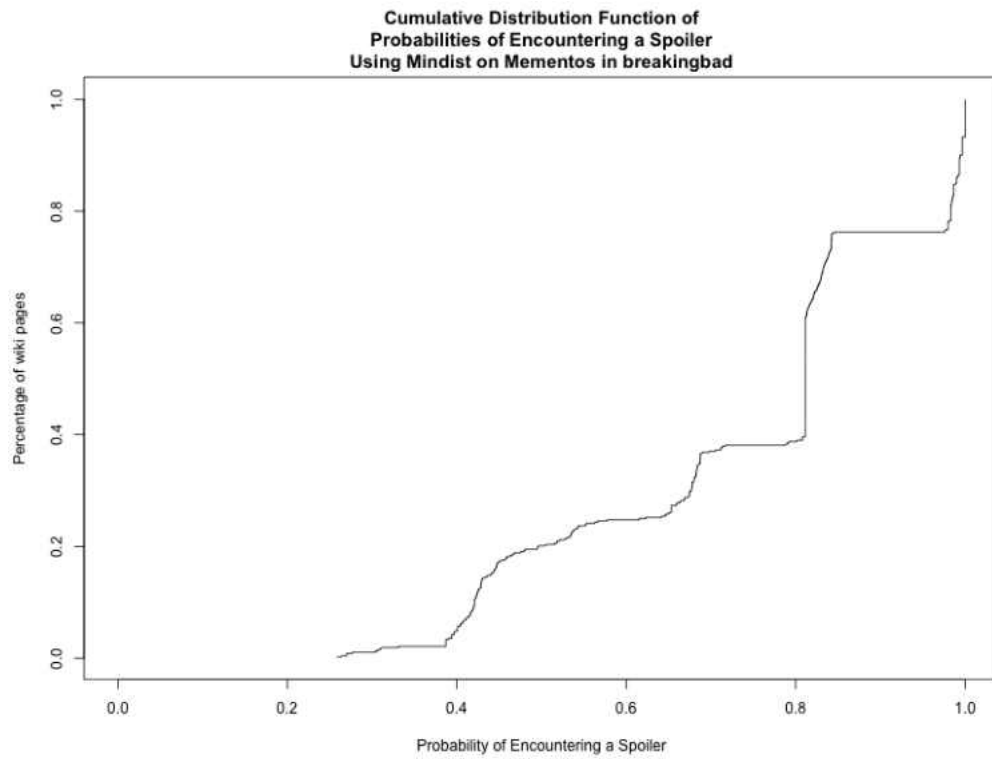


FIG. 123: Breaking Bad

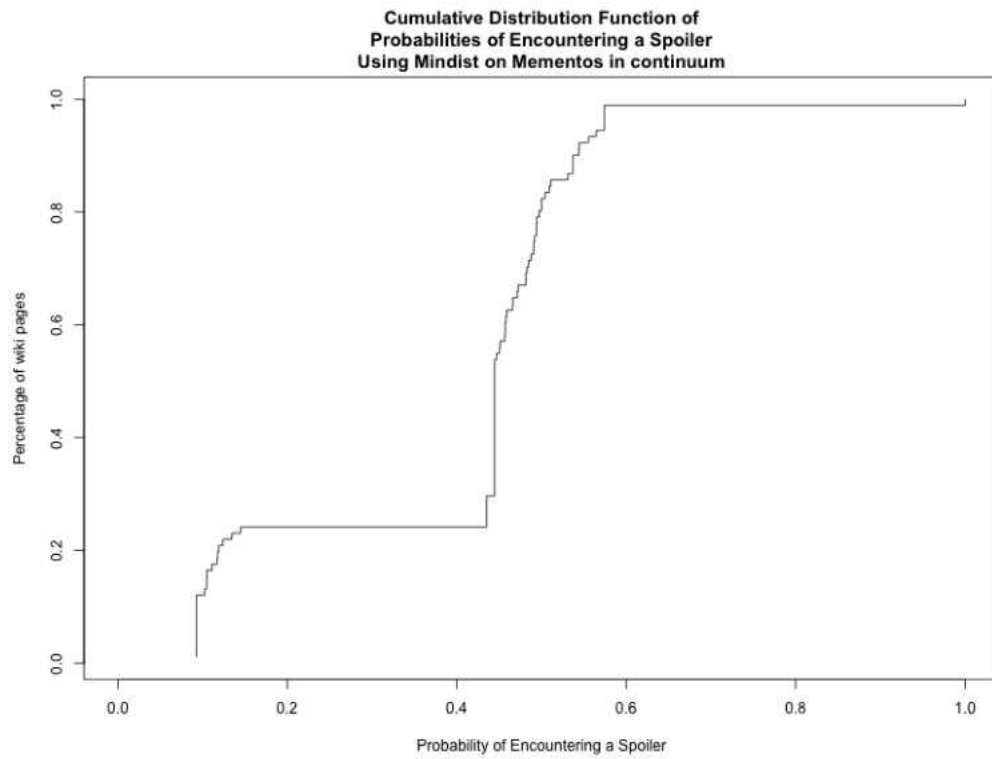


FIG. 124: Continuum

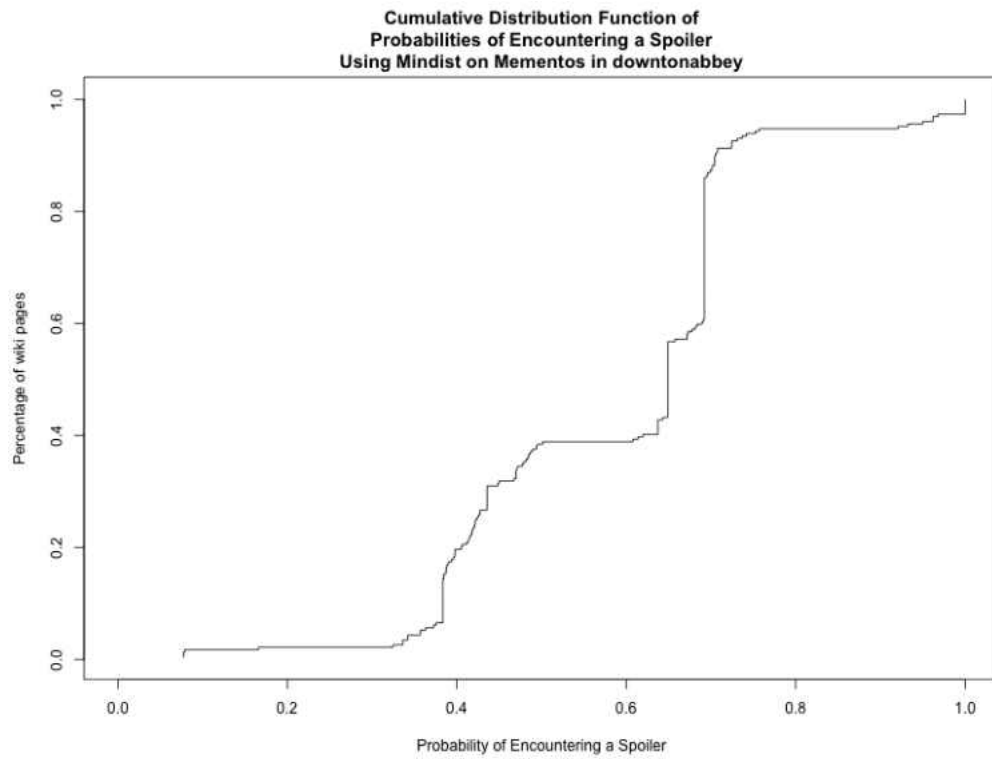


FIG. 125: Downton Abbey

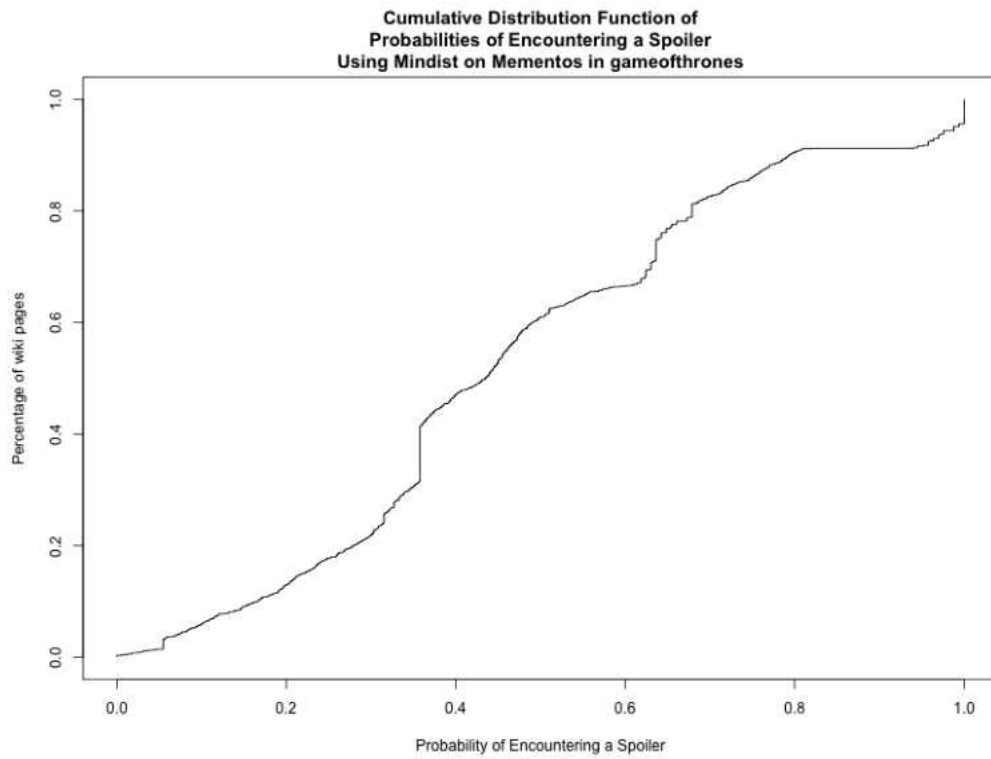


FIG. 126: Game of Thrones

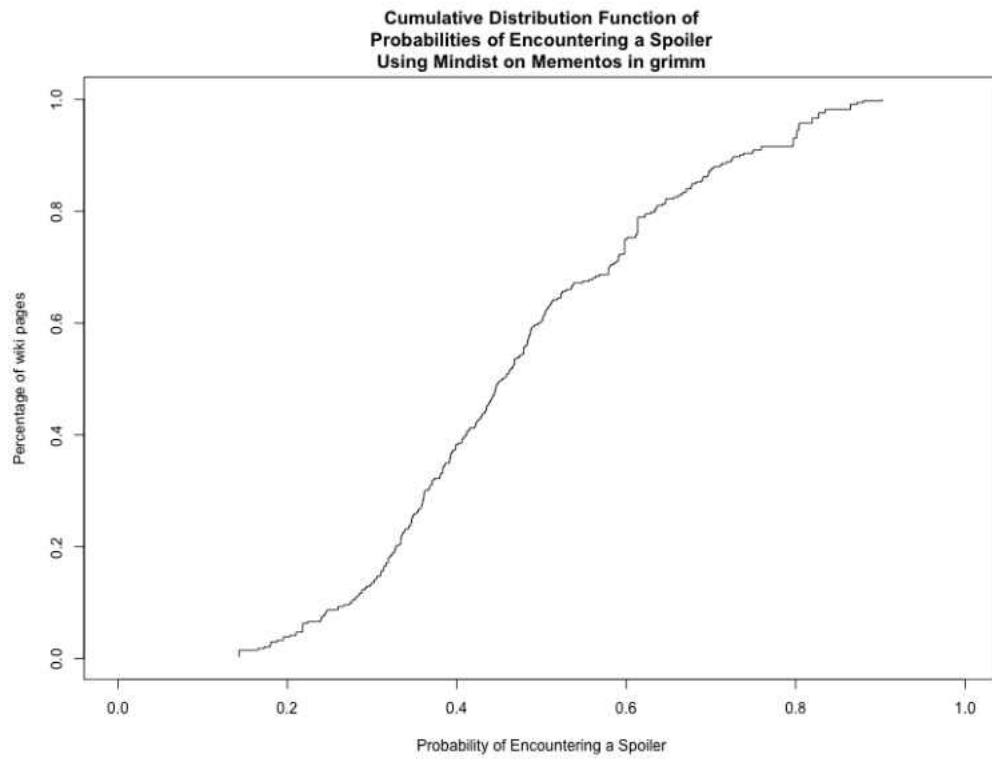


FIG. 127: Grimm

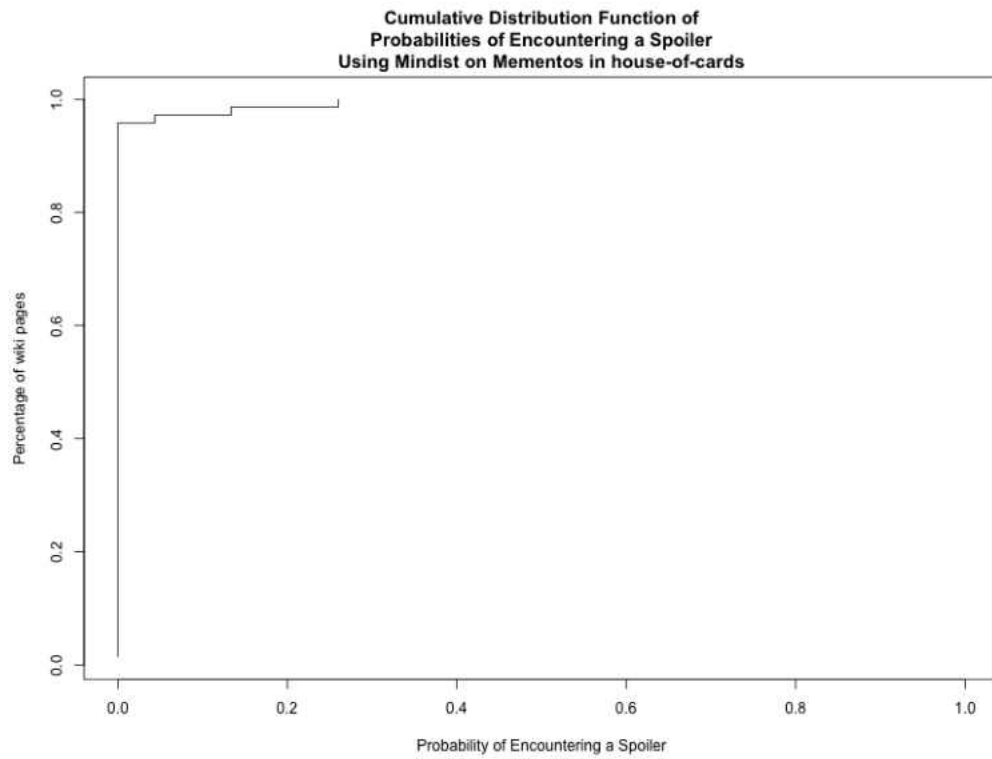


FIG. 128: House of Cards

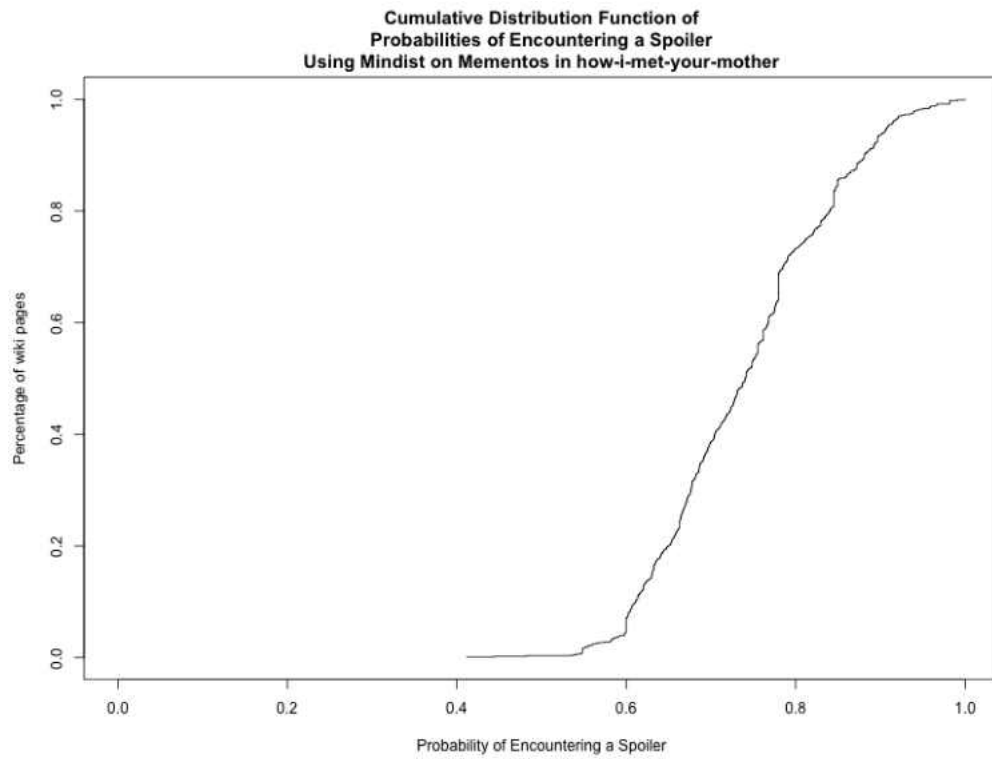


FIG. 129: How I Met Your Mother

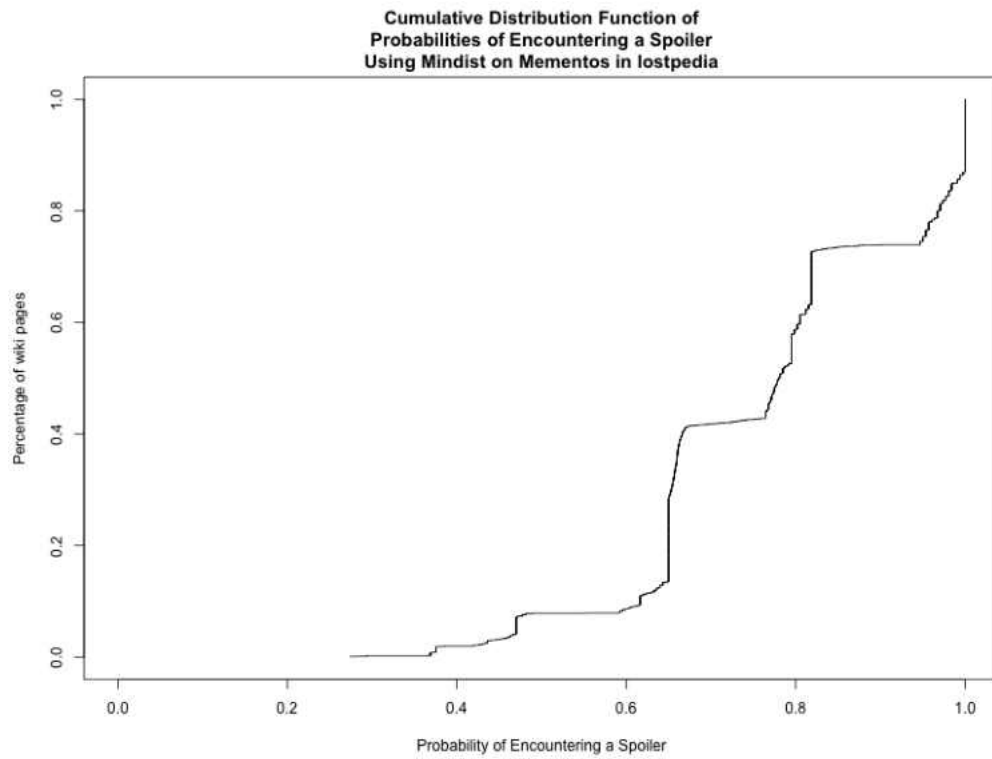


FIG. 130: Lostpedia

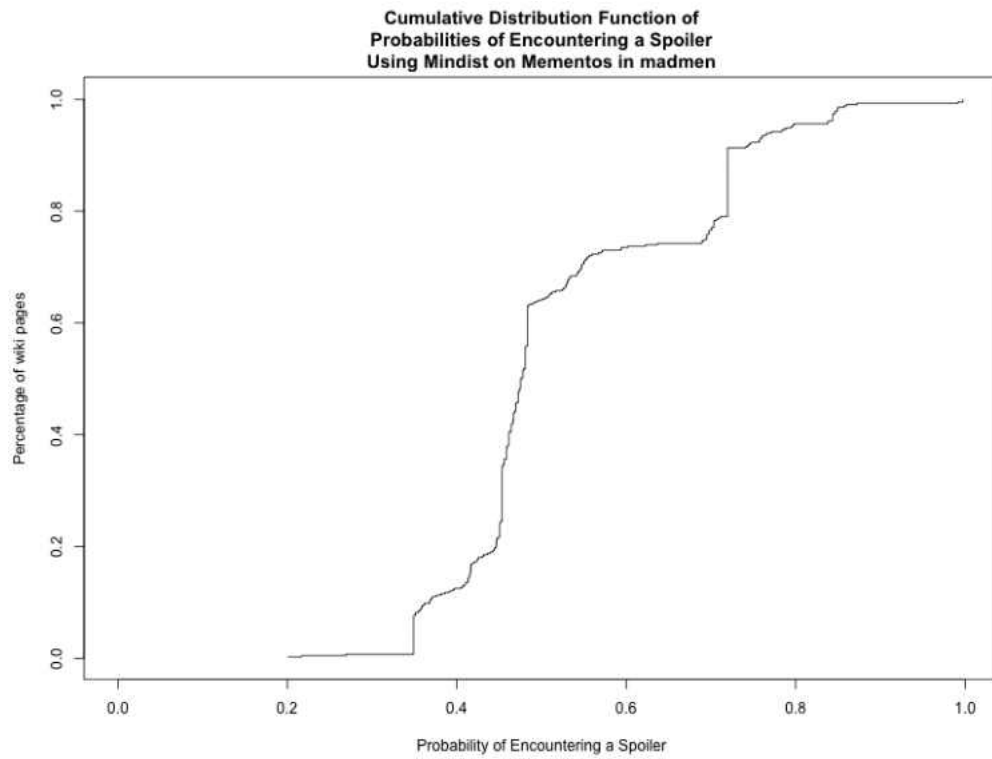


FIG. 131: Mad Men

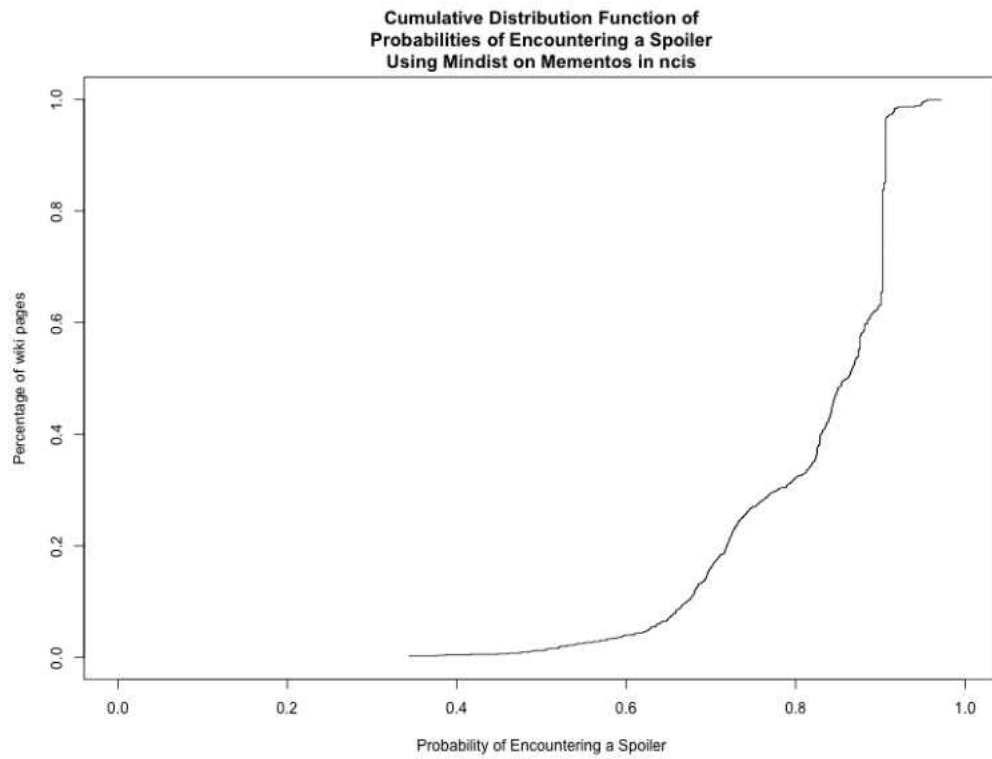


FIG. 132: NCIS

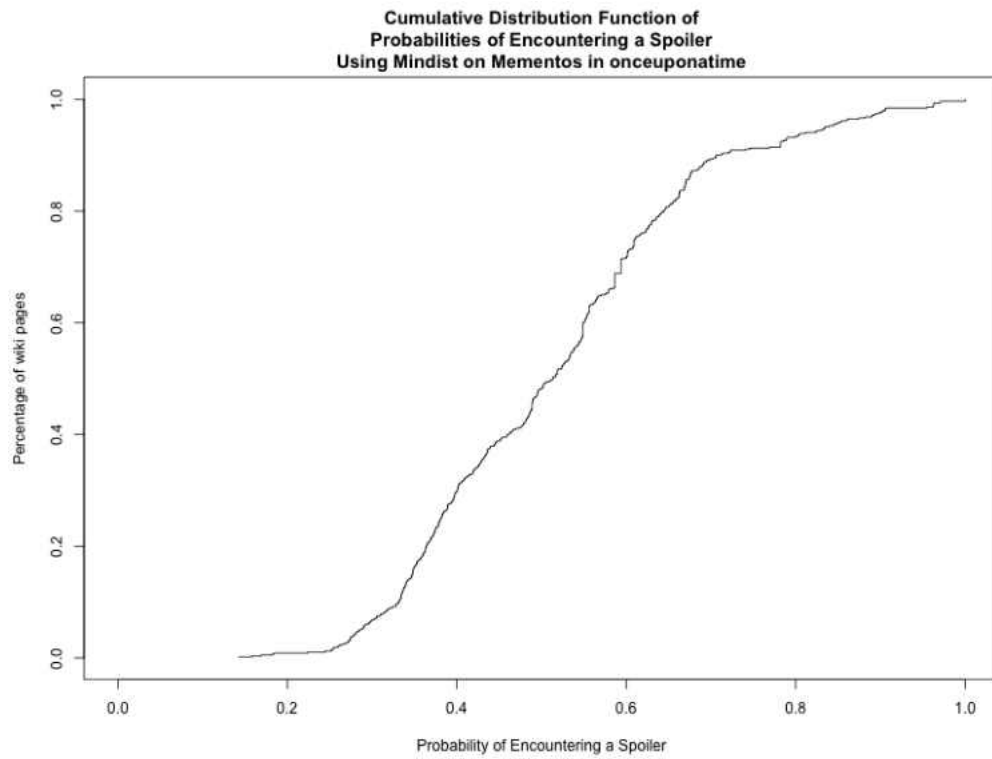


FIG. 133: Once Upon A Time

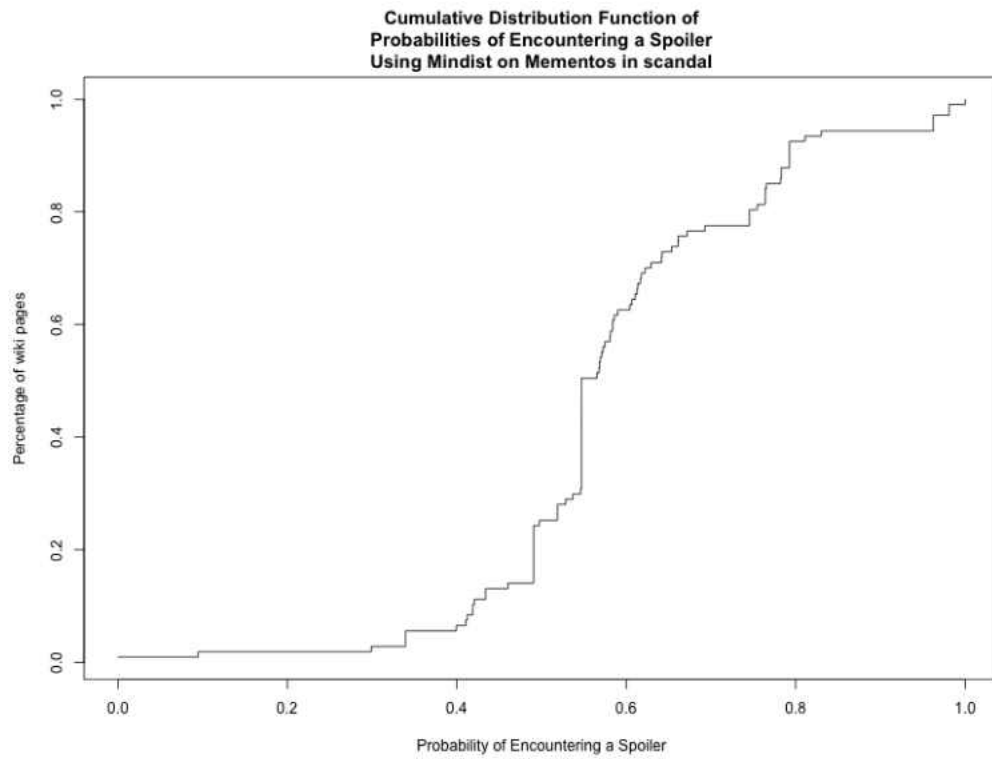


FIG. 134: Scandal

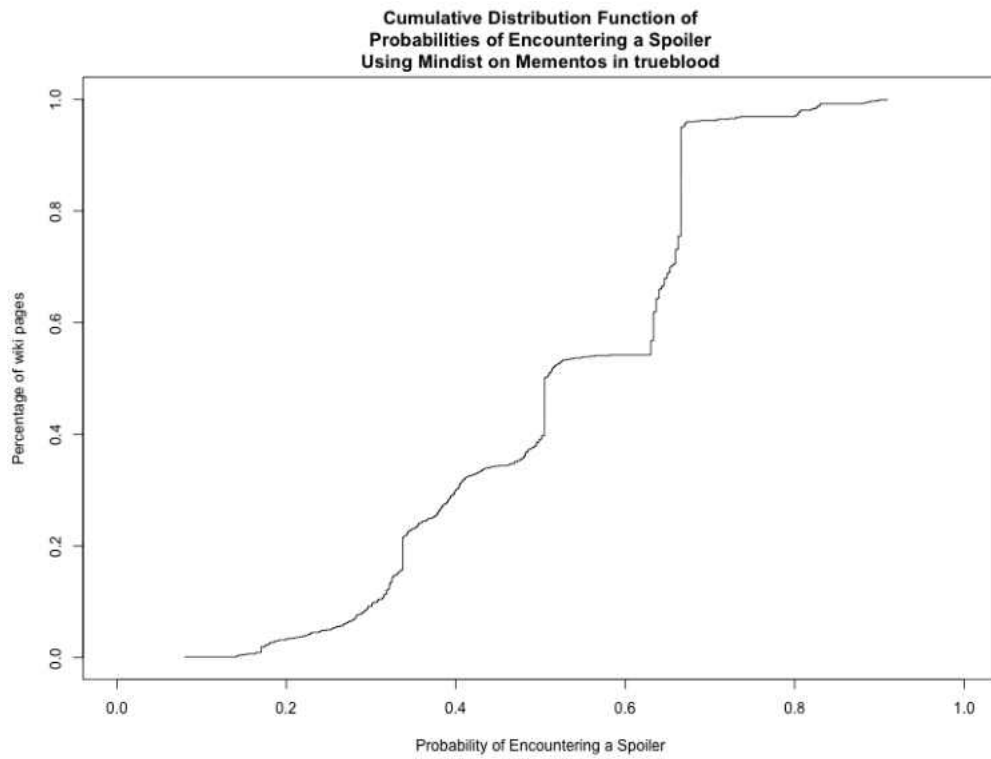


FIG. 135: True Blood

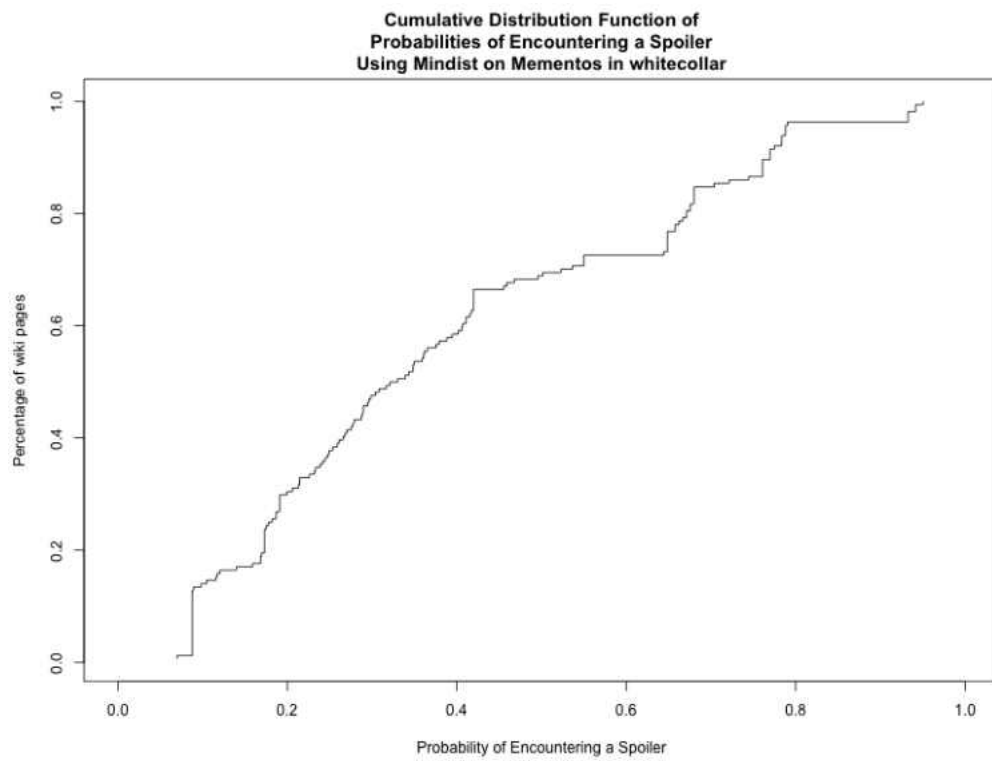


FIG. 136: White Collar

VITA

Shawn M. Jones
 Department of Computer Science
 Old Dominion University
 Norfolk, VA 23529

EDUCATION

M.S. Computer Science, Old Dominion University, 2015
 B.S. Computer Science, Old Dominion University, 1999

EMPLOYMENT

2005 – Computer Scientist, Space and Naval Warfare Systems Center
 Atlantic
 1997 – 2005 Information Technology Specialist, Space and Naval Warfare Sys-
 tems Center Atlantic

PUBLICATIONS AND PRESENTATIONS

2015 Avoiding Spoilers on MediaWiki Fan Sites Using Memento
 2014 Using the Memento MediaWiki Extension To Avoid Spoilers
 2014 Reconstructing the Past With MediaWiki: Programmatic Issues
 and Solutions
 2013 Bringing Web Time Travel to MediaWiki: An Assessment of the
 Memento MediaWiki Extension

AFFILIATIONS

Association for Computing Machinery
 International Information System Security Certification Consortium, Inc.

CONTACT

Email sjone@cs.odu.edu jones.shawn.m@gmail.com
 Homepage <http://www.cs.odu.edu/~sjone>

Typeset using L^AT_EX.