Ricardo Baeza-Yates
Julien Masanès
Marc Spaniol (Eds.)

International Temporal Web Analytics Workshop TWAW 2011

in conjunction with
The 20th International World Wide Web Conference
WWW 2011

1st International Workshop
Hyderabad, India
March 28, 2011
Proceedings

Preface

By its very nature, the Web has a rich relation to time. Emerging and disappearing unpredictably from any part of the planet, reflecting the world, sometimes nearly in real time, full of assertions about the past as well as the future, made at any time of a complex mix of old and new content items linked together, the Web represents a challenging object for temporal-centric research. The difficulty to get appropriate material for this research (time series corpora, large scale archives etc.), adds to the difficulty of exploring the Web's temporal dimension.

Yet, the benefits for research are huge. Trend analysis, emergence of concepts or ideas, representation of the past and the future, network dynamic, shaping and decay of communities, and in general, any Web research topic where a dynamic understanding is superior to a static view, requires integration of the time dimension. For this, establishing solid methods and approaches for temporal Web analytics is a necessity.

Meanwhile, marking the World Wide Web's 20^{th} and the Internet Archive's 15^{th} anniversary in 2011, we now see emerging large collections of digitally-born content to support this research. These archives not only capture the history of digitally-born content but also reflect the zeitgeist of different time periods over more than a decade. These data are a potential goldmine for temporal Web analytics at Web scale.

This, we think, is the perfect timing to establish a dedicated workshop series, the Temporal Web Analytic Workshop (TWAW), which took place for the first time this year, in conjunction with the World Wide Web conference (www2011) in Hyderabad (India).

The objective of this workshop was to provide a venue for researchers of all Web-related domains (IE/IR, Web mining etc.) where the temporal dimension opens up an entirely new range of challenges and possibilities. The workshop's ambition was to help shaping and establishing a community of interest on the research challenges and possibilities resulting from the introduction of the time dimension in Web analysis. Being the first workshop of this kind, it marked the kick-off of an annual workshop series about this novel and emerging topic.

TWAW 2011 was jointly organized by Yahoo! Research (Barcelona, Spain), Internet Memory Foundation (Paris, France), the Max-Planck-Institut für Informatik (Saarbrücken, Germany), and supported by the 7^{th} Framework IST programme of the European Union through the focused research project (STREP) on Longitudinal Analytics of Web Archive data (LAWA) under contract no. 258105.

Ricardo Baeza-Yates
Julien Masanès
Marc Spaniol
Hyderabad (India), March 2011

Organization

Chairs

Ricardo Baeza-Yates (Yahoo! Research, Spain)
Julien Masanès (European Archive Foundation, France and Netherlands)
Marc Spaniol (Max-Planck-Institut für Informatik, Germany)

International Program Committee

Eytan Adar (University of Michigan, USA)
Omar Alonso (Microsoft Bing, USA)
Srikanta Bedathur (IIIT-Delhi, India)
András A. Benczúr (Hungarian Academy of Science)
Klaus Berberich (Max-Planck-Institut für Informatik, Germany)
Adam Jatowt (Kyoto University, Japan)
Scott Kirkpatrick (Hebrew University Jerusalem, Israel)
Michael Matthews (Yahoo! Research, Spain)
Christian König (Microsoft Research, USA)
Frank McCown (Harding University, USA)
Michael Nelson (Old Dominion University, USA)
Kjetil Nørvåg (Norwegian University of Science and Technology, Norway)
Thomas Risse (L3S Research Center, Germany)
Pierre Senellart (Télécom ParisTech, France)
Torsten Suel (NYU Polytechnic, USA)
Masashi Toyoda (Tokyo University, Japan)
Peter Triantafillou (University of Patras, Greece)
Gerhard Weikum (Max-Planck-Institut für Informatik, Germany)

Table of Contents

Temporal Information Retrieval: Challenges and Opportunities

Omar Alonso
Microsoft Corp.
Mountain View, CA
omar.alonso@microsoft.com

Jannik Strötgen
Institute of Computer Science
University of Heidelberg, Germany
stroetgen@uni-hd.de

Ricardo Baeza-Yates
Yahoo! Research
Barcelona, Spain
rbaeza@acm.org

Michael Gertz
Institute of Computer Science
University of Heidelberg, Germany
gertz@uni-hd.de

ABSTRACT

Time is an important dimension of any information space. It can be very useful for a wide range of information retrieval tasks such as document exploration, similarity search, summarization, and clustering. Traditionally, information retrieval applications do not take full advantage of all the temporal information embedded in documents to provide alternative search features and user experience. However, in the last few years there has been exciting work on analyzing and exploiting temporal information for the presentation, organization, and in particular the exploration of search results.

In this paper, we review the current research trends and present a number of interesting applications along with open problems. The goal is to discuss interesting areas and future work for this exciting field of information management.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Language models, Text analysis*

Keywords

temporal information, information retrieval

1. INTRODUCTION

Time clearly plays a central role in any information space, and it has been studied in several areas like information extraction, topic-detection, question-answering, query log analysis, and summarization. Time and temporal measurements can help recreating a particular historical period or describing the chronological context of a document or a collection of documents. As an extension to existing ranking techniques, which are primarily based on popularity or reputation, time can be in particular valuable for exploring search results along well-defined timelines and at multiple time granularities due to the key characteristics of temporal information:

- Temporal information is well-defined: Assuming two points in time or two intervals, the relationship between them can be identified, e.g., the relationship can be of the types before, overlap, or after [3].
- Temporal information can be normalized: Regardless of the used terms or the used language, every temporal expression referring to the same semantics can be normalized to the same value in some standard format. This property makes temporal information term- and language-independent.
- Temporal information can be organized hierarchically: Temporal expressions can be of different granularities, e.g., of type day ("May 20, 2011") or of type year ("2011"). Due to the fact that years consist of months, and months and weeks consist of days, temporal expressions can be mapped to coarser granularities based on the hierarchy of temporal expressions.

Using these key characteristics, temporal information about documents can be used to develop time-specific information retrieval and exploration applications. The most obvious type of temporal information associated with a document is its creation time or the date of its last modification. This kind of information, which is directly accessible through the metadata of a document, can already be used for several tasks such as time-aware search or temporal clustering. However, the document creation time is only valuable in a specific context such as the news domain. In other areas, and even in the news domain itself, a lot of temporal information is neglected if the document creation time is used as the only time information associated with a document. This is because there is a lot of temporal information latently available in a document's text. Assume a news document reports on an event that is already dated. Then, if only the

TWAW 2011, March 28, 2011, Hyderabad, India.

document creation time is taken into account, the information when the event occurred is ignored. But to make use of such information, so-called temporal taggers are applied to extract and normalize temporal expressions contained in documents.

The remainder of the paper is organized as follows. After a discussion of how time appears in documents and how it is possible to extract such temporal data, in Section 3, we survey research on temporal tagging. In Section 4, we present the current research trends on temporal information retrieval. We then describe application areas and challenges. Finally, we present our concluding remarks.

2. TIME IN DOCUMENTS

As indicated in the introduction, there is a lot of temporal information in any collection of documents, be it ranked documents in a hit list or a corpus of topic specific documents. To take advantage of such time related information in particular for document exploration purposes, in a document processing step, it is important to extract this information, anchor it in time, compute some (aggregated) measures, and make all this information explicit to subsequent exploration tasks.

In this section, we give a description of the different types of temporal information mentioned in documents (Sec. 2.1), explain how temporal expressions can be realized in natural language (Sec. 2.2), and demonstrate how they can be extracted and normalized (Sec. 2.3).

2.1 Types of Temporal Information

Temporal expressions mentioned in text documents can be grouped into four types according to TimeML [27], the standard markup language for temporal information: date, time, duration, and set. While duration expressions are used to provide information about the length of an interval (e.g., "three years" in *they have been traveling through the U.S. for three years*), set expressions inform about the periodical aspect of an event (e.g., "twice a week" in *she goes to the gym twice a week*). In contrast, time and date expressions (e.g., "3 p.m." or "January 25, 2010") both refer to a specific point in time – though in a different granularity.

An interesting key feature of temporal information is that it can be normalized to some standard format. Assuming a Gregorian calendar as representation of time, expressions of time and date can be directly placed on a timeline. A date is then typically represented as YYYY-MM-DD, e.g., the expression "January 25, 2010" is normalized to "2010-01-25". However, the normalization is not always as simple as in this example, but depends on the way temporal information is expressed in a document, which will be discussed in the next paragraph.

2.2 Occurrences of Temporal Expressions

There are many different ways how to express temporal information of the types date and time in documents. Similar to the work by Schilder and Habel [31], we distinguish between explicit, implicit, and relative temporal expressions.

Explicit temporal expressions refer to a specific point in time. Note that this point in time can be of different granularities. For example, the expression "January 25, 2010" refers to a specific day while the expression "November 2005" refers to a specific month. The key characteristic of explicit temporal expressions is that they can be normalized without any further knowledge. That is, the expression itself contains all the information needed for normalization and is thus fully specified.

In contrast, *relative temporal expressions* cannot be normalized without taking into account some context information. For example, the expression "today" cannot be normalized without knowing the corresponding reference time. This reference time can either be the document creation time or another temporal expression in the document. Typically, in news articles, the document creation time is important and often used as reference time. Note that this kind of information is directly accessible in form of a timestamp through the metadata of a document. The expression "yesterday" in *Thousands of prisoners in Egypt managed to escape from prison yesterday* can be normalized to "2011-01-29" if we know the document creation time to be "2011-01-30". In other types of documents, the reference time is usually represented by another temporal expression in the document. In general, there are many occurrences of relative temporal expressions. While sometimes the reference time is sufficient for normalization, further information is needed if the relation to the reference time has to be identified as well. For example, "on Monday" can either refer to the previous or to the next Monday with respect to the reference time. An indicator for determining the relationship can be the tense of the sentence with future tense and present tense indicating an after-relationship to the reference time and past tense a before-relationship. Figure 1 shows some parts of a news article containing explicit and relative temporal expressions and illustrate what kind of context information is needed for normalizing the relative expressions.

The third type of temporal expressions are *implicit expressions* such as names of holidays or events. These expressions can be anchored on a timeline if a mapping of the expression to its normalized value is available. For example, "New Year's Day 2002" can be normalized to "2002-01-01" since "New Year's Day" always refers to January 1. In addition, there are expressions for which a temporal function has to be applied. "Labor Day", for example, refers to the first Monday in September so that "Labor Day 2009" can be normalized to "2009-09-07" if we know this day to be the first Monday in September 2009.

Although there are many different ways to refer to a specific point in time, all expressions referring to the same point in time shall be normalized to the same value in the standard format. This normalization process is one of the tasks of so-called temporal taggers, as described in the next paragraph.

2.3 Temporal Tagging

Temporal tagging is a specific task in named entity recognition and normalization. The goals of so-called temporal taggers are (i) the extraction of temporal expressions and (ii) the normalization of these expressions to some standard format. As this standard format, TIMEX2 and TIMEX3 are often used. While TIMEX2 tags include pre- and post-modifiers of the temporal expression itself (e.g., dependent clauses) and allow for nested temporal expressions [11], such modifiers and nested tags are not included by TIMEX3 tags. Instead, TIMEX3 is part of the TimeML markup language in which further annotation types are available for capturing more complex temporal semantics. Nevertheless, although there are significant differences between TIMEX2

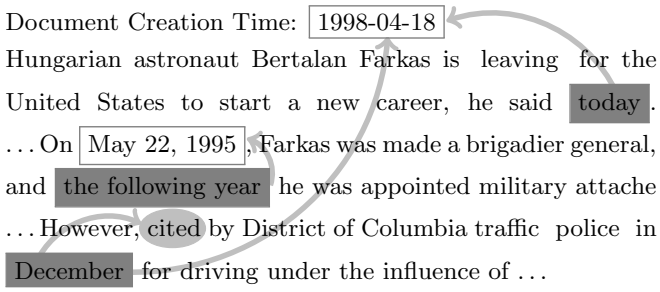

Figure 1: Examples of temporal expressions in a news article with explicit (transparent boxes) and relative (solid boxes) expressions. Arrows indicate what kind of context information is needed to normalize the temporal expression.

and TIMEX3, they are very similar in many ways and a detailed analysis goes beyond the scope of this paper.[1] According to the TimeML annotation guidelines, a TIMEX3 tag contains, among others, the following information about a temporal expression:

- offset: the start and end position of the expression in the document
- type: whether the expression is of type date, time, duration, or set
- value: the normalized value of the expression

To identify this information, i.e., to extract and normalize temporal expressions, temporal taggers are applied after the text is preprocessed. Usually, sentence and token boundaries are detected and a part-of-speech tag is associated with every token. This information can then be used by the temporal tagger to identify temporal expressions. The first goal, i.e., the identification of the boundaries of temporal expressions, can be seen as typical classification task. Therefore, there has been some work on addressing this problem by applying machine learning techniques (e.g., [15, 38]). The classification problem can be described in the following way: For every token t, decide whether t is outside (O) of temporal expressions, inside (I) a temporal expression, or the beginning (B) of a temporal expression. The well-known IOB-format can be used for annotating tokens according to their property.

In addition to machine-learning approaches, there are several rule-based approaches to extract temporal expressions (e.g., [24, 34]). These are usually based on regular expressions although they may use other information about the text as well, such as part-of-speech information.

The more difficult task is the normalization of the temporal expressions. While explicit expressions can be normalized without further knowledge, the normalization of relative expressions is challenging. As described above, context information has to be identified to determine the correct reference time and the temporal relation between a temporal expression and its reference time. While there are rule-based and machine learning based approaches for the extraction of temporal expressions, the normalization is usually done in a rule-based way by all temporal taggers.

Due to their importance for temporal information retrieval, we give an overview of existing temporal taggers and their quality in the next section. In addition, we present resources for evaluating temporal taggers and survey temporal evaluation challenges organized so far.

3. RESEARCH ON TEMPORAL TAGGING

Temporal processing of text documents in terms of the extraction and normalization of temporal expressions as well as the extraction of temporal relations between events is very important for several NLP tasks requiring a deep understanding of language such as question answering or document summarization. Due to this fact, there has been significant research in temporal annotation of text documents. The markup language TimeML has become an ISO standard for temporal annotation [27], and the TimeBank corpus was developed [28]. The latest version of the TimeBank corpus contains 183 news articles and can be regarded as the gold standard for temporal annotation. However, there has been important research activity before, and several evaluation challenges have been held to bring forward research in the area of temporal information extraction as described in the following section.

3.1 Evaluation Challenges

The earliest competitions for the extraction of temporal expressions have been the named entity recognition tasks of the Message Understanding Conferences MUC 1995 and 1997 [8, 12]. A combination of the extraction and the normalization was introduced in the ACE (Automatic Content Extraction) time expression recognition and normalization (TERN) challenges in 2004, 2005, and 2007[2]. Several temporal taggers have been developed by the participants of these challenges (see Section 3.2). Often, the TERN 2004 and 2005 corpora[3] are used to compare the quality of temporal taggers. The TERN corpora are annotated with respect to the TIMEX2 annotation guidelines [11].

A further indication of the importance of temporal annotation and the activity in the research domain are the TempEval challenges. Motivated by the importance of temporal annotation for many NLP tasks, TempEval was organized the first time as one task of the SemEval workshop in 2007 [39]. In this competition, the organizers provided annotated text documents based on the TimeBank corpus. While the annotations of events and temporal expressions were given, the task for the tools to be developed was to identify temporal relations between events and the document creation time, between events and temporal expressions, and between two events.

In 2010, the full task of identifying all temporal related expressions and relations was faced in the follow-up challenge. That is, for TempEval-2, two further tasks were added [41]: the extraction and normalization of temporal expressions and of events. In addition, the discovery of relations between two events was split into two tasks, namely the identification of relations between two main events in consecu-

[1]For further information on temporal annotation according to TimeML and differences between TIMEX2 and TIMEX3, see `http://www.timeml.org`.

[2]`http://www.itl.nist.gov/iad/mig/tests/ace/`.

[3]The TERN development corpora are available through the Linguistic Data Consortium. See: `http://fofoca.mitre.org/tern.html`.

tive sentences and relations between two events where one event syntactically dominates the other event. The TempEval corpora are based on the TimeBank corpus and annotated according the TimeML annotation guidelines, i.e., using TIMEX3 tags for temporal expressions. A further novelty in the second TempEval challenge was that the tasks were offered not only in English but in six languages. However, only two languages where addressed by the participants, namely English and Spanish. Nevertheless, thanks to the creation of an annotation standard, a gold standard corpus, and competitions such as the TempEval challenges, there has been significant improvements in temporal relation identification and temporal tagging. Some existing temporal taggers and their quality is presented in the next paragraph by comparing their results in the TempEval-2 challenge.

3.2 Temporal Taggers

Having applied a temporal tagger on a document collection, the previously hidden temporal information is made available for tasks such as temporal relation extraction or temporal clustering. One often applied temporal tagger is GuTime, which is part of the Tarsqi tool kit [40]. It is based on the TempEx tagger, which was the first temporal tagger for the extraction of temporal expressions and their normalizations [22]. GuTime was developed as automatic evaluation tool for TimeML and extends the capabilities of the TempEx tagger. It was evaluated on the TERN 2004 training corpus and achieves F-measures of 85%, 78%, and 82% for lenient and strict detection and for normalization, respectively.

In the TempEval-2 challenge, eight teams participated in the task for temporal expression extraction and normalization for English documents. The best-performing system was HeidelTime with an F-Score of 86% for the extraction and an accuracy of 85% for the normalization [34]. For Spanish documents, the best result for the extraction was an F-Score of 91% [16], while another system achieved the highest accuracy for normalization (83%) [42]. While both machine-learning and rule-based approaches were applied for the extraction, the normalization was done in a rule-based way by all systems. As the best performing system, HeidelTime uses rules consisting of an extraction and a normalization part. Thus, all temporal expressions that are identified are normalized as well. Due to the strict separation of the code and the rules, HeidelTime is applicable for multi-lingual temporal tagging[4].

Although there has been significant advances in temporal tagging, there is still room for improvement, especially when switching the processing language or the domain of the document collection. For example, Mazur and Dale recently presented a new corpus for research on temporal expressions containing long, narrative-style documents, namely Wikipedia articles describing the historical course of wars [25]. Using their temporal tagger, they show that the normalization of temporal expressions in such documents is very challenging due to the rich discourse structure and the huge number of often underspecified temporal expressions in these documents compared to the usually used short news documents.

[4]For details, see `http://dbs.ifi.uni-heidelberg.de/stixx/`.

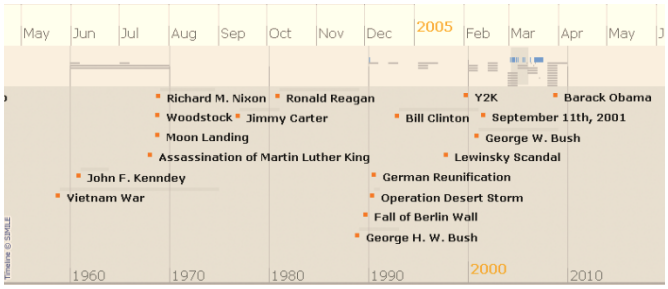

Figure 2: Annotation of a timeline by workers using crowdsourcing.

4. RESEARCH TRENDS

Research work on fully utilizing the temporal information embedded in the text of documents for exploration and search purposes is very recent. The work by Alonso et al. presents an approach for extracting temporal information and how it can be used for clustering search results [5]. Berberich et al. describe a model for temporal information needs [7]. Figure 2 shows the annotated timeline for the NYTimes data set for the latter reference using the Timeline widget[5]. These last two projects rely on crowdsourcing, mainly using Amazon Mechanical Turk, for evaluating parts of their work.

News sources have been the primary focus of a number of projects on exploiting time information in documents. For example, the Time Frames project realizes an approach to augment news articles by extracting time information [14]. Google's news timeline[6] is an experimental feature that allows a user to explore news by time.

Extensions to document operations such as comparing the temporal similarity of two documents in the context of news articles is presented by Makkonen and Ahonen-Myka [17]. An interesting approach that combines topic detection and tracking with timelines as a browsing interface is presented by Swan and Allan [37]. Time information is also used in temporal mining of blogs to extract useful information [29]. New research has also emerged for *future retrieval* where temporal information is used for searching the future [6].

There is exciting research on adding a time dimension to certain applications like news summaries [2], temporal patterns [33], and temporal Web search [26]. The special issue on temporal information processing gives a clear map of current directions [20]. Harvesting temporal information and how it can be used for entities and relationships is also a very recent rich area [43].

Closely related to information extraction is the recent research on *temporal annotations*, which is covered in depth in the book by Mani et al. [21]. Identification of time related information depends heavily on the language and the corpora, so traditional information extraction systems tend to fall short in terms of temporal extraction. Based on the latest advances, new research is emerging for automatic assignment of document event-time periods and automatic tagging of news messages using entity extraction [31].

Now, we outline a number of applications that can benefit from leveraging more temporal information either by temporal expressions or timestamps. For each application, we

[5]`http://simile.mit.edu/`
[6]`http://www.newstimeline.googlelabs.com`

describe why it is important and present a number of challenges.

4.1 Exploratory Search

Research in exploratory search systems has gained a lot of attention lately as they add a significant user interface component to help users search, navigate, and discover new facts and relationships. As the amount of information on the Web keeps growing, exploratory search interfaces are starting to surface. That said, it is not clear how to leverage temporal information. A few problems are:

- How to expose temporal information in exploratory search systems?
- What's the best way of presenting temporal information as a retrieval *cue*?
- For which data sources, besides news, does exploratory search make sense?
- Is e-discovery a vertical application that can benefit from temporal information?

4.2 Micro-blogging and Real-time Search

Micro-blogging sites like Twitter have gained a lot of attention lately as the ultimate mechanism to broadcast what's going on. Due to its nature, a typical message is very short and its lifespan is basically the *crowd* interest about that particular event be a football final game or an earthquake.

In the case of Twitter, it is very difficult to beat the timely broadcasting of an important event if one compares this to a news article. Each tweet has a timestamp but the organization of such information is still not clear. In the news context, the reporter has to write an article that contains a few paragraphs and submit the final version through some content management version that would push it to an external website so a search engine can hopefully crawl and index it in time. In parallel, if a tweet is so important by the time the reporter is finishing with the article, the main idea would be trending in Twitter, therefore highlighting its importance at a world scale. This is very similar to the traditional notion of topic detection and tracking [1, 18], with one key difference: speed to detect that the topic is important and therefore a candidate for trending. Some problems are:

- What is the best way to provide a timeline of events in micro-blogging?
- What is the lifespan of the main event?
- How fast and precise can one detect trending events?
- What is the fraction of new content on the topic stream?

4.3 Temporal Summaries

There has been seminal work on temporal summaries of news topics by Allan [2] that shows how important temporal information is. One extension is to generate time sensitive summaries that can be used as temporal snippets [4].

By design, the main goal of a snippet (or caption) is to present a document surrogate that the user can quickly scan in the search results page without the need to click and read the full content of a document. There is a limit to the number of lines of text that the snippet should present. Interesting questions include:

- When to show a timestamp or temporal expressions?
- Should the snippet present the matching lines in a timeline?
- Is a temporal summary a good surrogate for a document?
- For which kind of queries is a temporal summary appropriate?
- Should temporal summaries be query independent?

4.4 Temporal Clustering

The notion of clustering search results by temporal attributes has been presented in [5]. Preliminary results indicate that users are interested in dissecting a document collection by time. At the same time it is not clear for which kind of scenarios besides "research-like" questions this approach would work. Key issues are:

- Can we identify documents that are contemporary and therefore related?
- Which chronons can be more useful for clustering?
- How can we cluster micro-blogging data by time?
- Is a timeline the best way to cluster search results?

4.5 Temporal Querying

The temporal information extracted from documents can directly be used to allow the user of a search engine to constrain his/her query in a temporal manner. That is, in addition to a textual part, a query contains a temporal part. For example, in addition to "world war" a temporal constraint like "1944-1945" could be specified. The user would obviously expect documents about World War II as results for his query. The objective when using a combination of a text and a temporal query can thus be formulated in the following way: The more both parts of the query are satisfied, i.e, the more the textual and the temporal parts fit to a document, the higher should be the rank of this document. The main problems for such a combination of constraints is the following:

- How can a combined score for the textual part and the temporal part of a query be calculated in a reasonable way?
- Should a document in which the "textual match" and the "temporal match" are far away from each other be penalized?
- What about documents satisfying one of the constraints but "slightly" fail to satisfy the other constraint?

4.6 Temporal Question Answering

To be able to answer time-related questions, a question answering system has to know when specific events took place. For this, temporal information can be associated with extracted facts from text documents [26]. While this may be applicable for famous facts and events, question answering systems are often faced with imperfect temporal information. For this, identifying relationships between events described in documents is important as it is for many further NLP tasks (see Section 3.1). Especially historic events tend

to have a gradual beginning and ending so that knowing the temporal relationship between two events may allow to answer a temporal query although no explicit temporal information is associated with the events [30]. Research issues are:

- How can inconsistent temporal information be dealt with?
- How can temporal reasoning be executed if temporal relationships are missing?

4.7 Temporal Similarity

A related research question to temporal querying is temporal document similarity. Instead of comparing a temporal query with the temporal information of a document, two documents can be compared with respect to their temporal similarity. The main problem arising here is what makes two documents temporally similar? This leads to the following questions:

- Should two documents be considered similar if they cover the same temporal interval?
- Should the temporal focus of the documents be important for their temporal similarity?
- Can two documents be regarded as temporally similar if one contains a small temporal interval of the other document in a detailed way?

4.8 Timelines and User Interfaces

One important use of time entities of a document is to create a sorted list of events, a timeline. A timeline can be shown as a list of vertical textual items or visualized in many different ways. For example, as in Yahoo!'s Correlator[7]. More sophisticated visualizations allow to focus on specific named entities with respect to time like in Yahoo!'s News Explorer [10, 19]. Here, interesting questions are:

- What is the appropriate way to present a timeline?
- Is a linear timeline the only way to present and anchor documents in time?
- How can one leverage document temporal measures to present a good display?
- Are there specific visualizations or user interfaces that can benefit from temporal information?

4.9 Searching in Time

Time entities can also be used to search in documents or log files that can be used to search the past for different purposes such as digital forensics, historical analysis or linguistic analysis. We can even search the future [6, 13], for example, in news for events that are scheduled or may happen in the future. This idea is supported in the Yahoo!s News Explorer tool already mentioned [19]. Microsoft Academic Search[8] is an example of presenting publications and citations in a timeline. Some problems are:

- Besides news, what other sources would one like to use to search in the past and/or the future?
- How far does one need to go back in time?
- Can we improve bibliographic search instead of just sorting by publication date?
- How can we evaluate the quality of such systems?

4.10 Web Archiving

The goal of Web archiving is to collect and store digital content so that it is accessible for future tasks. Besides the detection of spam, which can be dealt with analyzing the evolution of the link structure of web pages [9], a main challenge in Web archiving is to take care of the temporal coherence of Web pages since it is not possible to collect all pages at the same time. Thus, the content of parts of the collection may change during the crawling process. In [32], Spaniol et al. introduce a coherence framework to overcome the temporal diffusion of the Web crawls, i.e., to minimize the risk of incoherences. Nevertheless, open problems remain:

- How can temporal information be used to predict which pages are likely to change over time?
- How can temporal coherence be achieved for any point in time or time interval?

4.11 Spatio-temporal Information Exploration

Recently, there has been some research on combining spatial and temporal information extracted from documents for exploration tasks [23, 36]. In the same way as temporal information can be normalized using a timeline, spatial information can be normalized according its latitude/longitude information. To extract geographic expressions from documents, so-called geo taggers can be applied. Combining the information extracted from a temporal tagger with the information extracted from a geo tagger allows the exploration of documents according to the events mentioned in the text since events usually happen at some specific time and place. A system for the exploration of such spatio-temporal information from documents is TimeTrails [35]. Some questions are:

- What's the best way to represent maps and time?
- Which kinds of scenarios can benefit from spatio-temporal exploration?

5. CONCLUDING REMARKS

Temporal information embedded in documents in the form of temporal expressions offer an interesting means to further enhance the functionality of current information retrieval applications.

We have presented a number of examples and scenarios where temporal information can be very useful. We have identified research trends in this new area and a number of interesting practical applications as well as problems.

The problems we outline are difficult because they include several areas of computer science, mainly information retrieval, natural language processing, and user interfaces. Moreover, several of them are multidisciplinary because they touch issues related to psychology or design, to mention just two, making them even more challenging.

[7] http://correlator.sandbox.yahoo.com/.
[8] http://academic.research.microsoft.com/.

6. REFERENCES

[1] J. Allan, editor. *Topic Detection and Tracking: Event-based Information Organization*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[2] J. Allan, R. Gupta, and V. Khandelwal. Temporal Summaries of New Topics. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)*, pages 10–18, 2001.

[3] J. F. Allen. Maintaining Knowledge about Temporal Intervals. In *Communications of the ACM*, 26(11):832–843, 1983.

[4] O. Alonso, R. Baeza-Yates, and M. Gertz. Effectiveness of Temporal Snippets. In *Proceedings of the Workshop on Web Search Result Summarization and Presentation (WSSP 09)*, pages 1–4, 2009.

[5] O. Alonso, M. Gertz, and R. Baeza-Yates. Clustering and Exploring Search Results Using Timeline Constructions. In *Proceedings of the 18th ACM International Conference on Information and Knowledge Management (CIKM '09)*, pages 97–106, 2009.

[6] R. Baeza-Yates. Searching the Future. In *Proceedings of the ACM SIGIR 2005 Workshop on Mathematical/Formal Methods in Information Retrieval (MF/IR 05)*, pages 1–6, 2005.

[7] K. Berberich, S. Bedathur, O. Alonso, and G. Weikum. A Language Modeling Approach for Temporal Information Needs. In *Proceedings of the 32nd European Conference on Information Retrieval Research (ECIR '10)*, pages 13–25, 2010.

[8] N. Chinchor. Overview of MUC-7/MET-2. In *Proceedings of the 7th Conference on Message Understanding (MUC '97)*, pages 1–11, 1997.

[9] Y. Chung, M. Toyoda, and M. Kitsuregawa. A Study of Link Farm Distribution and Evolution Using a Time Series of Web Snapshots. In *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb '09)*, pages 9–16, 2009.

[10] G. Demartini, M. Missen, R. Blanco, and H. Zaragoza. TAER: Time-aware Entity Retrieval. Exploiting the Past to Find Relevant Entities in News Articles. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM '10)*, pages 1517–1520, 2010.

[11] L. Ferro, L. Gerber, I. Mani, B. Sundheim, and G. Wilson. TIDES - 2005 Standard for the Annotation of Temporal Expressions. 2005. `http://fofoca.mitre.org/annotation_guidelines/2005_timex2_standard_v1.1.pdf`

[12] R. Grishman and B. Sundheim. Design of the MUC-6 Evaluation. In *Proceedings of the 6th Conference on Message Understanding (MUC '95)*, pages 1–11, 1995.

[13] A. Jatowt, K. Kanazawa, S. Oyama, and K. Tanaka. Supporting Analysis of Future-related Information in News Archives and the Web. In *Proceedings of the 9th Joint Conference on Digital Libraries (JCDL '09)*, pages 115–124, 2009.

[14] D. Koen and W. Bender. Time Frames: Temporal Augmentation of the News. In *IBM Systems Journal*, 39(4): 597–616, 2000.

[15] O. Kolomiyets and M.-F. Moens. Meeting TempEval-2: Shallow Approach for Temporal Tagger. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW '09)*, pages 52–57, 2009.

[16] H. Llorens, E. Saquete, and B. Navarro. TIPSEM (English and Spanish): Evaluating CRFs and Semantic Roles in TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval '10)*, pages 284–291, 2010.

[17] J. Makkonen and H. Ahonen-Myka. Utilizing Temporal Information in Topic Detection and Tracking. In *Proceedings of 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL '03)*, pages 393–404, 2003.

[18] J. Makkonen, H. Ahonen-Myka, and M. Salmenkivi. Topic Detection and Tracking with Spatio-temporal Evidence. In *Proceedings of the 25th European Conference on Information Retrieval Research (ECIR '03)*, pages 251–265, 2003.

[19] M. Matthews, P. Tolchinsky, R. Blanco, J. Atserias, P. Mika, and H. Zaragoza. Searching Through Time in the New York Times. In *Proceedings of the Fourth Workshop on Human-Computer Interaction and Information Retrieval (HCIR '10)*, pages 41–44, 2010.

[20] I. Mani, J. Pustejovsky, and B. Sundheim. Introduction to the Special Issue on Temporal Information Processing. In *ACM Transactions on Asian Language Information Processing* 3(1): 1–10, 2004.

[21] I. Mani, J. Pustejovsky, and R. Gaizauskas, editors. *The Language of Time*. Oxford University Press, New York, NY, USA, 2005.

[22] I. Mani and G. Wilson. Robust Temporal Processing of News. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL '00)*, pages 69–76, 2000.

[23] B. Martins, H. Manguinhas, and J. Borbinha. Extracting and Exploring the Geo-temporal Semantics of Textual Resources. In *Proceedings of the IEEE International Conference on Semantic Computing (ICSC '08)*, pages 1–9, 2008.

[24] P. Mazur and R. Dale. The DANTE Temporal Expression Tagger. In *Proceedings of the 3rd Language and Technology Conference (LTC '09)*, pages 245–257, 2009.

[25] P. Mazur and R. Dale. WikiWars: A New Corpus for Research on Temporal Expressions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP '10)*, pages 913–922, 2010.

[26] M. Pasca. Towards Temporal Web Search. In *Proceedings of the 2008 ACM Symposium on Applied Computing (SAC '08)*, pages 1117–1121, 2008.

[27] J. Pustejovsky, J. M. Castaño, R. Ingria, R. Sauri, R. J. Gaizauskas, A. Setzer, G. Katz, and D. R. Radev. TimeML: Robust Specification of Event and Temporal Expressions in Text. In *Proceedings of the AAAI Spring Symposium on New Directions in Question Answering*, pages 28–34, 2003.

[28] J. Pustejovsky, P. Hanks, R. Sauri, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, and M. Lazo. The TIMEBANK

Corpus. In *Proceedings of Corpus Linguistics Conference*, pages 647–656, 2003.

[29] A. Qamra, B. Tseng, and E. Chang. Mining Blog Stories Using Community-based and Temporal Clustering. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM '06)*, pages 58–67, 2006.

[30] S. Schockaert1, D. Ahn, M. De Cock, and E. Kerre. Question Answering with Imperfect Temporal Information. In *Proceedings of the 7th Conference on Flexible Query Answering Systems (FQAS 06)*, pages 647-658, 2006.

[31] F. Schilder and C. Habel. From Temporal Expressions to Temporal Information: Semantic Tagging of News Messages. In *Proceedings of the Workshop on Temporal and Spatial Information Processing (TASIP '01)*, pages 65–72, 2001.

[32] M. Spaniol, D. Denev, A. Mazeika, G. Weikum, and P. Senellart. Data Quality in Web Archiving. In *Proceedings of the 3rd Workshop on Information Credibility on the Web (WICOW '09)*, pages 19–26, 2009.

[33] B. Shaparenko, R. Caruana, J. Gehrke, and T. Joachims. Identifying Temporal Patterns and Key Players in Document Collections. In *Proceedings of the IEEE ICDM Workshop on Temporal Data Mining: Algorithms, Theory and Applications (TDM '05)*, pages 165–174, 2005.

[34] J. Strötgen and M. Gertz. HeidelTime: High Quality Rule-based Extraction and Normalization of Temporal Expressions. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval '10)*, pages 321-324, 2010.

[35] J. Strötgen and M. Gertz. TimeTrails: A System for Exploring Spatio-Temporal Information in Documents. In *Proceedings of the 36th International Conference on Very Large Data Bases (VLDB '10)*, pages 1569–1572, 2010.

[36] J. Strötgen, M. Gertz, and P. Popov. Extraction and Exploration of Spatio-temporal Information in Documents. In *Proceedings of the 6th Workshop on Geographic Information Retrieval (GIR '10)*, pages 1–8, 2010.

[37] R. Swan and J. Allan. TimeMine: Visualizing Automatically Constructed Timelines. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '00)*, page 393, 2000.

[38] N. UzZaman and J. Allen. TRIPS and TRIOS System for TempEval-2: Extracting Temporal Information from Text. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval '10)*, pages 276–283, 2010.

[39] M. Verhagen, R. Gaizauskas, F. Schilder, M. Hepple, G. Katz, and J. Pustejovsky. SemEval-2007 Task 15: TempEval Temporal Relation Identification. In *Proceedings of the 4th International Workshop on Semantic Evaluation (SemEval '07)*, pages 75–80, 2007.

[40] M. Verhagen and J. Pustejovsky. Temporal Processing with the TARSQI Toolkit. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling '08)*, pages 189–192, 2008.

[41] M. Verhagen, R. Sauri, T. Caselli, and J. Pustejovsky. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval '10)*, pages 57–62, 2010.

[42] M. T. Vicente-Díez, J. Moreno-Schneider, and P. Martínez. UC3M System: Determining the Extent, Type and Value of Time Expressions in TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval '10)*, pages 329–332, 2010.

[43] G. Weikum and M. Theobald From Information to Knowledge: Harvesting Entities and Relationships from Web Sources. In *Proceedings of the 29th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems of Data (PODS' 10)*, pages 65–76, 2010.

What is the Temporal Value of Web Snippets?

Ricardo Campos
LIAAD – INESC Porto, LA
Centre for Human Language Technology and Bioinformatics,
University of Beira Interior,
Polytechnic Institute of Tomar, Portugal
ricardo.campos@ipt.pt

Gaël Dias
Centre for Human Language Technology and Bioinformatics
University of Beira Interior,
Portugal
ddg@di.ubi.pt

Alípio Mário Jorge
LIAAD – INESC Porto, LA
DCC - FCUP
University of Porto, Portugal
amjorge@fc.up.pt

ABSTRACT

The World Wide Web (WWW) is a huge information network from which retrieving and organizing quality relevant content remains an open question for mostly all implicit temporal queries, i.e., queries without any date but with an underlying temporal intent. In this research, we aim at studying the temporal nature of any given query by means of web snippets or web query logs. For that purpose, we conducted a set of experiments, which goal is to assess the percentage of web snippets or queries (in query logs) having temporal features, thus checking whether they are a valuable source of data to help on inferring the temporal intent of queries, namely implicit ones. Our results show that web snippets, as opposed to web query logs, are an important source of concentrated information, where time clues often appear. As a consequence, they can be particularly useful to identify and understand "on-the-fly" the implicit temporal nature of queries in the context of ephemeral clustering.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Search Process, Query formulation.*

General Terms

Measurement, Experiments.

Keywords

Temporal Information Retrieval, Implicit and Explicit Temporal Queries, Temporal Query Classification.

1. INTRODUCTION

Time is everywhere in the World Wide Web causing Temporal Information Retrieval (T-IR) to be increasingly involved in recent years in a systematic research process by the IR community. Notwithstanding this, few works have fully used temporal information for exploration and search purposes [3]. The lack of such an approach from major search engines, with the exception of Google[1], prevents on the one hand, users from being aware of possible historical perspectives of given subjects, and on the other hand, the modeling of user queries according to a specific period over time, potentially causing a loss in accuracy and recall. From a query point of view, this is mainly due to the fact that systems do not infer temporal intents expressed by users in a query. From the document point of view, although time clues such as dates may be found in the texts, they are usually not taken into account in the representation and indexing processes. The main reason is certainly due to the difficulties that exist to correlate temporal information to topics in documents.

Understanding the temporal intent of documents and queries is therefore of the utmost importance to produce high quality information retrieval systems. In our research, we aim at developing a methodology able to explicitly time-stamp temporal implicit queries such that temporal disambiguation can be reached "on-the-fly". This work takes place in the context of ephemeral clustering, specifically within our meta-search engine VipAccess[2] In particular, we intend to develop a language independent solution. Thus, we will specifically focus on the identification and extraction of year dates. Our framework is based on web content analysis, rather than on metadata information. As claimed by [7] this is an interesting future research direction, for which there is not a clear solution yet.

This paper is the result of part of this research. Our main purpose here is to study whether web snippets are a valuable source of data to help inferring the temporal intents of queries, either implicitly or explicitly formulated. Since results will be performed "on-the-fly", we need to adopt a web content analysis approach over the set of *k*-top web snippets retrieved, as opposed to an analysis over full web pages. In parallel with what has been done by [21], we also analyzed the temporal value of web query logs. We considered the possibility of using them for temporal query understanding purposes, but we came to the conclusion that beyond being highly dependent on the user own intents, web query logs are a particular hard temporal reference collection to access, outside the big industrial labs, and do not easily deal with query ambiguity [19].

To the best of our knowledge, this is the first work towards a comprehensive data analysis having web snippets as a data source. Indeed, [2] used them, but in another line of research, namely the construction of a time-centered snippet that highlights temporal information. Results obtained from our experiments are promising. They show that regardless of the implicit temporal nature of the query, web snippets contain a broad range of temporal information that can be used as a valuable source to help on inferring the temporal intent of queries, either implicitly or explicitly formulated.

However, one possible drawback of our research, is that web snippets are computed by search engines, which we do not control. As a consequence, basing our system upon results generated by a black box may prevent from obtaining a clear picture of the temporal values of web snippets. Nevertheless, [28] proved in the context of ephemeral clustering that web snippets are likely to provide the correct clustering of documents as they

[1] With its timeline tool incorporated within Google.com, although without any details about its architecture.

[2] http://hultig.di.ubi.pt/vipaccess [7th February, 2011].

TWAW 2011, March 28, 2011, Hyderabad, India.

embody the excerpts of documents mostly related to the query terms. Given this, we think that our analysis can serve as a good approximation of the real situation. In particular, we will base our study on three different search engines: Goggle[3], Yahoo![4] and Bing[5].

The remainder of this paper is organized as follows. In Section 2, we review temporal data reference collections commonly used in the most varied activities of T-IR. We particularly emphasize data collections of web snippets and web query logs. In Section 3, we present our experiments based on a web snippet data set and a well-known web query log. In particular, we detail each data set and introduce the methodology to extract the temporal information. Then, in Section 4, we intend to assess the temporal value of web snippets and web query logs. In particular, we first analyze the temporal intents of web snippets and their use to date implicit queries. In a second experiment, we analyze explicit temporal data in web query logs. Both results are then examined in Section 5 and we conclude this paper in Section 6 with some final remarks.

2. TEMPORAL DATA COLLECTIONS

Time can be expressed in a number of different forms, depending on how the temporal intent is defined. In queries, for example, they tend to occur by means of explicit (e.g., *Football World Cup 2010*) or implicit temporal intents (e.g., *Football World Cup*). Usually, this information is stored in web query logs, which are a flat set of files that record the activity registered in a server in different ways: (1) from an infrastructural perspective (e.g., date and time of the request) or (2) from the analysis of information use (e.g., user search intentions: *CHI 2011 vs. CHI*). Due to its private nature, some collections were made publicly available for research purposes. One of the most known is the AOL collection consisting of 21,011,240 queries collected from 650.000 users over three months (01 March, 2006 - 31 May, 2006) of activity within the AOL search engine[6]. More recently, Microsoft released two large scale datasets for research purposes on learning to rank [26]. The first one consists of 30,000 queries and the later of 10,000 queries.

Temporal expressions can also be found in a number of different types of documents such as web pages, web snippets or news articles, following an explicit (e.g., *WWW 2011*), implicit (e.g., *SIGIR*) and relative temporal intent (e.g., *in the next month*). Web documents are one of the most used sources for research purposes, with some publicly available datasets such as the Clueweb09 [8]. Compared to this data source, web snippets are a simple small section of text that provides an easy way to quickly represent information, forming a very interesting set of data where dates, especially in the form of years, often appear. Both, are usually used for content analysis purposes, clearly in the opposite direction of news document collections, where the focus is not so much on content but on metadata information, namely time-stamped documents annotated with the date of creation or publication (e.g., Reuters news stories [23]; TREC corpora [20]; TDT corpus [18]; AQUAINT-2 collection [27]; TimeBank 1.2 Corpus [22]; The New York Times Annotated Corpus [24]; ACE Time Normalization (TERN) 2004 English Training Data [12]; mostly accessible through the LCD web site [17]).

3. EXPERIMENTAL SETUP

To understand the temporal value of web snippets and web query logs, we considered the use of two independent datasets. We describe each one in the following sections along with the rule-based model used to automatically identify dates within each of them.

3.1 Data Sets

For our experiments, we rely on two datasets: a series of web snippets and associated titles and URLs obtained for the execution of 465 and 450 queries executed in December 2010 and a large-scale query log dataset belonging to 2006. Both are fully available for download[7], together with all the information produced during the execution of the experiments.

3.1.1 Web Snippets Data Set

To construct our first data set, we executed a set of queries based on our meta-search engine VipAccess excluding Google from its search interface. At first sight, it may seem strange not to use Google, but there are two main reasons for that: (1) web snippets have recently began to come together with some metadata information (e.g., blog post dates), which would introduce noise in our study and (2) Google has officially deprecated its web search API in November 2010, limiting the number of requests that can be executed per day.

The selection of the queries was made from Google Insights for Search[8], which registers the hottest queries performed worldwide. We manually selected a total of 540 queries from the period between January 2010 and October 2010, as a result of an individual selection of 20 queries for each of the 27 pre-defined categories. We were left with 465 queries, including some temporally explicit ones, after removing duplicates. We then created a reduced version of 450 queries, consisting of only the implicit ones. Each query is then executed on our meta-search engine VipAccess defined to retrieve a set of 20 and 100 triple items <snippet, title, url> so as to observe any variations that may exist due to different amounts of retrieved data. In practice this represents 40 and 200 results, given our meta-search engine runs over two search engines.

Given this, we ended up by constructing three different collections[9] (*Q465R20*, *Q450R20* and *Q450R100)*, each one gathering a different number of retrieved items (see Table 1). Most of the queries belong to the categories of Internet (12.69%), Computer & Electronics (9.89%) and Entertainment (7.96%).

3.1.2 Web Query Log Data Set

Our second data set, labeled Q601, is a set of queries extracted from the AOL collection. It consists of 21,011,240 queries, 10,154,742 if we only consider single entries. From this collection, we automatically selected only those queries with explicit temporal intent. So, we ended up with 143,590 queries, from which we selected a representative sample of n=601 queries with a maximum tolerated average sampling error of E=4% for a

[3] http://www.google.com [7th February, 2011].

[4] http://www.yahoo.com [7th February, 2011].

[5] http://www.bing.com [7th February, 2011].

[6] http://search.aol.com [7th February, 2011].

[7] http://www.ccc.ipt.pt/~ricardo/software [7th February, 2011].

[8] http://www.google.com/insights/search [7th February, 2011].

[9] Where *Q* means the number of queries to run and *R* the number of results to retrieve for each query.

confidence interval of 95% (see Equation (1) based on [6]) where Z_p is the *p*-th quantile of the normal distribution, which in this case is equal to 1.96.

$$n = \frac{z_p}{4E^2}, \quad (1)$$

Each query was then manually classified into a set of 29 pre-defined categories (see [9] for a detailed description of each one). Most of the queries belong to the categories of Automotive (21.96%), Entertainment (9.48%), Sports (8.15%), Business & Economics (5.99%) and News & Events (5.16%).

3.2 Rule-based Model

The identification of dates, either within web queries or web documents, is probably the most recognized area within this recent research field, with TempEx[10], GUTime[11] and ANNIE[12] taking the lead. However, given that we only aim at detecting dates in the form of numerical years in order to meet an independent language solution, there is no need to use any of these methodologies, which are all language-dependent. As a consequence, we ended up by defining our own rule-based model. Similarly to the works abovementioned, we considered numbers in the interval [1000..2099] that satisfy some specific patterns, such as *yyyy*, *yyyy-yyyy*, *yyyy/yyyy*, *mm/dd/yyyy*, *mm.dd.yyyy*, *dd/mm/yyyy* and *dd.mm/yyyy*.

Just by using this simple rule-based method, we achieved results of almost 96% accuracy in the detection of dates within documents, namely within titles and web snippets (see Table 1). However, the application of these rules within URLs results in the return of some noisy information, which is no big surprise. Overall, false dates usually tend to occur in the response of queries belonging to the categories of Internet (e.g., *1600 YouTube Videos*), Computer & Electronics (e.g., *1024 x 768)*, Games & Toys (e.g., *1000 games*) and Food & Drink (e.g., *1001 recipes*). To reach this conclusion, we manually went through each of the items of the three collections *Q465R20, Q450R20 and Q450R100* and checked whether they were correctly labeled or not.

We performed the exact same process for the identification of dates within queries (*Q601*). Results obtained in this case show an accuracy of 86%. Again false dates tend to occur mostly within the category of Computer & Electronics (e.g., *hp 1430*).

Of course, we could improve our rule-based model by limiting the appearance of some noisy information. One way to do this would be to include some terms like *year*, or some temporal prepositions such as *in* or *between*. However, this would go in the opposite direction of our language independent goal. As such, we should try to find alternative solutions in future work. One of the possibilities is to model, with some degree of confidence, the relationship existing between the topics and the possible dates.

4. METHODOLOGY

Based on the web snippet data set, we performed a set of experiments to analyze their temporal value and their potential usefulness in the process of dating temporal implicit queries. Additionally, we also intended to study the temporal value of web query logs, particularly the relationship existing between queries and dates, i.e. temporal explicit queries. We believe this information will serve to improve our temporal knowledge with respect to the user query intents, either implicitly or explicitly defined. We detail each framework in the following sections.

4.1 Temporal Value of Web Snippets

In this first experiment, we are particularly interested in studying the existence of temporal information within web snippets, and to check if such information can be used to temporally classify implicit queries. We follow a twofold approach: (1) we collect the necessary data and (2) we classify queries with respect to their temporal nature. We summarize the overall evaluation framework in Figure 1.

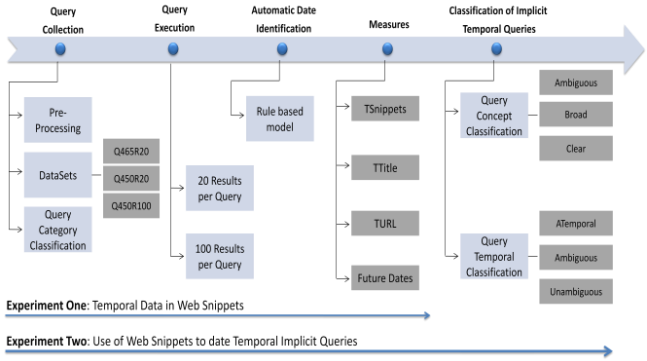

Figure 1: Web Snippets Framework.

4.1.1 Query Execution

The first step of our framework, after minor adjustments (pre processing phase and query category classification), is to process the three collections (*Q465R20*, *Q450R20* and *Q450R100*) on our meta-search engine, which defines the web snippet data set.

4.1.2 Automatic Date Identification

Upon the retrieved results, particularly over each triple item <snippet, title, url>, we ran our self-defined rule-based model in order to mark dates expressed by means of numerical patterns, particularly year dates (see Figure 2).

Figure 2: Title, Web Snippet and URL returned to the query *Alice in Wonderland*.

Each item is individually marked through an XML schema. The set of files is available for download[13].

4.1.3 Evaluation Metrics

In order to better understand and determine the temporal value of each item, we defined three basic measures taking into account the query *q*. They are represented by the functions *TSnippets(q)*, *TTitle(q)* and *TUrl(q)*. *TSnippets(q)* is computed as the ratio between the number of snippets returned with dates divided by the total number of snippets returned by our meta search engine (see Equation (2)). *TTitle(q)* and *TUrl(q)* are computed similarly and respectively defined in Equations (3) and (4).

[10] http://fofoca.mitre.com [7th February, 2011].

[11] http://www.timeml.org/site/tarsqi/modules/gutime/ [7th February,2011].

[12] http://www.aktors.org/technologies/annie/ [7th February, 2011].

[13] http://www.ccc.ipt.pt/~ricardo/software [7th February, 2011].

$$TSnippets(q) = \frac{\#\ Snippets\ Retrieved\ With\ Dates}{\#\ Snipptes\ Retrieved} \quad (2)$$

$$TTitle(q) = \frac{\#\ Titles\ Retrieved\ With\ Dates}{\#\ Titles\ Retrieved} \quad (3)$$

$$TUrl(q) = \frac{\#\ Urls\ Retrieved\ With\ Dates}{\#\ Urls\ Retrieved} \quad (4)$$

4.1.4 Temporal Classification of Implicit Queries

Query temporal classification is a particular hard task in the case of implicit temporal queries (e.g., *WWW*, *Scorpions*), in the sense that temporal information is not available, at least in a direct way. One possible solution to estimate the temporal value of a query is to compare it with complementary knowledge sources, such as web snippets or web queries logs. In this experiment, we aim at classifying temporal implicit queries with web snippets. A preliminary step however is involved. We need to first classify the query with regard to its conceptual ambiguity such that each different meaning[14] may have a different temporal dimension. To this purpose we follow the approach of [25], which defines three types of concept queries (see Figure 3), reducing this to a simple classification problem, with two possible classes: A (ambiguous queries) and $\overline{A}$ (broad or clear query).

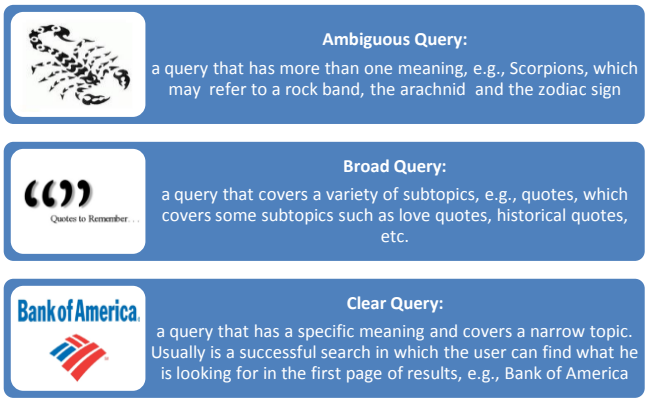

Figure 3: Taxonomy of ambiguous queries in concept. Adapted from [25].

We adopted the following methodology. First, we attempt to confirm if the query is ambiguous in terms of concept, i.e., if it belongs to class A. We use the disambiguation Wikipedia feature, to confirm, whether or not the query under study has more than one meaning. Second, if we conclude that the query has a single meaning, thus belonging to class $\overline{A}$, we attempt to confirm whether the query is broad or clear. For that purpose, we run each query on our meta-search engine, which clusters web page results through an ephemeral clustering process and retrieves its possible different sub-topics. Then, based on the assumption that only clear concept queries may be temporally classified, we introduce (see Figure 4), as in [14], three possible temporal classes: ATemporal, i.e. queries not sensitive to time (e.g. *make my trip*), Temporal Unambiguous, i.e. queries that take place in a very concrete time period (e.g. *Haiti Earthquake* or *BP Oil Spill*) and Temporal Ambiguous, i.e. queries with multiple instances over time (e.g. *World Cup* or *Oil Spill*).

[14] *WWW* may be the web conference, but also the World Wide Web itself. *Scorpions* can be associated to the rock band, the arachnid and the zodiac sign.

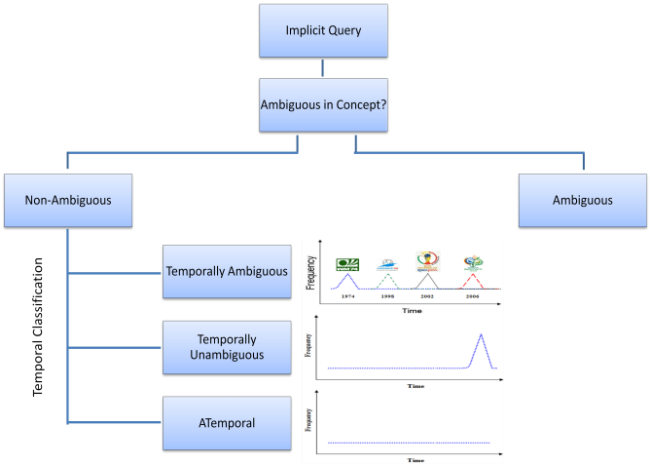

Figure 4: Temporal classification of clear concept queries.

Each query is classified into one of these three categories based on the temporal value of the triple items <snippet, title, url> retrieved. We call this temporal value temporal ambiguity. To compute it, we define (see Equation (5)) a weighted average of the *TSnippet(.)*, *TTile(.)* and *TUrl(.)* functions. Given the fact that dates occur in a different proportion in any of the items <snippet, title, url>, we value each differently through ω_f, where f is the function regarding the corresponding item. Considering the average measures obtained for *TSnippet(.)*, *TTile(.)* and *TUrl(.)*, for each of the collections (see Table 1) we define ω_f as 66.10% for *TSnippet(.)*, 20.75% for *TTile(.)* and 13.15% *TUrl(.)* for the *Q450R20* collection and respectively 50.91%, 18.14% and 30.95%, for the *Q450R100* collection.

$$TA(q) = \sum_{f \in I} \omega_f \,.\, f(q), I = \{TSnippet(.), TTitle(.), TUrl(.)\} \quad (5)$$

Based on the *TA(.)* function, we aim at classifying each query as A (ATemporal query) or $\overline{A}$ (Temporal Ambiguous or Temporal Unambiguous query). In particular, we classify each query as ATemporal if it has a *TA(.)* value below 10%. Otherwise, we conclude that the query belongs to class $\overline{A}$. In this case, we just have to decide whether the query is temporally ambiguous or not, based on the fact that it is associated to more than one year.

4.2 Temporal Value of Web Query Logs

Our purpose in this second experiment, based on the *Q601* collection, is to complement our knowledge about temporal information by understanding the explicit relationships existing between queries and dates. We are particularly interested in studying the temporal value of web query logs, but this may also prove interesting to understand two phenomena: (1) are users interested in future dates when looking for a given subject? and (2) what is the type of information they are seeking when issuing a query together with a date? In Figure 5, we summarize the overall evaluation framework. The first step is to use our rule-based model in order to automatically identify years within queries. Then, we manually identify false and future dates, and classify each of the queries in a set of 29 pre-defined categories listed in [9].

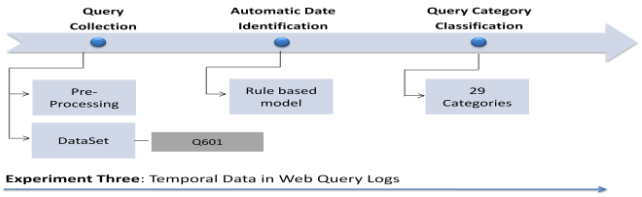

Figure 5: Web Query Logs Framework.

5. DISCUSSION

In this section, we discuss the various issues of our temporal web mining proposal over our web snippet data set and web query log. Our aim is to evaluate the extent and usefulness of temporal information in both collection. Questions discussed include, for example, to what extent temporal information in a result web snippet can indicate the temporal interest of the information needed by the user. In another strand of this research, we analyze the temporal value of web query logs and discuss to what extent the lack of a temporal value prevents the use of this kind of data source to work as knowledge base for query understanding purposes.

5.1 Temporal Value of Web Snippets

5.1.1 Temporal Data in Web Snippets

In this first experiment, we are particularly interested in studying the existence of temporal information within web snippets, namely within triple items <snippet, title, url>. We are particularly focused on extracting year dates, which are a kind of temporal information that often appears in this type of collection.

To this end, we conducted three tests in December 2010, *Q465R20*, *Q450R20* and *Q465R100*. Each corresponding query was executed on our meta-search engine VipAccess. Thus, we collected 16,648 triple items for the first test, 16,129 for the second one, and 62,842 for the final one. Then, upon these web page results, particularly over each triple item retrieved, we ran our rule-based model so as to automatically identify dates in the form of numerical years. We achieved results at about 96% of accuracy within web snippets, 98% within titles, but significantly less in the case of URLs with the worst value equals to 75%. To get these values we went through each of the items of the three collections, manually correcting each false positive occurrence.

Upon these labeled examples, we then computed the corresponding metrics *TSnippets(.)*, *TTitle(.)* and *TUrl(.)*. Obtained results (see Table 1) are according to our expectations. On average 10% of the web snippets retrieved for *Q450R20* and *Q450R100* collections have a temporal feature. This value is significantly higher for *Q465R20* collection, as it includes 15 explicit temporal queries (e.g., *hairstyles 2010*), which obviously implies the retrieval of a larger range of outcomes than usual. The occurrence of temporal features is particularly evident in the case of web snippets, but still significant in the case of titles and URLs.

We can also note that the differences between *Q450R20* and *Q450R100* collections are minimal in terms of ratio. The only exception is with *TUrl(.)*.In fact, retrieving 20 results as opposed to 100 show a decrease in the number of identified dates. This clearly shows that the first web snippets retrieved by search engines do not tend so much to include years in their web snippets. A detailed analysis within the whole set of results led us to conclude that this last collection will obviously retrieve a large range of different dates, which may be useful for a full understanding of the temporal value of a given query.

Another important remark is that the occurrence of dates within triple items <snippet, title, url> often occurs with more than one date, with values close to 23% in the case of web snippets, but in a much smaller scale for the remaining dimensions (see Figure 6).

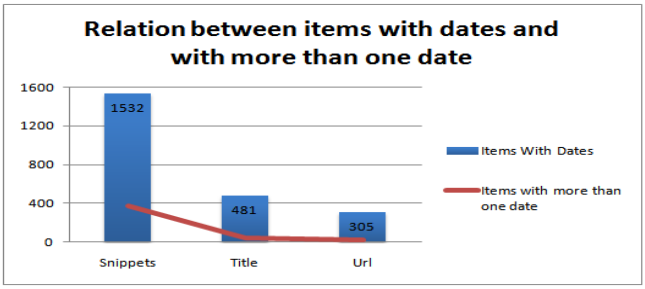

Figure 6: *Q450R20* items marked with more than one date.

It also becomes clear when observing Figure 7, that from 2003 onwards the existence of dates occur in a more intense manner, with a high peak in the period of 2008-2010 and that future dates, first introduced by [5] in 2005, also occur to a considerable extent, particularly in the case of titles, where 20% of the results retrieved have a future temporal intent. More recently [13] investigated the distribution of such information in documents and in historical document collections.

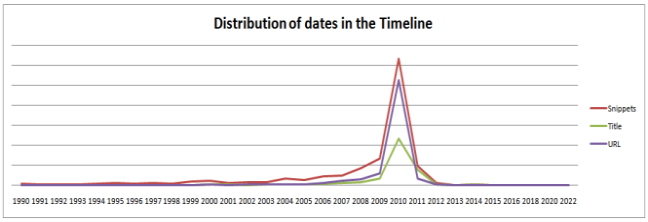

Figure 7: Distribution of dates in the timeline for *Q450R100* experiment.

Overall, we can conclude that dates occur more frequently in response to queries belonging to the categories of Dates (e.g. *calendar*), Sports (e.g., *football*), Automotive (e.g., *dacia duster*), Society (e.g., *baby*) and Politics (e.g., *Barack Obama*).

An individual analysis of one or two queries may also show us some interesting results. For instance, the implicit query *Tour de France* clearly formulated with an inherent temporal nature has a value of 77.78% for *TSnippet(.)* in *Q450R20*, with dates ranging at an annual basis, from the far distant year of 1903 to the current year 2011. Another example is the query *Toyota Recall* (see Figure 8), which despite the occurrence of other events over time may benefit of a clear temporal positioning because of a more distinct peak of occurrences between 2010-2011.

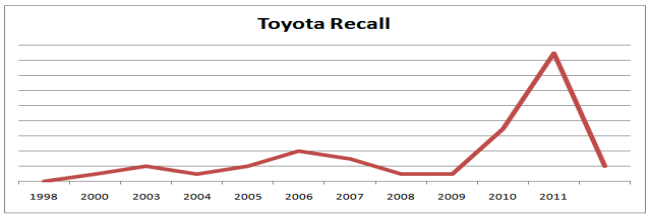

Figure 8: Graphical representation of the query *Toyota Recall* in the period of 1998-2011.

In Table 1 we can observe a summary of the results for the different experiments.

Table 1. Summary of Results for *Q465R20, Q450R20* and *Q450R100*

Tests	Web Page Results	Automatic Date Identification Accuracy		Average Measures	Future Dates
Q465R20	16,648	**WSp**	95.8%	12.4%	18.6%
		Title	97.9%	5.69%	13.8%
		URL	85.1%	4.26%	9.64%
Q450R20	16,129	**WSp**	94.3%	9.50%	6.5%
		Title	95.8%	2.98%	18.9%
		URL	75.0%	1.89%	8.8%
Q450R100	62,842	**WSp**	93.1%	9.19%	7.9%
		Title	95.3%	3.27%	19.7%
		URL	87.4%	5.59%	5.7%

Both tests show that documents in the web, particularly web snippets, are full of temporal expressions. However, they are not always exploited to increase the quality of the retrieval process.

5.1.2 Dating Implicit Queries using Web Snippets

Given the temporal value of web snippets, we aim at understanding if this temporal information can be used to automatically disambiguate query terms, namely implicit temporal queries. We rely on the *Q450R20* and *Q450R100* collections. Our approach is twofold. First, we classify each query with regard to its concept ambiguity. Final results (see Table 2) show that most of the queries are ambiguous in concept, although there is a large set of clear queries, which do not offer any doubt in terms of their meaning and a small set of broad queries.

Table 2. Classification of Queries both in Concept and Temporal meaning for the *Q450R20* collection

Conceptual Classification	Number Queries	Temporal Classification	Number Queries	%
Ambiguous	220			
Clear	176	ATemporal	132	75%
		Ambiguous	40	23%
		Unambiguous	4	2%
Broad	54			

We then temporally classified each of the 176 clear concept queries according to its temporal ambiguity *TA(.)* value. We classified each query as ATemporal if it had a *TA(.)* value below 10% (see Figure 9) and as temporally ambiguous or unambiguous, otherwise.

1	Results for **20 Results per Query**						
2	Query	AmbQuery	TempQuery	tsnippets	ttitle	turl	Temporal Ambiguity
56	tattoos for girls	Clear Query	ATemporal	2,86%	0,00%	0,0%	1,89%
190	bed bugs	Clear Query	ATemporal	0,00%	0,00%	0,0%	0,00%
201	oil spill	Clear Query	TempAmbiguous	26,47%	17,65%	0,0%	21,16%
204	bp oil spill	Clear Query	TempUnambiguous	15,15%	9,09%	0,0%	11,90%
340	love quotes	Clear Query	ATemporal	0,00%	0,00%	0,0%	0,00%
452	make my trip	Clear Query	ATemporal	0,00%	0,00%	0,0%	0,00%
453							
1	Results for **100 Results per Query**						
2	Query	AmbQuery	TempQuery	tsnippets	ttitle	turl	Temporal Ambiguity
56	tattoos for girls	Clear Query	ATemporal	4,3%	0,00%	0,72%	1,89%
190	bed bugs	Clear Query	ATemporal	2,4%	0,00%	1,57%	0,00%
201	oil spill	Clear Query	TempAmbiguous	19,8%	6,87%	9,16%	14,19%
204	bp oil spill	Clear Query	TempUnambiguous	15,3%	8,03%	18,98%	15,13%
340	love quotes	Clear Query	ATemporal	0,8%	0,00%	0,76%	0,00%
452	make my trip	Clear Query	ATemporal	3,0%	0,75%	0,75%	0,00%

Figure 9: Temporal Ambiguity results for *tattoos for girls, bed bugs, oil spill, bp oil spill, love quotes* and *make my trip*.

The final analysis (see Table 2) based on the *Q450R20* collection show that of the total number of clear concept queries, 25% have implicit temporal intent, of which 23% are temporal ambiguous queries and 2% unambiguous. The remaining 75% are ATemporal queries i.e. queries for which no temporal intent is inherent. Results are very similar for *Q450R100* collection.

These values contrast with those presented in [19] who, based on web query logs, estimated that about 7% of all queries have an implicit temporal nature.

5.2 Temporal Value of Web Query Logs

In this experiment, we intend to analyze the temporal value of web query logs. In particular, we are interested in seeking for explicit temporal queries e.g. *Iraq War 1991* or *World Cup 2000*. For this purpose, we ran our rule-based model over the overall AOL queries log data set. We ended up with 143,590 queries. This represents only 1.41% of the entire collection, which is in line with the value of 1.5% presented by [21]. However, if we consider there are some false positives, due to the set of dates wrongly marked by our rule-based model, we may end up with an even lower value.

To better estimate these results, we decided to analyze the *Q601*collection. Each query was manually classified according to one of two categories: real date or false date. A further analysis of the results allowed us to conclude that 14.14% of the sample i.e. 87 queries were false positives. If we generalize these results to the whole temporal explicit queries set, then we can conclude that instead of having 143,590 implicit temporal queries we should only have 123,286 queries, causing, albeit not significantly, the reduction of the initial value of 1.41% to 1.21%.

Based on this study, we can conclude that dates are seldom used by the user to express his intents. Moreover, web query logs do not provide a data source adapted to concept disambiguation, although some studies tend to prove the contrary [19]. Besides, they are extremely hard to access outside the big industrial labs and highly dependent on the user own intents. We can particularly think in the two following illustrative cases: (1) queries that have never been typed, thus not existing in the web search log e.g. *Blaise Pascal 1623* (his year birth date) or (2) less year qualified queries that may be as relevant as the most frequents ones.

To better understand this problem, we can observe Figure 10, which clearly shows that despite significant peaks along the timeline, there is a lack of queries outside the period 2000-2006.

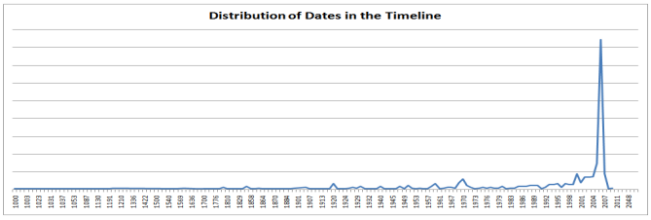

Figure 10: Distribution of dates in the timeline for *Q601* experiment.

Another interesting remark to note is that, notwithstanding a decrease from 2006 onwards (we recall that this collection is from 2006), future dates still represent 3.49% of the sample collection and that users are more interested in queries belonging to the categories of Automotive, Entertainment and Sports when queries are explicit in its temporal intent.

6. Conclusion

In recent years, time has been gaining an increasing importance in Information Retrieval in a large number of sub-areas. However, and despite the fact that the documents are full of temporal expressions, existing IR systems still do not exploit this information. On the one hand, this prevents systems to model the search process according to specific periods of time. On the other hand, it prevents the user to have an historical perspective of the results.

Inferring the user intentions and the period the user has in mind may therefore play an extremely important role in the possible improvements of IR systems by adding the temporal dimension, practically nonexistent until now. Aware of this, some works have been emerging in the last few years, yet most only use temporal information extracted within metadata, particularly from web news documents [7] [10] [14], [15], and only a few, [1] [3] [4] and [19] have attempted to extract temporal information within the contents of web documents, although the latter with the help of web query logs. Simultaneously other studies have appeared related with recency ranking, as opposed to user intentions understanding. In this regard [11], [16], [29]suggest to rank documents by relevance taking into account its freshness.

In this paper, we showed that query understanding is possible through the use of web snippets. Our experiments, strictly dependent on the value of the temporal information retrieved for a given query, show that web snippets are a very rich data source, where dates, especially years, often appear, and that they can embody a very important data source in query understanding, thus allowing "on-the-fly" temporal disambiguation for real-time IR systems. Moreover, working with web snippets may afford faster processing.

Our approach, however, may be negatively influenced by the production of the results of existing search engines considered in our meta-search engine, which we do not control. Such can cause for example, some documents to be discriminated from the very outset, preventing a broader temporal analysis. An elucidative example of this drawback is given by the query *Iraq War*, which if issued in any of the major search engines, will mostly retrieve results with regard to the conflict of 2003, as opposed to 1991 despite the fact the web is full of results concerning *Iraq War 1991*. Given this, we need to evaluate the feasibility of developing a search engine, albeit of a small scale, which will also enable us to compare a full text analysis approach with a web snippet based one.

7. ACKNOWLEDGMENTS

This research was partially funded by a grant (Reference: SFRH/BD/63646/2009) under the Portuguese Foundation for Science and Technology. A special thank to the four reviewers for the quality of their comments, which were extremely important to significantly improve the quality of this article.

8. REFERENCES

[1] Alonso, O., & Gertz, M. (2006). Clustering of Search Results using Temporal Attributes. *In SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 597 - 598). Seattle, Washington, USA. August 6 - 11: ACM Press.

[2] Alonso, O., Baeza-Yates, R., & Gertz, M. (2009). Effectiveness of Temporal Snippets. *In WSSP2009: Proceedings of the Workshop on Web Search Result Summarization and Presentation associated to WWW2009: 18th International World Wide Web Conference.* Madrid, Spain. April 20 - 24: ACM Press.

[3] Alonso, O., Gertz, M., & Baeza-Yates, R. (2009). Clustering and Exploring Search Results using Timeline Constructions. *In CIKM 2009: Proceedings of the 18th International ACM Conference on Information and Knowledge Management.* Hong Kong, China. November 2 - 6: ACM Press.

[4] Arikan, I., Bedathur, S., & Berberich, K. (2009). Time Will Tell: Leveraging Temporal Expressions in IR. *In WSDM 2009: Proceedings of the 2nd ACM International Conference on Web Search and Data Mining.* Barcelona, Spain. February 09 - 12: ACM Press.

[5] Baeza-Yates, R. (2005). Searching the Future. In S. Dominich, I. Ounis, & J.-Y. Nie (Ed.), *MFIR2005: Proceedings of the Mathematical/Formal Methods in Information Retrieval Workshop associated to SIGIR 2005: 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Salvador, Brazil. August 15 - 19: ACM Press.

[6] Barbetta, P. A., Reis, M. M., & Bornia, A. C. (2004). *Estatística para Cursos de Engenharia e Informática.* Lisboa: Atlas.

[7] Berberich, K., Bedathur, S., Alonso, O., & Weikum, G. (2010). A Language Modeling Approach for Temporal Information Needs. In C. Gurrin, Y. He, G. Kazai, U. Kruschwitz, S. Little, T. Roelleke, et al. (Eds.), *In Lecture Notes in Computer Science - Research and Advanced Technology for Digital Libraries, ECIR 2010: 32nd European Conference on Information Retrieval* (Vol. 5993/2010, pp. 13 - 25). Milton Keynes, UK. March 28 - 31: Springer Berlin / Heidelberg.

[8] Callan, J., Hoy, M., Yoo, C., & Zhao, L. (2009, 02). *ClueWeb09 Dataset*. Retrieved 12 23, 2010, from Carnegie Mellon University: http://boston.lti.cs.cmu.edu/Data/clueweb09/

[9] Campos, R. (2011). *Analysis of Temporal Data in Explicit Temporal Queries. AOL dataset.* Available at: http://www.ccc.ipt.pt/~ricardo/software

[10] Dakka, W., Gravano, L., & Ipeirotis, P. G. (2008). Answering General Time Sensitive Queries. *In CIKM 2008: Proceedings of the 17th International ACM Conference on Information and Knowledge Management* (pp. 1437 - 1438). Napa Valley, California, USA. October 26 - 30: ACM Press.

[11] Dong, A., Chang, Y., Zheng, Z., Mishne, G., Bai, J., Zhang, R., et al. (2010). Towards Recency Ranking in Web Search. In WSDM2010: In Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (pp. 11 - 20). New York, USA. February 3 - 6: ACM Press.

[12] Ferro, L., Gerber, L., Hitzeman, J., Lima, E., & Sundheim, B. (2005). ACE Time Normalization (TERN) 2004 English Training Data v 1.0. (L. D. Consortium, Ed.) Philadelphia, USA.

[13] Jatowt, A., Kawai, H., Kanazawa, K., Tanaka, K., & Kunieda, K. (2010). Analyzing Collective View of Future, Time-referenced Events on the Web. In WWW2010: Proceedings of the 19th International World Wide Web

Conference (pp. 1123 - 1124). Raleigh, USA. April 26 - 30: ACM Press.

[14] Jones, R., & Diaz, F. (2007). Temporal Profiles of Queries. *In TOIS: ACM Transactions on Information Systems , 25(3). Article No.: 14*.

[15] Kanhabua, N., & Nørvåg, K. (2010). Determining Time of Queries for Re-Ranking Search Results. *In ECDL2010: Proceedings of The European Conference on Research and Advanced Technology for Digital Libraries*. Glasgow, Scotland. September 6 - 10: Springer Berlin / Heidelberg.

[16] Li, X., & Croft, B. W. (2003). Time-Based Language Models. *In CIKM 2003: Proceedings of the 12th International ACM Conference on Information and Knowledge Management* (pp. 469 - 475). New Orleans, Louisiana, USA. November 2 - 8: ACM Press.

[17] Linguistic Data Consortium. (1992). *Linguistic Data Consortium*. Retrieved 12 23, 23, from Linguistic Data Consortium: http://www.ldc.upenn.edu/

[18] Linguistic Data Consortium. (n.d.). *Topic Detection and Tracking*. Retrieved 12 23, 2010, from Linguistic Data Consortium: http://projects.ldc.upenn.edu/TDT/

[19] Metzler, D., Jones, R., Peng, F., & Zhang, R. (2009). Improving Search Relevance for Implicitly Temporal Queries. *In SIGIR 2009: Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 700 - 701). Boston, MA, USA. July 19 - 23: ACM Press.

[20] NIST. (2000, 08 01). *Text REtrieval Conference (TREC) Home Page*. Retrieved 08 07, 2009, from National Institute of Standards and Technologies: http://trec.nist.gov/

[21] Nunes, S., Ribeiro, C., & David, G. (2008). Use of Temporal Expressions in Web Search. In C. Macdonald, I. Ounis, V. Plachouras, I. Ruthven, & R. White (Eds.), *Lecture Notes in Computer Science - Advances in Information Retrieval, ECIR 2008: European Conference on IR Research* (Vol. 4956/2008, pp. 580 - 584). Glasgow, Scotland. 30th March - 3rd April: Springer Berlin / Heidelberg.

[22] Pustejovsky, J., Verhagen, M., Sauri, R., Littman, J., Gaizauskas, R., Katz, G., et al. (2006). TimeBank 1.2. (L. D. Consortium, Ed.) Philadelphia, USA.

[23] *Reuters Corpora @ NIST*. (2004, 11 2). Retrieved 12 23, 2010, from Text Retrieval Conference: http://trec.nist.gov/data/reuters/reuters.html

[24] Sandhaus, E. (2008). *The New York Times Annotated Corpus*. Retrieved 12 23, 2010, from Linguistic Data Consortium: http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2008T19

[25] Song, R., Luo, Z., Nie, J.-Y., Yu, Y., & Hon, H.-W. (2009). Identification of Ambiguous Queries in Web Search. *In Information Processing & Management: An International Journal , 45(2)*, 216 - 229.

[26] Tao, Q., Tie-Yan, L., Wenkui, D., Jun, X., & Hang, L. (2010, 06 16). *Microsoft Learning to Rank Datasets*. Retrieved 12 23, 2010, from Microsoft Research: http://research.microsoft.com/en-us/projects/mslr/default.aspx

[27] Vorhees, E., & Graff, D. (2008). *AQUAINT-2 Information-Retrieval Text Research Collection*. Retrieved 12 23, 2010, from Linguistic Data Consortium: http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2008T25

[28] Zamir, O., & Etzioni, O. (1998). Web Document Clustering: A Feasibility Demonstration. In SIGIR 1998: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 46 - 54). Melbourne, Australia. August 24 - 28: ACM Press.

[29] Zhang, R., Chang, Y., Zheng, Z., Metzler, D., & Nie, J.-y. (2009). Search Result Re-ranking by Feedback Control Adjustment for Time-sensitive Query. In NAACL2009: Proceedings of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies, (pp. 165 - 168). Boulder, Colorado, USA. May 31 - June 5.

Temporal Analysis for Web Spam Detection: An Overview*

Miklós Erdélyi[1,2] András A. Benczúr[1]
[1]Institute for Computer Science and Control, Hungarian Academy of Sciences
[2]University of Pannonia, Department of Computer Science & Systems Technology, Veszprém
{miklos, benczur}@ilab.sztaki.hu

ABSTRACT

In this paper we give a comprehensive overview of temporal features devised for Web spam detection providing measurements for different feature sets.

- We make a temporal feature research data set publicly available[1]. The features are based on eight UbiCrawler crawl snapshots of the `.uk` domain between October 2006 and May 2007 and use the WEBSPAM-UK2007 labels.
- We explore the performance of previously published temporal spam features and in particular the strength and sensitivity of linkage change.
- We propose new temporal link similarity based features and show how to compute them efficiently on large graphs.

Our experiments are conducted over the collection of eight `.uk` crawl snapshots that include WEBSPAM-UK2007.

Categories and Subject Descriptors

H.3 [**Information Systems**]: Information Storage and Retrieval; I.2 [**Computing Methodologies**]: Artificial Intelligence; I.7.5 [**Computing Methodologies**]: Document Capture—*Document analysis*

General Terms

Hyperlink Analysis, Feature Selection, Document Classification, Information Retrieval

Keywords

Web spam, Document Classification, Time series analysis, Hyperlink analysis

*This work was supported by the EU FP7 Project LIWA (Living Web Archives), LAWA (Large-Scale Longitudal Web Analytics) and by the grant OTKA NK 72845.

[1]`http://datamining.ilab.sztaki.hu/?q=en/downloads`

TWAW 2011, March 28, 2011, Hyderabad, India.

1. INTRODUCTION

Web spam filtering, the area of devising methods to identify useless Web content with the sole purpose of manipulating search engine results, has drawn much attention in the past years [37, 28, 27]. Although recently there seems to be a slowdown in the achievements, temporal analysis appears as a new area with several recent papers [36, 31, 17, 30, 20].

In this paper we present, to our best knowledge, the most comprehensive experimentation based on content, link as well as temporal features, both new and recently published. We compare our result with the very strong baseline of the Web Spam Challenge 2008 data set.

We extend link-based similarity algorithms by proposing metrics to capture the linkage change of Web pages over time. We describe a method to calculate these metrics efficiently on the Web graph and then measure their performance when used as features in Web spam classification. We propose an extension of two link-based similarity measures: XJaccard and PSimRank [23].

We investigate the combination of temporal and non-temporal, both link- and content-based features using ensemble selection. We evaluate the performance of ensembles built on the latter feature sets and compare our results to that of state-of-the-art techniques reported on our dataset. Our conclusion is that temporal and link-based features in general do not significantly increase Web spam filtering accuracy. However, information about linkage change might improve the performance of a language independent classifier: the best results for the French and German classification tasks of the ECML/PKDD Discovery Challenge [26] were achieved by using host level link features only, outperforming those who used all features [1].

The rest of this paper is organized as follows. After listing related results, in Section 2 we describe the features we add to the baseline set of [13] including new temporal features introduced first in this paper. In Section 3 we describe our classification framework. The results of the experiments to classify WEBSPAM-UK2007 by also relying on 7 additional crawl snapshots of the same domain can be found in Section 4.

1.1 Related Results

An excellent overview of Web spam filtering methods, both temporal and non-temporal approaches is found in [12]. When building our baseline classifier, we considered the known features as well as the classification methods used by the winners of the Web Spam Challenge 2008 [25] and the ECML/PKDD Discovery Challenge [26]. The baseline

ensemble classifier tested on two data sets is taken from our work [21].

Recently the evolution of the Web has attracted interest in defining features, signals for ranking [18] and spam filtering [36, 31, 17, 30, 20]. The earliest results investigate the changes of Web content with the primary interest of keeping a search engine index up-to-date [15, 16]. The decay of Web pages and links and its consequences on ranking are discussed in [3, 19]. One main goal of Boldi et al. [7] who collected the `.uk` crawl snapshots also used in our experiments was the efficient handling of time-aware graphs. Closest to our temporal features is the investigation of host overlap, deletion and content dynamics in the same data set by Bordino et al. [8].

Perhaps the first result on the applicability of temporal features for Web spam filtering is due to Shen et al. [36] who compare pairs of crawl snapshots and define features based on the link growth and death rate. However by extending their ideas to consider multi-step neighborhood, we are able to define a very strong feature set that can be computed by the Monte Carlo estimation of Fogaras and Rácz [23].

Another related result defines features based on the change of the content [17] who obtain page history from the Wayback Machine. They only present classification results for a selected subset of hosts and they do not compare their performance with the Web Spam Challenge 2008 results [11] as they only measure precision, recall and F-measure but not AUC (area under the ROC curve [24]). In order to be comparable with a larger set of spam detection techniques, we use the full 2,053 host Web Spam Challenge 2008 test set. In addition, we believe that AUC is more stable as it does not depend on the split point; indeed, while Web Spam Challenge 2007 used F-measure and AUC, Web Spam Challenge 2008 used AUC only as evaluation measure.

In a preliminary result [20] we suggested the applicability of Jaccard and cosine similarity metrics for capturing content change of groups of Web pages. Compared to that result, in this paper we show full-scale results of applying term-weight based temporal content features. In addition, we derive features based on the multi-step linkage similarity of Web hosts. This set of features extends the growth and death rate of [36] that we use as baseline features.

For a broader outlook, temporal analysis is also applied for splog detection, i.e. manipulative blogs with the sole purpose to attract search engine traffic and promote affiliate sites. Lin et al. [31] consider the dynamics of self-similarity matrices of time, content and link attributes of posts. They use the Jaccard similarity, a technique that we are also applying in our experiments.

2. TEMPORAL FEATURES FOR SPAM DETECTION

Spammers often create bursts in linkage and content: they may add thousands or even millions of machine generated links to pages that they want to promote [36] that they again very quickly regenerate for another target or remove if blacklisted by search engines. Therefore changes in both content and linkage may characterize spam pages.

2.1 Linkage Change

In this section we describe link-based temporal features that capture the extent and nature of linkage change. These features can be extracted from either the page or the host level graph where the latter has a directed link from host a to host b if there is a link from a page of a to a page of b.

The starting point of our new features is the observation of [36] that the in-link growth and death rate and change of clustering coefficient characterize the evolution patterns of spam pages. We extend these features for the multi-step neighborhood in the same way as PageRank extends the in-degree. The *ℓ-step neighborhood* of page v is the set of pages reachable from v over a path of length at most ℓ. The ℓ-step neighborhood of a host can be defined similarly over the host graph.

We argue that the changes in the multi-step neighborhood of a page should be more indicative of the spam or honest nature of the page than its single-step neighborhood because spam pages are mostly referred to by spam pages [14], and spam pages can be characterized by larger change of linkage when compared to honest pages [36].

In the following we review the features related to linkage growth and death from [36] in Section 2.1.1, then we introduce new features based on the similarity of the multi-step neighborhood of a page or host. We show how the XJaccard and PSimRank similarity measure can be used for capturing linkage change in Section 2.1.3 and Section 2.1.4, respectively.

2.1.1 Change Rate of In-links and Out-links

We compute the following features introduced by Shen et al. [36] on the host level for a node a for graph instances from time t_0 and t_1. We let $G(t)$ denote the graph instance at time t and $I^{(t)}(a)$, $\Gamma^{(t)}(a)$ denote the set of in and out-links of node a at time t, respectively.

- In-link death (IDR) and growth rate (IGR):

$$\mathrm{IDR}(a) = \frac{|I^{(t_0)}(a) - I^{(t_1)}(a)|}{|I^{(t_0)}(a)|}$$

$$\mathrm{IGR}(a) = \frac{|I^{(t_1)}(a) - I^{(t_0)}(a)|}{|I^{(t_0)}(a)|}$$

- Out-link death and growth rates (ODR, OGR): the above features calculated for out-links;
- Mean and variance of IDR, IGR, ODR and OGR across in-neighbors of a host (IDRMean, IDRVar, etc.);
- Change rate of the clustering coefficient (CRCC), i.e. the fraction of linked hosts within those pointed by pairs of edges from the same host:

$$CC(a,t) = \frac{|\{(b,c) \in G(t) | b,c \in \Gamma^{(t)}(a)|}{|\Gamma^{(t)}(a)|}$$

$$CRCC(a) = \frac{CC(a,t_1) - CC(a,t_0)}{CC(a,t_0)}$$

- Derivative features such as the ratio and product of the in and out-link rates, means and variances. We list the in-link derivatives; out-link ones are defined similarly:

  IGR·IDR, IGR/IDR, IGRMean/IGR, IGRVar/IGR, IDRMean/IDR, IDRVar/IDR, IGRMean·IDRMean, IGRMean/IDRMean, IGRVar·IDRVar, IGRVar/IDRVar.

2.1.2 Self-Similarity Along Time

In the next sections we introduce new linkage change features based on multi-step graph similarity measures that in some sense generalize the single-step neighborhood change features of the previous section. We characterize the change of the multi-step neighborhood of a node by defining the similarity of a single node *across* snapshots instead of two nodes within a single graph instance. The basic idea is that, for each node, we measure its similarity to itself in two identically labeled graphs representing two consecutive points of time. This enables us to measure the linkage change occurring in the observed time interval using ordinary graph similarity metrics.

We consider two graph similarity measures, XJaccard and PSimRank [23]; we also argue why SimRank [29] is inappropriate for constructing temporal features.

SimRank of a pair of nodes u and v is defined recursively as the average similarity of the neighbors of u and v:

$$\begin{aligned} \mathrm{Sim}_{\ell+1}(u,v) &= 1, \text{ if } u = v; \\ \mathrm{Sim}_{\ell+1}(u,v) &= \sum_{\substack{v' \in I(v) \\ u' \in I(u)}} \mathrm{Sim}_{\ell}(u',v'). \end{aligned} \tag{1}$$

In order to apply SimRank for similarity of a node v between two snapshots t_0 and t_1, we apply (1) so that v' and u' are taken from different snapshots.

Next we describe a known deficiency of SimRank in its original definition that rules out its applicability for temporal analysis. First we give the example for the single graph SimRank. Consider a bipartite graph with k nodes pointing all to another two u and v. In this graph there are no directed paths of length more than one and hence the Sim values can be computed in a single iteration. Counterintuitively, we get $\mathrm{Sim}(u,v) = c/k$, i.e. the larger the cocitation of u and v, the smaller their SimRank value. The reason is that the more the number of in-neighbors, the more likely is that a pair of random neighbors will be different.

While the example of the misbehavior for SimRank is somewhat artificial in the single-snapshot case, next we show that this phenomenon almost always happens if we consider the similarity of a single node v across two snapshots. If there is no change at all in the neighborhood of node v between the two snapshots, we expect the Sim value to be maximal. However the situation is identical to the bipartite graph case and Sim will be inversely proportional to the number of out-links.

2.1.3 Extended Jaccard Similarity Along Time

Our first definition of similarity is based on the extension of the Jaccard coefficient in a similar way XJaccard is defined in [23]. The Jaccard similarity of a page or host v across two snapshots t_0 and t_1 is defined by the overlap of its neighborhood in the two snapshots, $\Gamma^{(t_0)}(v)$ and $\Gamma^{(t_1)}(v)$ as

$$\mathsf{Jac}^{(t_0,t_1)}(v) = \frac{|\Gamma^{(t_0)}(v) \cap \Gamma^{(t_1)}(v)|}{|\Gamma^{(t_0)}(v) \cup \Gamma^{(t_1)}(v)|}$$

The *extended Jaccard coefficient, XJaccard* for length ℓ of a page or host is defined via the notion of the neighborhood $\Gamma_k^{(t)}(v)$ at distance exactly k as

$$\mathsf{XJac}_{\ell}^{(t_0,t_1)}(v) = \sum_{k=1}^{\ell} \frac{|\Gamma_k^{(t_0)}(v) \cap \Gamma_k^{(t_1)}(v)|}{|\Gamma_k^{(t_0)}(v) \cup \Gamma_k^{(t_1)}(v)|} \cdot c^k(1-c)$$

The XJac values can be approximated by the min-hash fingerprinting technique for Jaccard coefficients [10], as described in Algorithm 3 of [23]. The fingerprint generation algorithm has to be repeated for each graph snapshot, with the same set of independent random permutations.

We generate temporal features based on the XJac values for four length values $\ell = 1 \ldots 4$. We also repeat the computation on the transposed graph, i.e. replacing out-links $\Gamma^{(t)}(v)$ by in-links $I^{(t)}(v)$. As suggested in [23], we set the decay factor $c = 0.1$ as this is the value where, in their experiments, XJaccard yields best average quality for similarity prediction.

Similar to [36], we also calculate the mean and variance $\mathsf{XJac}^{(t_0,t_1)}{}_{\ell}(w)$ of the neighbors w for each node v. The following derived features are also calculated:

- similarity at path length $\ell = 2, 3, 4$ divided by similarity at path length $\ell - 1$, and the logarithm of these;
- logarithm of the minimum, maximum, and average of the similarity at path length $\ell = 2, 3, 4$ divided by the similarity at path length $\ell - 1$.

2.1.4 PSimRank Along Time

Next we define similarity over time based on PSimRank, a SimRank variant defined in [23] that can be applied similar to XJaccard in the previous section. As we saw in Section 2.1.2, SimRank is inappropriate for measuring linkage change in time. In the terminology of the previous subsection, the reason is that path fingerprints will be unlikely to meet in a large neighborhood and SimRank values will be low even if there is completely no change in time.

We solve the deficiency of SimRank by allowing the random walks to meet with higher probability when they are close to each other: a pair of random walks at vertices u', v' will advance to the same vertex (i.e., meet in one step) with probability of the Jaccard coefficient $\frac{|I(u') \cap I(v')|}{|I(u') \cup I(v')|}$ of their in-neighborhood $I(u')$ and $I(v')$.

The random walk procedure corresponding to PSimRank along with a fingerprint generation algorithm is defined in [23].

For the temporal version, we choose independent random permutations σ_ℓ on the hosts for each step ℓ. In step ℓ if the random walk from vertex u is at u', it will step to the in-neighbor with smallest index given by the permutation σ_ℓ in each graph snapshot.

Temporal features are derived from the PSimRank similarity measure very much the same way as for XJaccard, for four length values $\ell = 1 \ldots 4$. We also repeat the computation on the transposed graph, i.e. replacing out-links $\Gamma^{(t)}(v)$ by in-links $I^{(t)}(v)$. As suggested in [23], we set the decay factor $c = 0.15$ as this is the value where, in their experiments, PSimRank yields best average quality for similarity prediction. Additionally, we calculate the mean and variance $\mathrm{PSimRank}(w)$ of the neighbors w for each node v and derived features as for XJaccard.

2.2 Content and its Change

The content of Web pages can be deployed in content classification either via statistical features such as entropy [34] or via term weight vectors [39, 17]. Some of the more complex features that we do not consider in this work include language modeling [2].

In this section we focus on capturing term-level changes

over time. For each target site and crawl snapshot, we collect all the available HTML pages and represent the site as the bag-of-words union of all of their content. We tokenize content using the ICU library[2], remove stop words[3] and stem using Porter's method.

We treat the resulting term list as the virtual document for a given site at a point of time. As our vocabulary we use the most frequent 10,000 terms found in at least 10% and at most 50% of the virtual documents.

To measure the importance of each term i in a virtual document d at time snapshot T, we use the BM25 weighting [35]:

$$t_{i,d}^{(T)} = \text{IDF}_i^{(T)} \cdot \frac{(k_1+1)\text{tf}_{i,d}^{(T)}}{K + \text{tf}_{i,d}^{(T)}}$$

where $\text{tf}_{i,d}^{(T)}$ is the number of occurrences of term i in document d and $\text{IDF}_i^{(T)}$ is the inverse document frequency (Robertson-Spärck Jones weight) for the term at time T. The length normalized constant K is specified as

$$k_1((1-b) + b \times \text{dl}^{(T)}/\text{avdl}^{(T)})$$

such that $dl^{(T)}$ and $avdl^{(T)}$ denote the virtual document length and its average at time T, respectively. Finally

$$\text{IDF}^{(T)} = log\frac{N - n^{(T)} + 0.5}{n^{(T)} + 0.5}$$

where N denotes the total number of virtual documents and $n^{(T)}$ is the number of virtual documents containing term i. Note that we keep N independent of T and hence if document d does not exist at T, we consider all $\text{tf}_{i,d}^{(T)} = 0$.

By using the term vectors as above, we calculate the temporal content features described in [17] in the following five groups.

- **Ave:** Average BM25 score of term i over the $T_{\max}$ snapshots:

$$\text{Ave}_{i,d} = \frac{1}{T_{\max}} \cdot \sum_{T=1}^{T_{\max}} t_{i,d}^{(T)}$$

- **AveDiff:** Mean difference between temporally successive term weight scores:

$$\text{AveDiff}_{i,d} = \frac{1}{T_{\max}-1} \cdot \sum_{T=1}^{T_{\max}-1} |t_{i,d}^{(T+1)} - t_{i,d}^{(T)}|$$

- **Dev:** Variance of term weight vectors at all time points:

$$\text{Dev}_{i,d} = \frac{1}{T_{\max}-1} \cdot \sum_{T=1}^{T_{\max}} (t_{i,d}^{(T)} - \text{Ave}_{i,d})^2$$

- **DevDiff:** Variance of term weight vector differences of temporally successive virtual documents:

$$\text{DevDiff}_i = \frac{1}{T_{\max}-2} \cdot \sum_{T=1}^{T_{\max}-1} (|t_{i,d}^{(T+1)} - t_{i,d}^{(T)}| - \text{AveDiff}_i)^2$$

- **Decay:** Weighted sum of temporally successive term weight vectors with exponentially decaying weight. The base of the exponential function, the *decay rate* is denoted by λ. **Decay** is defined as follows:

$$\text{Decay}_{i,d} = \sum_{T=1}^{T_{\max}} \lambda e^{\lambda(T_{\max}-T)} t_{i,j}^{(T)}$$

[2]http://icu-project.org/

[3]http://www.lextek.com/manuals/onix/stopwords1.html

3. CLASSIFICATION FRAMEWORK

For the purposes of our experiments we computed all the public Web Spam Challenge content and link features of [13]. We applied the classification techniques found most effective in our work [21]. We built a classifier ensemble by splitting features into related sets and for each we use a collection of classifiers that fit the data type and scale. These classifiers were then combined by ensemble selection. We used the classifier implementations of the machine learning toolkit Weka [38].

The motivation for using ensemble selection is that recently this particular ensemble method gained more attention thanks to the winners of KDD Cup 2009 [33]. According to our experiments [21] ensemble selection performed significantly better than other classifier combination methods used for Web spam detection in the literature, such as log-odds based averaging [32] and bagging.

We used the ensemble selection implementation of Weka [38] for performing the experiments. The Weka implementation supports the proven strategies for avoiding overfitting such as model bagging, sort initialization and selection with replacement. We allow Weka to use all available models in the library for greedy sort initialization and use 5-fold embedded cross-validation during ensemble training and building. We set AUC as the target metric to optimize for and run 100 iterations of the hillclimbing algorithm.

We mention that we have to be careful with treating missing feature values. Since the temporal features are based on at least two snapshots, for a site that appears only in the last one, all temporal features have missing value. For classifiers that are unable to treat missing values we define default values depending on the type of the feature.

3.1 Learning Methods

We use the following models in our ensemble: bagged and boosted decision trees, logistic regression, naive Bayes and variants of random forests. For most classes of features we use all classifiers and let selection choose the best ones. The exception is static and temporal term vector based features where, due to the very large number of features, we may only use Random Forest and SVM. We train our models as follows.

Bagged LogitBoost: we do 10 iterations of bagging and vary the number of iterations from 2 to 64 in multiples of two for LogitBoost.

Decision Trees: we generate J48 decision trees by varying the splitting criterion, pruning options and use either Laplacian smoothing or no smoothing at all.

Bagged Cost-sensitive Decision Trees: we generate J48 decision trees with default parameters but vary the cost sensitivity for false positives in steps of 10 from 10 to 300. We do the same number of iterations of bagging as for LogitBoost models.

Logistic Regression: we use a regularized model varying the ridge parameter between 10^{-8} to 10^4 by factors of 10. We normalize features to have mean 0 and standard deviation 1.

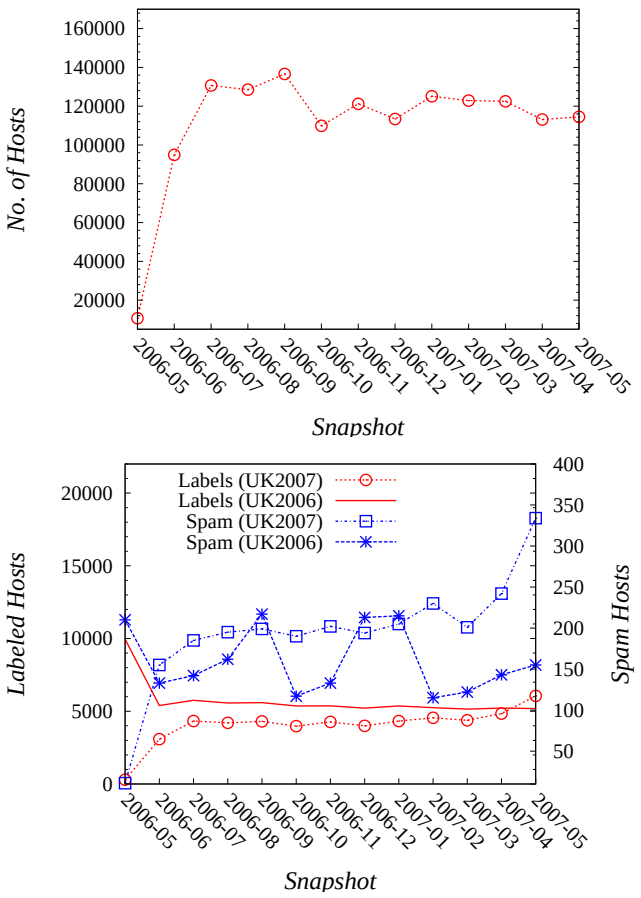

Figure 1: The number of total hosts as well as labeled hosts distinguished by the label set and the label value.

Random Forests: we use FastRandomForest [22] instead of the native Weka implementation for faster computation. The forests have 250 trees and, as suggested in [9], the number of features considered at each split is $s/2$, s, $2s$, $4s$ and $8s$, where s is the square root of the total number of features available.

Naive Bayes: we allow Weka to model continuous features either as a single normal or with kernel estimation, or we let it discretize them with supervised discretization.

4. RESULTS AND DISCUSSION

Our data set is derived from the 13 `.uk` snapshots provided by the Laboratory for Web Algorithmics of the Università degli studi di Milano together with the Web Spam Challenge labels WEBSPAM-UK2007. We extracted maximum 400 pages per site from the original crawls. The last 12 of the above `.uk` snapshots were analyzed by Bordino et al. [8] who among others observe a relative low URL but high host overlap. The first snapshot (2006-05) that is identical to WEBSPAM-UK2006 was chosen to be left out from their experiment since it was provided by a different crawl strategy. We observed that the last 8 snapshots contain a stable fraction of hosts (both labeled and unlabeled) for our experiments as seen in Fig. 1. From now on we restrict attention to the latter snapshots and the WEBSPAM-UK2007 labels only.

For calculating the temporal link-based features described in Section 2 we use the host level graph. Similar to the observation of [8], pages are much more unstable over time compared to hosts. Note that page-level fluctuations may simply result from the sequence the crawler visited the pages and not necessarily reflect real changes. The crawl induced noise and problems with URL canonization [4] rule out the applicability of features based on the change of page-level linkage.

To make it easy to compare our results to previous results, we cite the Web Spam Challenge 2008 winner's performance in each table in the following, as published in their original paper [25]. They trained a bagged classifier on the standard content-based and link-based features published by the organizers of the Web Spam Challenge 2008 and on custom host-graph based features, using the ERUS strategy for class-inbalance learning [25].

4.1 Classifier Models

In this subsection we describe the performance of various classifier ensemble combinations[4]. We do not aim to provide an exhaustive evaluation of all combinations. Instead, we concentrate our efforts on determining whether temporal information is valuable for Web spam detection.

For training and testing we use the official Web Spam Challenge 2008 training and test sets [13]. As it can be seen in Table 1 these show considerable class imbalance which makes the classification problem harder.

Label Set	Instances	%Positive
Training	4000	5.95%
Testing	2053	4.68%

Table 1: Summary of label sets for Web Spam Challenge 2008.

4.1.1 Temporal Link-only

First, we compare the temporal link features proposed in Section 2.1 with those published earlier [36]. Then, we build ensembles that combine the temporal with the public link-based features described by [5]. The results are summarized in Table 2.

Section	Feature Set	No. of Features	AUC
2.1.1	Growth/death rates	29	0.617
2.1.2	XJaccard + PSimRank	63	0.625
	Public link-based [5]	176	0.765
2.1.1	Public + growth/death rates	205	0.758
2.1.2	Public + XJaccard + PSimRank	239	**0.769**
	All link-based	268	0.765
	WSC 2008 Winner	-	0.852

Table 2: Performance of ensembles built on link-based features.

[4]The exact classifier model specification files used for Weka and the data files used for the experiments are available upon request from the authors.

As these measurements show, our proposed graph similarity based features successfully extend the growth and death rate based ones by achieving higher accuracy, improving AUC by 1.3%. However, by adding temporal to static link-based features we get only marginally better ensemble performance.

To rank the link-based feature sets by their contribution in the ensemble, we build classifier models on the three separate feature subsets (public link-based, growth/death rate based and graph similarity based features, respectively) and let ensemble selection combine them. This restricted combination results in a slightly worse AUC of 0.762. By calculating the total weight contribution, we get the following ranked list (weight contribution showed in parenthesis: public link-based (60.8%), graph similarity based (21.5%), growth/death rate based (17.7%). This ranking also supports the findings presented in Table 2 that graph similarity based temporal link-based features should be combined with public link-based features if temporal link-based features are used.

To separate the effect of ensemble selection on the performance of temporal link-based feature sets we repeat the experiments with bagged cost-sensitive decision trees only, a model reported to be effective for web spam classification [34]. The results for these experiments are shown in Table 3.

Section	Feature Set	No. of Features	AUC
2.1.1	Growth/death rates	29	0.605
2.1.2	XJaccard	42	0.626
2.1.3	PSimRank	21	0.593
2.1.2-3	XJaccard + PSimRank	63	0.610
	Public link-based [5]	176	**0.731**
2.1.1	Public + growth/death rates	205	0.696
2.1.2-3	Public + XJaccard + PSimRank	239	*0.710*
	All link-based	268	0.707
	WSC 2008 Winner	-	0.852

Table 3: Performance of bagged cost-sensitive decision trees trained on link-based features.

As it can be seen in Table 3, when using bagged cost-sensitive decision trees, our proposed temporal link-based similarity features achieve 3.5% better performance than the growth/death rate based features published earlier.

When comparing results in Table 3 and in Table 2 we can see that ensemble selection i) significantly improves accuracy (as expected) and ii) diminishes the performance advantage achievable by the proposed temporal link-based features over the previously published ones.

As prevalent from Table 3, the proposed PSimRank based temporal features perform roughly the same as the growth and death rate based ones while the XJaccard based temporal features perform slightly better.

Next we perform sensitivity analysis of the temporal link-based features by using bagged cost-sensitive decision trees. We build 10 different random training samples for each of the possible fractions 10%, 20%, ..., 100% of all available labels. In Fig. 2 we can see that the growth/death rate based features as well as the PSimRank based features are not sensitive to training set size while the XJaccard based ones are. That is, even though XJaccard is better in terms of performance than the other two feature sets considered it is more sensitive to the amount of training data used as well.

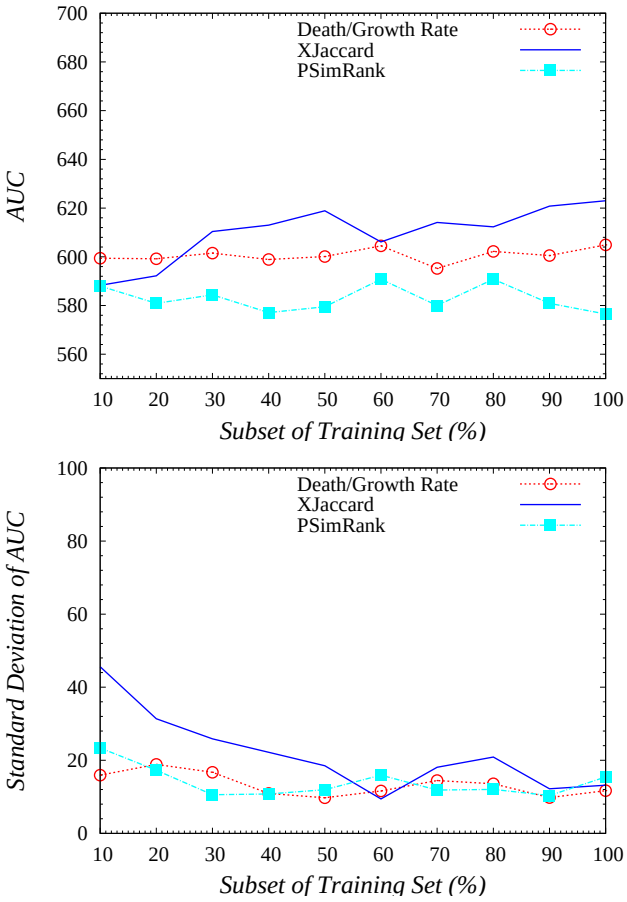

Figure 2: Sensitivity of temporal link-based features. Top: AUC values averaged across 10 measurements. Bottom: standard deviations of AUC for different training set sizes.

4.1.2 Content-only Ensemble

We build two ensembles, the first based on the Public content [34] features and the second on static term weight vector derived from the BM25 term weighting scheme (see Section 2.2). As seen in Table 4, the combination is by 5% stronger than the Web Spam Challenge 2008 winner [25].

Feature Set	No. of Features	AUC
Public content [34]	96	0.879
Public content + BM25	10096	**0.893**
WSC 2008 Winner [25]	-	0.852

Table 4: Performance of ensembles built on static content-based features.

4.1.3 Content-only Ensembles

We build ensembles based on the temporal content features described in Section 2.2 and their combination them-

selves, with the static BM25 features, and with the content-based features of [34]. The performance comparison of temporal content-based ensembles is presented in Table 5.

Feature Set	AUC
Static BM25	0.736
Ave	0.749
AveDiff	0.737
Dev	0.767
DevDiff	0.752
Decay	0.709
Temporal combined	0.782
Temporal combined + BM25	**0.789**
Public content-based [34] + temporal	0.901
All combined	**0.902**

Table 5: Performance of ensembles built on temporal content-based features.

4.1.4 Full Ensemble

By combining all the content and link-based features, both temporal and static ones, we train an ensemble which incorporates all the previous classifiers. This combination resulted in an AUC of 0.908 meaning no significant improvement can be achieved with link-based features over the content-based ensemble.

5. CONCLUSIONS

With the illustration over the 100,000 page WEBSPAM-UK2007 data along with 7 previous monthly snapshots of the `.uk` domain, we have presented a survey of temporal features for Web spam classification. We investigated the performance of both link- and content-based Web spam features with ensemble selection, focusing on temporal link-based features[5].

We proposed graph similarity based temporal features which aim to capture the nature of linkage change of the neighborhoods of hosts. We have shown how to compute these features efficiently on large graphs using a Monte Carlo method. Our features achieve better performance than previously published methods, however, when combining them with the public link-based feature set we get only marginal performance gain.

By our experiments it has turned out that the appropriate choice of the machine learning techniques is probably more important than devising new complex features. However, by using temporal information, we reach improvement in linkage based classification, a promising direction for filtering mixed language domains where content cannot be reliably used for classification [26].

Acknowledgment

To Sebastiano Vigna, Paolo Boldi and Massimo Santini for providing us with the UbiCrawler crawls [6, 7]. In addition to them, also to Ilaria Bordino, Carlos Castillo and Debora Donato for discussions on the WEBSPAM-UK data sets [8].

[5]The temporal feature data used in our research is available at: `http://datamining.ilab.sztaki.hu/?q=en/downloads`

6. REFERENCES

[1] L. D. Artem Sokolov, Tanguy Urvoy and O. Ricard. Madspam consortium at the ecml/pkdd discovery challenge 2010. In *Proceedings of the ECML/PKDD 2010 Discovery Challenge*, 2010.

[2] J. Attenberg and T. Suel. Cleaning search results using term distance features. In *Proceedings of the 4th international workshop on Adversarial information retrieval on the web*, pages 21–24. ACM New York, NY, USA, 2008.

[3] Z. Bar-Yossef, A. Z. Broder, R. Kumar, and A. Tomkins. Sic transit gloria telae: Towards an understanding of the web's decay. In *Proceedings of the 13th World Wide Web Conference (WWW)*, pages 328–337. ACM Press, 2004.

[4] Z. Bar-Yossef, I. Keidar, and U. Schonfeld. Do not crawl in the dust: different urls with similar text. *ACM Transactions on the Web (TWEB)*, 3(1):1–31, 2009.

[5] L. Becchetti, C. Castillo, D. Donato, S. Leonardi, and R. Baeza-Yates. Link-based characterization and detection of web spam. In *Proceedings of the 2nd International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2006.

[6] P. Boldi, B. Codenotti, M. Santini, and S. Vigna. Ubicrawler: A scalable fully distributed web crawler. *Software: Practice & Experience*, 34(8):721–726, 2004.

[7] P. Boldi, M. Santini, and S. Vigna. A Large Time Aware Web Graph. *SIGIR Forum*, 42, 2008.

[8] I. Bordino, P. Boldi, D. Donato, M. Santini, and S. Vigna. Temporal evolution of the uk web. In *Workshop on Analysis of Dynamic Networks (ICDM-ADN'08)*, 2008.

[9] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] A. Z. Broder. On the Resemblance and Containment of Documents. In *Proceedings of the Compression and Complexity of Sequences (SEQUENCES'97)*, pages 21–29, 1997.

[11] C. Castillo, K. Chellapilla, and L. Denoyer. Web spam challenge 2008. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2008.

[12] C. Castillo and B. D. Davison. Adversarial web search. *Foundations and Trends in Information Retrieval*, 4(5):377–486, 2010.

[13] C. Castillo, D. Donato, L. Becchetti, P. Boldi, S. Leonardi, M. Santini, and S. Vigna. A reference collection for web spam. *SIGIR Forum*, 40(2):11–24, December 2006.

[14] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: Web spam detection using the web topology. Technical report, DELIS – Dynamically Evolving, Large-Scale Information Systems, 2006.

[15] J. Cho and H. Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *The VLDB Journal*, pages 200–209, 2000.

[16] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In *Proceedings of the International Conference on Management of Data*, pages 117–128, 2000.

[17] N. Dai, B. D. Davison, and X. Qi. Looking into the past to better classify web spam. In *AIRWeb '09: Proceedings of the 5th international workshop on Adversarial information retrieval on the web*. ACM Press, 2009.

[18] A. Dong, Y. Chang, Z. Zheng, G. Mishne, J. Bai, K. Buchner, R. Zhang, C. Liao, and F. Diaz. Towards recency ranking in web search. In *Proc. WSDM*, 2010.

[19] N. Eiron, K. S. McCurley, and J. A. Tomlin. Ranking the web frontier. In *Proceedings of the 13th International World Wide Web Conference (WWW)*, pages 309–318, New York, NY, USA, 2004. ACM Press.

[20] M. Erdélyi, A. A. Benczúr, J. Masanés, and D. Siklósi. Web spam filtering in internet archives. In *AIRWeb '09: Proceedings of the 5th international workshop on Adversarial information retrieval on the web*. ACM Press, 2009.

[21] M. Erdélyi, A. Garzó, and A. A. Benczúr. Web spam classification: a few features worth more. In *Joint WICOW/AIRWeb Workshop on Web Quality (WebQuality 2011) In conjunction with the 20th International World Wide Web Conference in Hyderabad, India.* ACM Press, 2011.

[22] FastRandomForest. Re-implementation of the random forest classifier for the weka environment. `http://code.google.com/p/fast-random-forest/`.

[23] D. Fogaras and B. Rácz. Scaling link-based similarity search. In *Proceedings of the 14th World Wide Web Conference (WWW)*, pages 641–650, Chiba, Japan, 2005.

[24] J. Fogarty, R. S. Baker, and S. E. Hudson. Case studies in the use of roc curve analysis for sensor-based estimates in human computer interaction. In *Proceedings of Graphics Interface 2005*, GI '05, pages 129–136, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[25] G. Geng, X. Jin, and C. Wang. CASIA at WSC2008. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2008.

[26] X.-C. Z. Guang-Gang Geng, Xiao-Bo Jin and D. Zhang. Evaluating web content quality via multi-scale features. In *Proceedings of the ECML/PKDD 2010 Discovery Challenge*, 2010.

[27] Z. Gyöngyi and H. Garcia-Molina. Spam: It's not just for inboxes anymore. *IEEE Computer Magazine*, 38(10):28–34, October 2005.

[28] M. R. Henzinger, R. Motwani, and C. Silverstein. Challenges in web search engines. *SIGIR Forum*, 36(2):11–22, 2002.

[29] G. Jeh and J. Widom. SimRank: A measure of structural-context similarity. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 538–543, 2002.

[30] Y. joo Chung, M. Toyoda, and M. Kitsuregawa. A study of web spam evolution using a time series of web snapshots. In *AIRWeb '09: Proceedings of the 5th international workshop on Adversarial information retrieval on the web*. ACM Press, 2009.

[31] Y. Lin, H. Sundaram, Y. Chi, J. Tatemura, and B. Tseng. Splog detection using content, time and link structures. In *2007 IEEE International Conference on Multimedia and Expo*, pages 2030–2033, 2007.

[32] T. Lynam, G. Cormack, and D. Cheriton. On-line spam filter fusion. *Proc. of the 29th international ACM SIGIR conference on Research and development in information retrieval*, pages 123–130, 2006.

[33] A. Niculescu-Mizil, C. Perlich, G. Swirszcz, V. Sindhwani, Y. Liu, P. Melville, D. Wang, J. Xiao, J. Hu, M. Singh, et al. Winning the KDD Cup Orange Challenge with Ensemble Selection. In *KDD Cup and Workshop in conjunction with KDD 2009*, 2009.

[34] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam web pages through content analysis. In *Proceedings of the 15th International World Wide Web Conference (WWW)*, pages 83–92, Edinburgh, Scotland, 2006.

[35] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *In Proceedings of SIGIR'94*, pages 232–241. Springer-Verlag, 1994.

[36] G. Shen, B. Gao, T. Liu, G. Feng, S. Song, and H. Li. Detecting link spam using temporal information. In *ICDM'06.*, pages 1049–1053, 2006.

[37] A. Singhal. Challenges in running a commercial search engine. In *IBM Search and Collaboration Seminar 2004*. IBM Haifa Labs, 2004.

[38] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, second edition, June 2005.

[39] B. Zhou, J. Pei, and Z. Tang. A spamicity approach to web spam detection. In *Proceedings of the 2008 SIAM International Conference on Data Mining (SDM'08)*, pages 277–288. Citeseer, 2008.

Deriving Dynamics of Web Pages: A Survey*

Marilena Oita
INRIA Saclay – Île-de-France
& Télécom ParisTech; CNRS LTCI
Paris, France
marilena.oita@telecom-paristech.fr

Pierre Senellart
Institut Télécom
Télécom ParisTech; CNRS LTCI
Paris, France
pierre.senellart@telecom-paristech.fr

ABSTRACT

The World Wide Web is dynamic by nature: content is continuously added, deleted, or changed, which makes it challenging for Web crawlers to keep up-to-date with the current version of a Web page, all the more so since not all apparent changes are significant ones. We review major approaches to change detection in Web pages and extraction of temporal properties (especially, timestamps) of Web pages. We focus our attention on techniques and systems that have been proposed in the last ten years and we analyze them to get some insight into the practical solutions and best practices available. We aim at providing an analytical view of the range of methods that can be used, distinguishing them on several dimensions, especially, their static or dynamic nature, the modeling of Web pages, or, for dynamic methods relying on comparison of successive versions of a page, the similarity metrics used. We advocate for more comprehensive studies of the effectiveness of Web page change detection methods, and finally highlight open issues.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*

General Terms

Algorithms, Experimentation, Measurement

Keywords

Change monitoring, Web archiving, Timestamping

1. INTRODUCTION

The World Wide Web challenges our capacity to develop tools that can keep track of the huge amount of information that is getting modified at speed rate. Web archiving crawlers [25], especially, need the ability to detect change in Web content and to infer when a Web page last changed. This ability is fundamental in maintaining the coherence of the crawl, in adjusting its refresh rate, in versioning, and to allow a user to retrieve meaningful temporal data. The understanding of the dynamics of Web pages, that is, how fast the Web content changes and what the nature of these changes is, its implications on the structure of the Web, and the correlation with the topic of pages are also popular subjects in the research literature [4].

In addition to being of paramount importance to Web archiving, the subject of change detection is of interest in various applications and domains, such as: large-scale information monitoring and delivery systems [18, 15, 24, 17, 23] or services[1], Web cache improvement [11], version configuration and management of Web archives [30], active databases [18], servicing of continuous queries [1]. Research has focused on finding novel techniques for comparing snapshots of Web pages (a reference Web page and its updated version) in order to detect change and estimate its frequency. Change can however be detected at various levels: there are various aspects of dynamics which must be considered when studying how Web content changes and evolves.

The majority of works have seen the change detection problem from a *document-centric* perspective, as opposed to an *object-centric* one. By object or entity we mean here any Web content, part of a Web page, that represents meaningful information per se: image, news article, blog post, comment, etc.. Comparatively, little effort has been put on making the difference between relevant changes and those that might occur because of the dynamic template of a Web page (ads, active layout, etc.), i.e., its boilerplate [21].

We study in this article some of the strategies that have been established in different settings, with the aim at providing an overview of the existing techniques used to derive temporal properties of Web pages.

There is a large body of work on the related problem of change detection in XML documents, particularly for purposes of data integration and update management in XML-centric databases. However, the solutions developed for XML documents cannot be applied without serious revisions for HTML pages. The model assumptions made for XML do not really hold for HTML. Indeed, the HTML and XML formats have a key difference: while an XML page defines the nature of the content by its *meta* tags, HTML tags define

*This research was funded by the European Research Council grant Webdam FP7-ICT-226513.

TWAW 2011 March 28, 2011, Hyderabad, India

[1]Examples of these include `http://timelyweb.en.softonic.com/`, `http://www.urlywarning.net/`, `http://www.changealarm.com/`, `http://www.changedetect.com/`.

mainly presentational aspects of content within. In addition to the challenges that exist in XML documents, Web pages add some others by their lack of formal semantics, a fuzziness regarding the formatting, by the embedding of multimedia and scripts, etc.. Separate approaches commonly need to be adopted and the research done on XML documents is beyond the scope of our paper, although we mention some works on XML [37, 14] that have particular relevance to Web page change detection. We refer the reader to [13] for a survey of XML change detection algorithms.

We focus in this survey on deriving dynamics of **HTML documents**.

Change detection mechanisms can either be *static*, estimating the date of last modification of content from the Web page itself (its code, semantics or neighbors), or *dynamic*, by comparing successive versions of a Web page. The structure of this article reflects these dimensions. In Section 2, we present static approaches to timestamping Web pages, while Section 3 introduce dynamic methods. We then analyze in Section 4 the different models of a Web page used by existing techniques for comparing successive versions. Similarity metrics used in dynamic methods are independently investigated in Section 5. We briefly describe statistical modeling approaches to estimate change frequency in Section 6. We conclude with a discussion of some remaining open questions.

2. STATIC APPROACHES: TIMESTAMPING

This section deals with methods for inferring temporal properties of a Web page in a *static* manner as opposed to the commonly used *dynamic* computation of the difference between successive versions of a given Web page. The goal here is to infer the creation or the last modification date of a Web page or, possibly, of some parts of it. We study sources of data that can be useful for that purpose: metadata, the content of the Web page itself, or its graph neighborhood. The canonical way for timestamping a Web page is to use the `Last-Modified` HTTP header. Unfortunately, studies have shown this approach is not reliable in general [12]. We describe next why this happens in practice and other techniques for timestamping Web pages.

2.1 HTTP metadata

HTTP/1.1, the main protocol used by Web clients and servers to exchange information, offers several features of interest for timestamping, the foremost of which are the `ETag` and `Last-Modified` HTTP response headers. Entity tags (or ETags) are unique identifiers for a given version of a particular document. They are supposed to change if and only if the document itself changes. Servers can return this with any response, and clients can use the `If-Match` and `If-None-Match` HTTP requests headers to condition the retrieval of the document to a change in the ETag value, avoiding then to retrieve already known contents. `If-Modified-Since` and `If-Unmodified-Since` provide conditional downloading features, in a similar way as for ETags. Even when conditional downloading is not possible, Etags and HTTP timestamps can be retrieved by a Web client without downloading a whole Web page by making use of the `head` HTTP method. The problem is that while this information is generally provided and is very reliable for static content (e.g., static HTML pages or PDF), it is most of the time missing or changed at every request (the timestamp given is that of the request, not of the content change) when the content is dynamic (generated by content management systems, etc.). Some CMSs do return correct HTTP timestamps, such as MediaWiki[2], but they seem to be a minority.

In [12], Clausen presents an experimental study of the reliability of Etags and HTTP timestamps on a collection of a few million Danish Web pages. He finds out that the best strategy for avoiding useless downloads of versions of Web pages already available in a Web archive is to always download when the ETag server is missing, and otherwise download only if the `Last-Modified` header indicates change. This rather counterintuitive result yielded in this experimental study an almost perfect prediction of change, and a 63% accuracy in predicting non-change. Given that the majority of Web servers run some version of the open-source Apache[3] HTTP server [26], it would be interesting to see whether this strategy is correlated with some inherent behavior of this software. Furthermore, repeating this experiment on a larger scale and with a more recent set of Web pages would be of interest.

HTTP also provides the `Cache-Control` and `Expires` response headers. This information is often given, but with a zero or very low expiration delay, which means that nothing interesting can be derived from it. In some specific and controlled environments (e.g., Intranets), it might still be useful to look at these two pieces of information to estimate the refresh rate of a Web page.

2.2 Timestamps in Web content

Content management systems, as well as Web authors, often provide in the content of a Web page some human-readable information about its last date of modification. This can be a global timestamp (for instance, preceded by a "Last modified" string, in the footer of a Web page) or a set of timestamps for individual items in the page, such as news stories, blog posts, comments, etc. In the latter case, the global timestamp might be computed as the maximum of the set of individual timestamps. It is actually quite easy to extract and recognize such information, with keyword selection (*last*, *modification*, *date*, etc.) or with entity recognizers for dates (built out of simple regular expressions). However, this timestamp is often quite informal and partial: there is sometimes no time indication, and most of the time no timezone. To the best of our knowledge, no formal study of the precision reached by extracting timestamps from Web content has been carried out.

2.3 Semantic temporal associations

In addition to these timestamps provided to humans, documents on the Web may include additional semantic timestamps meant for machines. No mechanism for this exists in HTML *per se*, but the HTML specification [36] allows for arbitrary metadata in the form of `<meta>` tags, one particular profile of such metadata being Dublin Core[4] whose `modified` term indicates the date of last modification of a Web page. Both content management systems and Web authors occasionally use this possibility. Web *feeds* (in RSS or Atom formats) also have semantic timestamps, which are quite reliable, since they are essential to the working of applications that exploit them, such as feed readers. In some cases, external semantic content can be used for dating an HTML Web

[2] `http://www.mediawiki.org/`
[3] `http://www.apache.org/`
[4] `http://dublincore.org/documents/dcmi-terms/`

page: for instance, an RSS feed containing blog entries can be mapped to the corresponding Web page, in order to date individual items [29]. Another case is that of *sitemaps* [35]. Sitemaps are files that can by provided by the owner of a Web site to describe its organization, so as to improve its indexing by Web search engines. Sitemaps allow for both timestamps and change rate indications (*hourly*, *monthly*, etc.), but these features are not often used. Very few content management systems produce all of this, although it would have been the ideal case: the download of a single file would suffice to get all timestamping information about the whole Web site.

2.4 Using the neighborhood

It is possible to use the graph structure of the Web to help timestamping Web pages: [28] uses the neighboring pages of a Web page to estimate its timestamp. When no source of reliable timestamps is found for a given page using one of the technique described above, its timestamp is set to some form of average of the timestamps of pages pointed to and by this page. The inherent assumption is that pages linked together tend to have a similar update patterns. The precision is not very high, but better than nothing when no other information is available.

3. DYNAMIC METHODS

When static techniques do not give adequate results, there is still the possibility of comparing a Web page with its previous version in order to determine (sometimes in a rough way) an equivalent of `Last-Modified`. The timestamp that gives the last modification of a Web page can be inferred then as the date when change has been detected.

Nevertheless, for a precise estimation, a *just-in-time* crawl of versions is needed. In reality, the frequency of change is quite difficult to estimate because Web pages have different patterns of change. In general, many factors determine variations in the frequency of change for a given Web page: the CMS, the subject, the time of the year, even the time of the day, etc.

Estimating the Web pages' frequency of change is the subject of many studies [16, 2, 27, 10, 19]. Their results are however heavily dependent on the technique of detecting change that they have used.

There are two parameters that influence the process of determining the dynamics:

1. the **frequency of change** is not known in advance, but if we do not crawl the Web pages on time, we miss versions and the timestamp detected will be then imprecise;

2. the **change detection technique**, which heavily relies on the similarity metrics and on the model (that can capture more or less types of changes) and the filtering of dynamic elements that influence the frequency without being truly relevant (and which occur quite often in Web pages because of AJAX applications or advertisements).

A method of filtering irrelevant content is to know what is important rather than trying to filter what is unimportant. If we knew in advance the frequency of crawl, then we would know when change occurs and therefore set timestamps for new versions. This is unfortunately not the case, but we need however to crawl frequently enough (better too frequently than not frequent enough). Once we have these versions, we can detect if there is any change that happened in the interval of time that represents the interval of crawl. Based on this, timestamps can be derived with good approximation.

The majority of works consider that they have an access to the versions and they focus on detecting change efficiently. Few make a semantic distinction between changes by disregarding insignificant ones. Next, we present some general notions about changes: what kind of changes can occur in Web pages, which have been identified in our studied approaches and which have not, and finally we give an insight into how change is actually represented.

3.1 Types of changes

We summarize here the types of changes detected by different approaches. There are however other, more sophisticated types that are sometimes mentioned, but not treated. For instance, behavioral changes are mentioned in [38]; they occur in active HTML elements like scripts, embedded applications and multimedia. These new forms of Web content have a big impact on the Web today, but they require a more complex modeling.

All considered approaches detect changes in content.

Works that model the HTML page as a tree, including page digest encoding, usually detect also structural and attribute changes. However, differences exist in the coverage of cases for these types of changes. For example, MH-Diff [9] detects also *move* and *copy* structural changes, which is an improvement over the traditional detection of *insert*, *delete* and *update*. Structural (or layout) changes occur when the position of elements in the page is modified. Attribute (or presentation) changes are related to the representation of information, for instance changes in the font, colors or captions. For capturing structural and attribute changes, the model has to be aware their existence; this implies a more complex model which influences generally in a negatice manner the performance. Unlike flat-file models of Web pages, the output of content change detection in hierarchical models is more meaningful: the type of node in which the content change occured can also be identified.

There are also *type* changes, that are modifications which come about when the HTML tags change: e.g., a **p** tag which becomes a **div**. Type changes can be detected by [31] which uses the page digest encoding that provides a mechanism for locating nodes of a particular type (see further).

Semantic types of change capture the meaning of content that has changed. They are defined in SCD [23], a pioneer work in this direction.

Changes are sometimes captured in a quantitative manner rather than in a qualitative one. In contrast with the qualitative way, where the change is described (in a delta file) or visualized in a comparative manner, quantitative approaches estimate the amount of change of a specific type. More specifically, all approaches that use the similarity formula defined in CMW [17] do not reconstruct the complete sequence of changes, but give a numerical value of it. In this case, supposing a threshold of change, we can determine if a page has changed or not, which actually represent more an oracle response, still useful in the majority of applications.

3.2 The representation of change

There are various ways to present the difference between

two documents. Changes are usually stored in a physical structure generically called *delta file* or *delta tree*. The format of storing the change for RMS [38] consists in specialized set of arrays that capture the relationships among the nodes and the changes that occur both in structure and content. Systems for monitoring change like [24, 18] have typically a user interface and present changes in a graphical way. HTML*diff* merge the input Web page versions into one document that will summarize all the common parts and also the changed ones. The advantage is that the common parts are displayed just once, but on the other hand, the resulting merged HTML can be syntactically or semantically incorrect. Another choice linked to change presentation is to display only the differences and omit the common parts of the two Web pages. When the documents have a lot of data in common, presenting only the differences could be better, with the drawback that the context is missing. The last approach is to present the differences between the old and new version side by side.

These presentation modes are used in combination, rather than being the unique choice for a given system. For example, [24, 18] are presenting the results of the change monitoring service using a hybrid approach: presentation modes are combined depening on the type of change that is presented.

4. HTML DOCUMENT MODELS

This section contains an overview of the models that are considered in the quest of detecting changes in Web documents. The modeling step is a key one as it will determine the elements on which the comparison algorithms operate.

We first discuss the "naïve" approach, that is, to consider Web pages as flat files (strings); then we describe tree models, a popular choice in the literature. We explore also some approaches that are based on tree models, but which are essentially transforming the two versions to be compared in a bipartite graph on which specific algorithms are applied. Finally, we present the *Page Digest* design of Web pages, a manner of encoding that clearly separates structural elements of Web documents from their content, while remaining highly compact.

4.1 Flat-files

Some early change detection systems model Web pages as flat files [15]. As these models do not take into account the hierarchical structure of HTML documents and neither the characteristics of the layout, they can detect only *content changes* – and this without making any semantic difference in the content.

Some works [34] try to filter first irrelevant content by using heuristics on the type of content and regular expressions. After this basic filtering, the Web page content is direcly hashed and compared between versions. Unfortunately, we can never filter all kind of inconvenient content, especially when its manner of encoding or type get more complex.

4.2 Trees

A natural approach is to represent a Web page as a tree using the DOM model. By taking into account the hierarchies, *structural* and *attribute* changes can be detected besides *content* changes.

Differences between tree models appear in their *ordered* characteristics and in the level of granularity on which change is detected: node, branch, subtree or "object". We will further discuss these aspects.

First, the modeling of a Web page into a tree requires a preprocessing step of cleaning. This is a significant one because it corrects missing or mismatching, out-of-order end tags, as well as all other syntactic ill-formedness of the HTML document. A tree is constructed first by filtering the "tag soup" HTML into an XML document and second, by manipulating the result in order to obtain a workable tree structure using implementations of the DOM standard (that usually employ XSLT and XPath). Sometimes also an initial pruning of elements is done: [22] is filtering out scripts, applets, embedded objects and comments.

Many works do not specify how they realize the cleaning, so either they assume it to be done in advance, or they solve this issue by using an HTML cleaning tool. HTML Tidy[5] is well-suited for this purpose and mentioned in [3].

There are however cases [23] when the tree model does not need to be cleaned: as the semantics of tags (value, name) is leveraged in the technique, the structure does not have to be enforced.

Ordered trees.

The *ordered* characteristic of trees implies that the order of appearance of nodes is considered in the algorithm and therefore included in the model.

RMS algorithm [38] stores the level (*depth*) of a node, information which will be used in the tree traversal and parsing. The SCD [23] algorithm uses branches of ordered trees to detect *semantic* changes. The notion of branch is used to give the context of a node and is formalized as an ordered multiset, in which the elements are designated by the *tag name* of the HTML non-leaf node, or its content (text), if it represents a leaf node. The order is very important in this model because of the *data* hierarchy considered (e.g., `book.author.name.Eminescu` vs. a markup hierarchy like `div.p.b.PCDATA`). In this model, if we change the order, we change the semantics of hierarchies or this semantics becomes incoherent.

Ordered tree model is also used in [20].

Unordered trees.

The unordered labeled tree model does not consider the order of appearance of elements in the tree as relevant, instead only the parent-child relationships are captured.

It is mentioned in MHDiff [9] that the change detection problem for unordered trees is harder that for ordered ones. Like [17, 9], [22] constructs a weighted bipartite graph from the two trees given as entry. In these models, the order has not an importance on the final structure for which further processing will be done, therefore this feature is not captured.

[3] delimits and encodes subtrees; these are generated by analyzing the level of a node in the unordered tree. From the root, when we arrive at level % 3, a subtree is created; the node at (level % 3 + 1) becomes the *local* root of the following subtree, and this iteratively until the end of the hierarchy. Each subtree is marked with the tag name of its local root and indexed based on its start and end node identifiers. A hashtable will finally map metadata about nodes (like tag name, content, path to the root, attributes pair values, etc.).

[5] `http://tidy.sourceforge.net/`

4.3 Bipartite graph

The bipartite graph model is a model derived from unordered trees: it consists in two independent sets (acquired after tree pruning), each corresponding to a version, that are connected by cost edges. As set elements, subtrees are chosen over nodes in [22, 17]. The assumption [17] is that we might be more interested in which subtree the change occurs, than in which specific node.

The cost of an edge represent the cost of the edit scripting needed to make a model entity (node or subtree) of the first set *isomorphic* with one of the second set. The similarities between all subtrees in the first tree and all those of the second tree are computed and placed in a cost matrix. Having this matrix, the Hungarian [6] algorithm is used to find in polynomial time a minimum-cost bijection between the two partitions. This algorithm is typically used in linear programming to find the optimal solution to the assignment problem.

CMW [17] as well as MH-Diff [9] algorithm, are based on transforming the change detection problem in one of computing a minimum cost edge cover on a bipartite graph. Optimizations are brought out in [22] in comparison with the work in [6], but the general aim and similarity metrics remain the same as in [3].

4.4 Page Digest

The page digest model[6] has been adopted in [31] and [38], and represents a more compact encoding of data than HTML and XML formats, while preserving all their advantages. To construct this model, some steps are performed: counting the nodes, enumerating children in a depth-first manner (in order to capture the structure of the document), and content encoding and mapping for each node - encoding which will preserve the natural order of text in the Web page.

SDiff [31] is a Web change monitoring application that uses a digest format that includes also tag type and attribute information. RMS [38] also uses this model, although without giving many details. The advantages of the Page Digest over the DOM tree model are enumerated in [31]. We mention minimality and execution performance: the reduction of tag redundancy gives a more compact model, therefore a faster document traversal. The algorithms developed on this model run in linear time without making too many heuristics or restrictions, while capturing also a large palette of changes.

5. SIMILARITY METRICS

Similarity metrics are used in dynamic methods of detecting change, in the matching stage of the two versions modelled as described in section 4.

For string models, (content) change is identified when the data strings are discovered to be partially unsimilar. For tree models that have various dimensions, the problem gets more complex.

The aim is to identify model elements that are essentially the same, but which have been affected by change. Model elements that are the identical are pruned because they have basically not changed between versions; also, totally dissimilar model elements do not represent instances of the object that has evolved, so will not be included in further processing steps. Essentially, only *approximatively similar* model elements will be further studied. A typical matching is based on comparisons of attribute values of the model elements. If they have the same properties, then they are similar.

We will describe next the types of similarity metrics used for change detection in different settings or applications, for the studied cases.

[6] `http://www.cc.gatech.edu/projects/disl/PageDigest/`

5.1 String matching techniques

Approaches that compare two Web pages modeled as flat files (i.e. strings), rely on hash-based methods, edit distance metrics or techniques based on the longest common subsequence. We do not exhaustively cover all the possiblities, but rather present some of them that we have encountered in the analyzed works.

Naïve: Jaccard.

One simple technique for change detection in text is to count the number of words that are common (regardless of their position) for two string sequences and to divide the result by the number of distinct words. Another possiblity is to divide by the length of the first string, as done in [30].

Hash-based methods.

In this category, we mention [8, 20, 3]. The method of [8] uses shingling, which is a flexible method of detecting changes between two strings. It is usually referred to as w-shingling, where w the denotes the number of tokens in each shingle in the set. A *shingle* is a contiguous subsequence (w-gram) of the reference string. For the two strings to be compared, their shingles are hashed; if these strings have a lot of shingle values in common, then they are similar.

Another method is to use *signatures*. A signature of a string is a function of its hashed value. When the model is hierarchical, the signature of a node is computed based on its terminal path. For a formatting (or non-leaf) node, the signature represents the sum of the signatures of its children, until leaf nodes, where changes are actually detected. In [20], only nodes that have different signatures from those in the original version will be compared. Change detection algorithms that employ signatures have the disadvantage that false negatives are possible: change exists, but a different signature for it does not. It obviously depends on the careful choice of the space of hashing, and eventually on the application: if it can tolerate or not false results.

Another hash-based approach is presented in [3], where the change is detected at atomic element level using a direct hash.

Longest common subsequence.

A **diff** is a file comparison program that outputs the differences between two files. Diff tools are based on computing the longest common subsequence between two strings [32]. For instance, HTML*Diff* uses GNU diff utility adapted for HTML page change detection. This program treats Web pages as strings and, after processing, highlights the changes directly in a merged document; as mentioned in [15], HTML*Diff* can consume significant memory and computation resources, and this might have an influence on the scalability of the tool. Tree models of Web pages also use diff techniques, but not at HTML page level, but at subtree (or node) content level, which has the advantage of making the computation less complex. For instance, WebCQ [24] uses HTML*Diff* to

detect change at *object* level. Here, the object represents in fact a part of a Web page specified for monitoring by the user, either by means of regular expressions or by marking elements of the HTML DOM tree like a table, list, link etc..

Another example of system that uses GNU diff tool is WebVigiL [18].

Edit scripting on strings.

The edit distance between two strings of characters represents the number of operations required to transform one string into another. There are different ways of defining an edit distance, depending on which edit operations are allowed: delete, insert, etc.. In string edit scripting, the atomic element is a single character and the cost is usually unitary, for every edit operation defined.

Root Mean Square.

Another notable way [38] of compute similarity is to use RMS(Root Mean Square) value, i.e., the quadratic mean. RMS represents a statistical measure for the magnitude of a varying quantity and permits therefore to quantify the change. If its value is small, then the difference between the compared elements is not significative. Often used in engineering to do an estimation of the similarity between a canonical model and an empirical one (in order to see the precision of the experiment), RMS formula needs numeric values of the model parameters. In the HTML context, ASCII values of the Web document's text characters are utilized in the canonical formula. Although a good idea, RMS measure applied for the ASCII values of each character has some drawbacks: first it does not take into account the hierarchies (it considers that every character has equal influence, independently of its position in the page), and second, it cannot take into account the semantics of context. Variants of this measure are presented in [39].

5.2 Matching of hierarchical models

Edit scripting on trees.

has the same formal definition as for strings, excepting for the fact that the basic entities are here tree elements (nodes or subtrees). Also, edit operations can occur at different levels, depending on the types of changes considered. As a consequence, cost models of edit operations become more complex. Each edit operation has a cost associated (cost proportional to the complexity of the operation, or based on heuristics of the model), so the execution of an edit script as a sequence of operations will return a cumulated cost. The similarity measure can be then computed as the inverse of this total cost. The interpretation is the following: less *unimportant* modifications we do to the first structure of data in order to make it isomorphic with its version, the more similar the two structures are.

This problem of determining the distance between two trees is referred to as the *tree-to-tree* correction problem and this is more in depth covered in [5]. Some works [9] report that edit scripting with moving operations is NP-hard. Usually, every model introduces some kind of model or computational heuristics that make the problem (a little) less difficult. As an example, [23] computes an edit scripting on branches. It can be also done on subtrees, rather than on complete trees, for the same complexity reasons. Concerning the cost of change operations, every technique makes its own decisions. MH-Diff [9] for example defines the costs as being moderated by some constants, all depending on the type of change.

Quantitative measures of change.

Various works [17, 3, 22] use a composed measure of similarity that tries to better adapt to the specific of the types of changes considered. The change is measured by a formula that incorporates three sub-measures of specific similarity: on the content (**intersect**: the percentage of words that appear in both textual content of subtrees), attributes (**attdist**: the relative weight of the attributes that have the same value in the model elements) and on the types of elements considered in the path (**typedist** emphasizes differences on the tag names when going up in the hierarchy). The final measure incorporates all above-defined types of similarity, together with some parameters that are meant to influence the importance of certain types of changes over others. The advantage of this measure is that it captures the symbiosis of different types of changes that occur in a certain way independently: content changes in leaf nodes, attribute changes in internal nodes; the third submeasure is more focused on the position of nodes in the structure.

Another quantitative measure of change is proposed in [23]. Here, a weighted measure that determines the magnitude of the difference between two ordered multisets (i.e., branches) is employed. In an ordered multiset, the weight of the **i**th node is defined as $(2^i)^{-1}$ (where i represents the depth of an element of the branch considered). Finally, the quantity of change is measured by computing the sum of the weights of the nodes that appear in the symmetric difference set.

6. STATISTICALLY ESTIMATING CHANGE

6.1 Motivating estimative models

We have seen the multitude of domains that are interested in the change detection issue. The general aim is to get an idea about the dynamics of a certain type of content, at a given granularity.

In Web crawler-related applications, the interest is more in whether a Web page has changed or not, in order to know if a new version of a Web page shall be downloaded or not. Only a binary response is needed; this does not happen because it is not interesting to make a distinction between the different types of changes that can occur, but because current crawlers treat information at Web page level. In this case, an estimation of the change frequency is as effective as explicitly computing it, as we show in Section 3. Indeed, if archive crawlers were more aware of the semantics of data they process, they could clearly benefit of a broader, richer insight into the data and could develop different strategies related to storage and processing. An estimation of the change rate, although not very descriptive (we usually do not know where the change appeared or its type), is still useful, especially when we can imagine a strategy that combines estimative and comparative methods of deriving dynamics. For this reason, we shortly present some of the existing statistical approaches.

6.2 Poisson model

The study carried out in [10] reports the fact that changes that occur in Web pages can be modeled as a Poisson process. A Poisson process is used to model a sequence of *random*

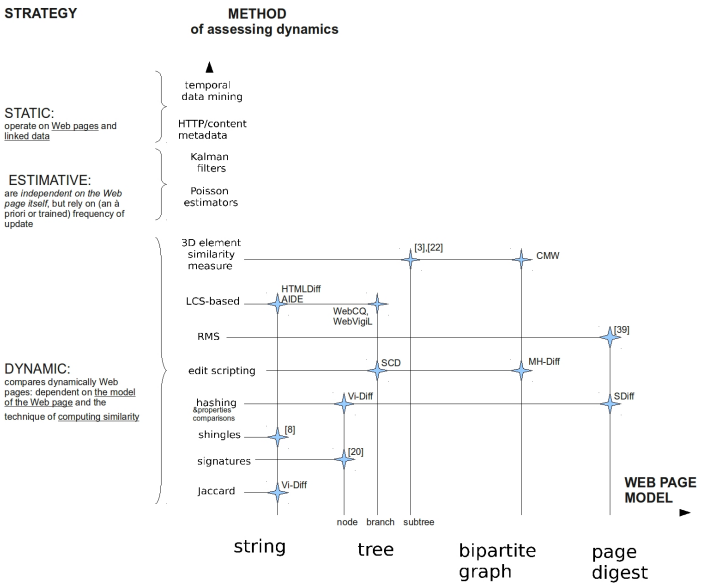

Figure 1: Summary of the presented approaches

events that happen *independently* with a fixed rate over time. Based on a(n ideally complete) change history, the frequency of change is estimated. The time-independence assumption in homogeneous models [10, 11] does not really capture the reality of the Web. The authors of [33] affirm that in the case of dynamic Web content (like blogs), the posting rates vary very much depending on different parameters. Hence, they propose an inhomogeneous Poisson model (which do not assume that events happen independently); this model learns the posting patterns of Web pages and *predicts* a recheck for new content.

The work in [11] formalizes some use cases where, by adapting the parameters of the Poisson model to the requirements of the application, a better accuracy of the estimation can be achieved. The situation when we do not have a complete change history of a Web page is treated, which is actually the real world case. Sometimes, we have only the last date of change or we at best just know that a change has occurred. Contributions are also related to the fact that the authors adapt the canonical model to interesting applications and feed different estimators, improving thus the technique for the considered cases.

6.3 Kalman filters

Other statistical approach to change detection is that of [7]. Here, the textual vector space model is employed to identify the patterns of the page and to train Kalman filters with these patterns. In the end, the change represents the event that does not match the prediction. The possible disadvantages of this method is that it assumes the linearity of the system (because of the the Kalman filter model, which is doing an exact inference in a linear dynamical system) and uses an incomplete vector space model (for complexity reasons).

7. OPEN QUESTIONS

We end this article by discussing several issues that have not been addressed yet and which are related to the deriving of temporal properties of Web pages. The selection of subjects reflects our personal vision on the topic.

Further studies on timestamping.

With the large number of sources of timestamping hints, it should indeed be possible to estimate the freshness of a Web page. An experimental study on the reliability of these sources, perhaps in specific contexts (a given Web server software, a given CMS, etc.), which could provide more insight into optimal strategies for timestamping Web pages, is still to be carried out.

Relevant change detection.

An important question when detecting changes is whether the changes are relevant to the interests or needs of the user or archivist. Web users are exposed to many advertisements and content becomes pre-fabricated, put in containers and delivered in a targeted way, so much more attention should be paid to the relevance factor. Additionally, Web pages have some components that are more dynamic than others; it is not possible to say if the dynamics come from irrelevant content (think of ads changing at every request) or precisely because the content is very informative (e.g. frequently updated news). To make the distinction between these, techniques that define and extract *semantics* (e.g., by looking at the topic and linked data) might be used as a filtering method or just for adding more significance to the change detection results. The cleaning of the Web page or its segmentation into semantic blocks is of interest in many information extraction fields and, although we mention only [40, 21, 29], there exists a large number of works that treat this subject.

As an observation, there exists a subtle difference in the use of the term "meaningful" in the various works that we have studied. While some works [23] use it to emphasize the fact that more types of changes are detected, other approaches [30] use it as synonym to "relevant", from the content point of view. Vi-Diff [30] uses the VIPS algorithm [40] to get from a Web page an hierarchy of semantic blocks and detect changes only from this perspective, hopefully ignoring all boilerplate content. However, a deeper insight into the relevance aspect is mentioned as future work; the authors of [30] talk about using machine learning techniques for this issue, which would be an interesting line of research. Usually, heuristics [40] are employed to get a measure of relevance because having a generic source of knowledge which can be automatically used for this task is very difficult.

Recently, [29] used the concepts that can be extracted from the description of Web feeds to get the content of interest from Web pages. However, this kind of semantics can only be obtained in the case of Web pages that have feeds associated.

With the emergence of the Semantic Web, we envision new ways of distinguishing timestamps or of filtering irrelevant content from Web pages and therefore more efficient methods for deriving dynamics of Web pages.

8. REFERENCES

[1] S. Abiteboul. Issues in monitoring web data. In *Proc. DEXA*, 2002.

[2] E. Adar, J. Teevan, S. T. Dumais, and J. L. Elsas. The Web changes everything: Understanding the dynamics of Web content. In *Proc. WSDM*, 2009.

[3] H. Artail and K. Fawaz. A fast HTML Web change detection approach based on hashing and reducing the

number of similarity computations. *Data Knowl. Eng.*, 66(2), 2008.

[4] R. Baeza-Yates, C. Castillo, and F. Saint-Jean. Web dynamics, structure, and page quality. In M. Levene and A. Poulovassilis, editors, *Web Dynamics*. Springer, 2004.

[5] D. T. Barnard, G. Clarke, and N. Duncan. Tree-to-tree correction for document trees. Technical Report 95-372, Queen's University, Kingston, Ontario, Canada, 1995.

[6] D. P. Bertsekas and D. A. Castañon. Parallel asynchronous Hungarian methods for the assignment problem. *INFORMS J. Computing*, 5(3), 1993.

[7] P. L. Bogen, II, J. Johnson, U. Karadkar, R. Furuta, and F. M. Shipman, III. Application of Kalman filters to identify unexpected change in blogs. In *Proc. JCDL*, 2008.

[8] A. Z. Broder. On the resemblance and containment of documents. In *Proc. SEQUENCES*, 1997.

[9] S. Chawathe and H. Garcia-Molina. Meaningful change detection in structured data. In *Proc. SIGMOD*, 1997.

[10] J. Cho and H. Garcia-Molina. The evolution of the Web and implications for an incremental crawler. In *Proc. VLDB*, 2000.

[11] J. Cho and H. Garcia-Molina. Estimating frequency of change. *ACM TOIT*, 3(3), 2003.

[12] L. R. Clausen. Concerning Etags and datestamps. In *Proc. IWAW*, 2004.

[13] G. Cobéna and T. Abdessalem. A comparative study of XML change detection algorithms. In *Service and Business Computing Solutions with XML*. IGI Global, 2009.

[14] G. Cobéna, S. Abiteboul, and A. Marian. Detecting changes in XML documents. In *Proc. ICDE*, 2002.

[15] F. Douglis, T. Ball, Y.-F. Chen, and E. Koutsofios. The AT&T Internet difference engine: Tracking and viewing changes on the Web. *World Wide Web*, 1(1), 1998.

[16] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of Web pages. In *Proc. WWW*, 2003.

[17] S. Flesca and E. Masciari. Efficient and effective Web page change detection. *Data Knowl. Eng.*, 46(2), 2003.

[18] J. Jacob, A. Sanka, N. Pandrangi, and S. Chakravarthy. WebVigiL: An approach to just-in-time information propagation in large network-centric environments. In M. Levene and A. Poulovassilis, editors, *Web Dynamics*. Springer, 2004.

[19] A. Jatowt, Y. Kawai, and K. Tanaka. Detecting age of page content. In *Proc. WIDM*, 2007.

[20] H. P. Khandagale and P. P. Halkarnikar. A novel approach for web page change detection system. *Intl. J. Comput. Theory Eng.*, 2(3), 2010.

[21] C. Kholschutter, P. Fankhauser, and W. Nejdi. Boilerplate detection using shallow text features. In *Proc. WSDM*, 2010.

[22] I. Khoury, R. M. El-Mawas, O. El-Rawas, E. F. Mounayar, and H. Artail. An efficient Web page change detection system based on an optimized Hungarian algorithm. *IEEE TKDE*, 19(5), 2007.

[23] S.-J. Lim and Y.-K. Ng. An automated change-detection algorithm for HTML documents based on semantic hierarchies. In *Proc. ICDE*, 2001.

[24] L. Liu, C. Pu, and W. Tang. WebCQ: Detecting and delivering information changes on the Web. In *Proc. CIKM*, 2000.

[25] J. Masanès, editor. *Web Archiving*. Springer-Verlag, 2006.

[26] Netcraft. January 2011 Web survey. `http://news.netcraft.com/archives/2011/01/12/january-2011-web-server-survey-4.html`, 2011.

[27] A. Ntoulas, J. Cho, and C. Olston. What's new on the Web? The evolution of the Web from a search engine perspective. In *Proc. WWW*, 2004.

[28] S. Nunes, C. Ribeiro, and G. David. Using neighbors to date Web documents. In *Proc. WIDM*, 2007.

[29] M. Oita and P. Senellart. Archiving data objects using web feeds. In *Proc. IWAW*, 2010.

[30] Z. Pehlivan, M. Ben Saad, and S. Gançarski. A novel Web archiving approach based on visual pages analysis. In *Proc. IWAW*, 2009.

[31] D. Rocco, D. Buttler, and L. Liu. Page digest for large-scale Web services. In *Proc. CEC*, 2003.

[32] S. C. Sahinalp and A. Utis. Hardness of string similarity search and other indexing problems. In *Proc. ICALP*, 2004.

[33] K. C. Sia, J. Cho, and H.-K. Cho. Efficient monitoring algorithm for fast news alerts. *IEEE TKDE*, 19(7), 2007.

[34] K. Sigurðsson. Incremental crawling with Heritrix. In *Proc. IWAW*, 2005.

[35] sitemaps.org. Sitemaps XML format. `http://www.sitemaps.org/protocol.php`, 2008.

[36] W3C. HTML 4.01 specification. `http://www.w3.org/TR/REC-html40/`, 1999.

[37] Y. Wang, D. J. DeWitt, and J.-Y. Cai. X-Diff: An effective change detection algorithm for XML documents. In *Proc. ICDE*, 2003.

[38] D. Yadav, A. K. Sharma, and J. P. Gupta. Change detection in Web pages. In *Proc. ICIT*, 2007.

[39] D. Yadav, A. K. Sharma, and J. P. Gupta. Parallel crawler architecture and web page change detection. *WSEAS Transactions on Computers*, 7(7), 2008.

[40] S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma. Improving pseudo-relevance feedback in Web information retrieval using Web page segmentation. In *Proc. WWW*, 2003.

Characterizing Search Behavior in Web Archives

Miguel Costa [1,2]
miguel.costa@fccn.pt

Mário J. Silva [2]
mjs@di.fc.ul.pt

[1] Foundation for National Scientific Computing, Lisbon, Portugal
[2] University of Lisbon, Faculty of Sciences, LaSIGE, Lisbon, Portugal

ABSTRACT

Web archives are a huge source of information to mine the past. However, tools to explore web archives are still in their infancy, in part due to the reduced knowledge that we have of their users. We contribute to this knowledge by presenting the first search behavior characterization of web archive users. We obtained detailed statistics about the users' sessions, queries, terms and clicks from the analysis of their search logs. The results show that users did not spend much time and effort searching the past. They prefer short sessions, composed of short queries and few clicks. Full-text search is preferred to URL search, but both are frequently used. There is a strong evidence that users prefer the oldest documents over the newest, but mostly search without any temporal restriction. We discuss all these findings and their implications on the design of future web archives.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process; H.3.7 [**Digital Libraries**]: User issues

General Terms

Web, Archive, Logs, User, Characterization

Keywords

Portuguese Web Archive, Search Behavior

1. INTRODUCTION

The web has a democratic nature, where everyone can publish all kinds of information. News, blogs, wikis, encyclopedias, interviews and public opinions are just a few examples. Part of this information is unique and historically valuable. However, since the web is too dynamic, a large amount of information is lost everyday. Ntoulas et al. discovered that 80% of the web pages are not available after one year [17]. In a few years they are all likely to disappear, creating a knowledge gap for future generations. Most of what has been written today will not persist and, as stated by UNESCO, this constitutes an impoverishment of the heritage of all nations [26].

Several initiatives of national libraries, national archives and consortia of organizations started to archive parts of the web to cope with this problem[1]. Some country code top-level domains and thematic collections are being archived regularly[2]. Other collections related to important events, such as September 11th, are created at particular points in time[3]. In total, billions of web documents are already archived and their number is increasing as time passes. The Internet Archive alone collected 150 billion documents since 1996. The historic interest in the documents is also growing as they age, becoming an unique source of past information for widely diverse areas, such as sociology, history, anthropology, politics or journalism. However, to make historical analysis possible, web archives must turn from mere document repositories into living archives. The development of innovative solutions to search and explore it are required.

Current web archives are built on top of web search engine technology. This seems like the logical solution, since the web is the main focus of both systems. However, web archives enable searching over multiple web snapshots of the past, while web search engines only enable searching over one snapshot of the close present. Users from both systems also have different information needs [4]. Hence, we also expected different search patterns and behaviors, which without a proper response, could degrade results and negatively influence users' satisfaction. We studied the above issues and drew the first profile of how web archive users search. It is based on the quantitative analysis of the Portuguese Web Archive (PWA) search logs [6].

Our results show that users of both types of systems have similar behaviors. In general, web search technology can be adopted to work on web archives. Nonetheless, our identification of the users' specificities provides insights on search behavior, that might contribute to better support the architectural design decisions of future web archives. Examples include optimizing their performance [1] or designing better web interfaces [8].

This paper is organized as follows. In Section 2, we cover the related work. In Section 3, we describe the search environment. The methodology of analysis is explained in Section 4 and the results are detailed in Section 5. Section 6 finalizes with the discussion of results and conclusions.

TWAW 2011, March 28, 2011, Hyderabad, India.

[1] see http://www.nla.gov.au/padi/topics/92.html
[2] see http://www.archive.org/
[3] see http://www.loc.gov/minerva/

2. RELATED WORK

2.1 Web Archive User Studies

There are several web archiving initiatives currently harvesting and preserving the web heritage, but very few studies about web archive users. The International Internet Preservation Consortium (IIPC) reported a number of possible user scenarios over a web archive [7]. The scenarios are related to professional scopes and have associated the technical requirements necessary to fulfill them. These requirements include a wide variety of search and data mining applications that have not been developed yet, but could play an important role. However, the hypothetical scenarios did not come directly from web archive users.

The National Library of the Netherlands conducted an usability test on the searching functionalities of its web archive [21]. Fifteen users participated on the test. One of the results was a compiled list of the top ten functionalities that users would like to see implemented. Full-text search was the first one, followed by URL search. Strangely, time was not mentioned in none of the top ten functionalities, despite being present in all the processes of a web archive. The users' choices can be explained by web archives being mostly based on web search engine technology. As a result, web archives offer the same search functionalities. This inevitably constrains the users' behaviors. Another explanation is that Google became the norm, influencing the way users search in other settings.

In a previous publication, we studied the information needs of web archive users [4]. We resort to three instruments to collect quantitative and qualitative data, namely search logs, an online questionnaire and a laboratory study. Our observations were coincident. Users perform mostly navigational searches without a temporal restriction. Other findings show that users prefer full-text over URL search, the oldest documents over the newest and many information needs are expressed as names of people, places or things. Results also show that users from web archives and web search engines have different information needs, which cannot be effectively supported by the same technology.

2.2 Search Log Analysis

Web usage mining focuses on using data mining to analyze search logs or other activity logs to discover interesting patterns. Srivastava et al. pointed five applications for web usage mining: personalization, for adjusting the results according to the user's profile; system improvement, for a fast and efficient use of resources; site modification, for providing feedback on how the site is being used; business intelligence, for knowledge discovery aimed to increase customer sales; and usage characterization to predict users' behavior [25]. We focus on usage characterization. However, our results can be applied to other purposes, such as the efficiency and effectiveness improvements of IR systems [23].

Search logs capture a large and varied amount of interactions between users and search engines. This large number of interactions is less susceptible to bias and enables identifying stronger relationships among data. Additionally, search logs can be analyzed at low cost and in a non intrusive way. Most users are not aware that their interactions are being logged. Users also try to fulfill their real information needs, instead of having tasks assigned by a researcher that can bias their behaviors. On the other hand, search logs are limited to what can be registered. They ignore the contextual information about users, such as their demographic characteristics, the motivations that lead them to start searching, and their degree of satisfaction with the system. Qualitative studies, such as surveys and laboratory studies, can complement log analysis with information that can explain some of the patterns found [14].

Several logs from web search engines were analyzed with the goal of understanding how these systems were used. A common observation across these studies is that most users conduct short sessions with only one or two queries, composed by one or two terms each [11]. When users submit more than one query, they tend to refine the next query by changing one term at a time. Most users only see the first search engine results page (SERP) and rarely use advanced search operators. These discoveries imply that the use of web search engines is different from traditional IR systems, which receive queries three to seven times longer [12]. Queries for special topics (e.g. sex), special types (e.g. question-format) and multimedia formats (e.g. images) are also longer [16]. This shows that the users' behavior varies not only between IR systems, such as search engines, online catalogs and digital libraries, but also depends on the type of information and the way users search. Another aspect that differentiates search behavior is users' demographics (i.e. age, gender, ethnicity, income, educational level) [27].

3. THE SEARCH ENVIRONMENT

The PWA preserves the Portuguese web, which is considered the subset having the most interesting contents for the Portuguese community. Specifically, we define the Portuguese web as all the documents[4] satisfying one the following rules: (1) hosted on a site under a .PT domain; (2) hosted on a site under another domain, but embedded in a document under the .PT domain; (3) suggested by the users and manually validated by the PWA team. Additionally, the PWA team integrated web collections from several other sources, such as the Internet Archive and the Portuguese National Library. The number of indexed documents have been growing and there are now more than 180 million accessible by full-text and URL search. As far as we know, this is the largest web archive collection searchable by full-text and over such a large time span (from 1996 to 2009). The experimental version of the PWA has been available as a service to the general public since 2010 at `http://archive.pt/`.

The interaction with the users and the layout of the results is similar to web search engines, such as Google. In a typical session, a user can submit a full-text query and receive a search engine results page (SERP) containing a list of 10 results matching the query. Figure 1 illustrates this case. Each result includes the title of the web page and its crawled date, a snippet of text containing the query terms and the URL. The user can then click on the results to see and navigate in the web pages as they were in the past. If the desired information is not found, the user can repeatedly modify and resubmit the query. In addition, the user can click on the navigation links to explore other SERPs or use the advanced search interface to restrict the query with advanced search operators. These operators can also be added to the query directly in the text box.

[4]The terms document and file are used interchangeably in this study. For instance, it can be a web page, an image or a PDF file.

Figure 1: Search interface after a full-text search.

Figure 2: Search interface after an URL search.

This interface has some specificities. First, the text box is complemented with a date range filter to narrow the results to a time period. Second, each result has an associated link to see all versions throughout time of the respective URL. When clicked, the PWA presents the same search engine versions page (SEVP) as when a user submits that URL on the text box. A table is shown to the user, where each column contains all the versions of a year sorted by date. The user can then click on any version to see it as it was on that date. Figure 2 depicts this interface.

3.1 Logs Dataset

Our analysis is based on the logs of the PWA, covering seven months of search interactions, from June to December, 2010. By interactions, we mean all queries and clicks submitted by the users and recorded by the PWA search engine (server side). The seven month span has the advantage of being less likely to be affected by ephemeral trends.

The logs follow the Apache Common Log Format[5]. Each entry corresponds to an interaction with the search engine in the form of a HTTP request. It contains the user's IP address and the user's session identifier. Each entry contains also a timestamp indicating when the interaction occurred and the HTTP request line that came from the client.

We never used the log data to match a real identity. However, we geographically mapped the IP addresses for a better characterization of the users. We counted 72% of PWA's users with IP addresses assigned to Portugal. Near 89% of the interactions were submitted through the Portuguese language interface. The remaining was submitted through the English language interface. This strongly indicates that users were mostly Portuguese.

[5]see `http://httpd.apache.org/docs/2.0/logs.html`

4. METHODOLOGY

The analysis focused on four dimensions: sessions, queries, terms and clicks. We define them in the following way:

- A *session* is a set of interactions by the same user when attempting to satisfy one information need. The session is the level of analysis in determining the success or failure of a search. It is composed by one or more queries and zero or more clicks.

- A *query* is a search request composed by a set of terms. We define an *initial query* as the first query submitted in a session, while all the following queries are defined as *subsequent*. An *identical query* is a query with exactly the same terms as the previous one submitted in the same session. A *unique query* corresponds to one query regardless of the number of times it was logged. The set of unique queries is the set of query variations. An *advanced query* is a query with at least one advanced search operator.

- A *term* is a series of characters bounded by white spaces, such as words, numbers, abbreviations, URLs, symbols or combinations between them. There are also advanced search operators, but they do not count as

terms. We define a *unique term* as one term on the dataset regardless of the number of times it was logged. The set of unique terms is the submitted lexicon.

- A *click* in this context refers to the following of a hyperlink to immediately view a query result (i.e. archived web page). It can be a *SERP click* or a *SEVP click*, depending if the user clicks in a SERP or SEVP.

Next, we briefly present the methods used on the search log analysis.

4.1 Log Preparation

We prepared the log fields for analysis through a series of data cleansing steps. All incomplete entries, empty queries and sessions without any query were discarded. Internal queries submitted by the PWA monitoring system, the queries by example displayed on the PWA entry page and sessions conducted by clients identified as web crawlers were also excluded. Additionally, sessions with more than 100 queries were likely to come from crawlers, so they were removed too. This cutoff value of 100 was used in some other studies, thus enabling a more direct comparison with our results [10]. The queries that resulted from navigation clicks to see another SERP were not counted as a new query. These are the same queries parameterized to show more results.

All terms were normalized to lowercase. Extra white spaces were removed. Since the PWA did not perform stemming, all variations of a query term were considered as different terms. The set of query terms also includes misspellings.

4.2 Session Delimitation

Most studies used the users' IP address and/or session identifier to delimit sessions [11]. We used these two parameters to track and delimit user interactions. We also used a time interval t of inactivity to delimit sessions. Two consecutive interactions are included in different sessions if they have an inactivity between them of at least t. Without this gap, we could have sessions of several days, which would hardly represent the reality. Studies diverge on the choice of this interval, from 5 to 120 minutes [11], while others argue that no time boundary is effective in segmenting sessions [13]. We selected the 30 minute interval, because this interval has shown to produce good results, close to the results produced by SVM classifiers that were designed for delimiting sessions [20].

5. LOG ANALYSIS

Statistics were computed from the logged interactions. The first pattern that we detected was that users mostly conducted two types of sessions: with only full-text queries and with only URL queries, in 59.34% and 31.10% of the times, respectively. We defined these as *full-text sessions* and *URL sessions*. In the analysis, we ignored the remaining 9.57% sessions with mixed queries for simplification.

Table 1 shows the general statistics. The users of the PWA performed 6,177 full-text sessions, averaging 2.23 queries per session. The number of query terms per query was 2.84, with 6.42 characters per term. The users saw 1.44 SERPs per query and clicked 1.06 times on their hyperlinks to view a result. They hardly clicked to see all versions of a result. This only happened in 0.06 times per query. Overall, these results mean that for each query, the users saw mostly the first and sometimes the next SERP, where they clicked once.

	full-text	URL
Sessions	6,177	3,237
Queries	13,770	4,986
Terms	39,132	-
SERPs	19,812	-
Clicks on SERPs	14,664	-
Clicks on SEVPs	-	3,861
Queries per Session	2.23	1.54
Terms per Query	2.84	-
SERPs per Query	1.44	-
Clicks on SERP per Query	1.06	-
Clicks on SEVP per Query	-	1.56
Characters per Term	6.42	27.27
Initial Queries	44.86%	64.92%
Subsequent Queries	55.14%	35.08%
- Modified	44.53%	-
- Identical	20.35%	21.44%
- Terms Swapped	3.75%	-
- New	31.37%	78.56%
Unique Queries	68.82%	73.95%
Unique Terms	26.66%	-
Queries never repeated	54.38%	59.99%
Terms never repeated	13.88%	-

Table 1: General statistics.

Session duration	% full-text sessions	% URL sessions
$[0, 1[$	59.93%	81.19%
$[1, 5[$	23.07%	12.42%
$[5, 10[$	6.22%	2.97%
$[10, 15[$	2.77%	1.95%
$[15, 30[$	4.95%	1.02%
$[30, 60[$	2.19%	0.46%
$[60, 120[$	0.73%	0.00%
$[120, 180[$	0.10%	0.00%
$[180, 240[$	0.05%	0.00%
$[240, \infty[$	0.00%	0.00%

Table 2: Session duration (minutes).

The users also submitted 3,237 URL sessions, roughly half of the full-text seassions. On average, each session had 1.54 queries with 27.27 characters. Half of the URLs submitted, 50.24%, were not found in the PWA. For the URLs found, the users clicked on 1.56 versions to see them as they were on past. Basically, a user submitted a URL and saw one or two versions of that URL. Next, we will detail our analysis and explain the remaining results.

5.1 Session Level Analysis

5.1.1 Session duration

The duration of a session is measured from the time the first query is submitted until the last time the user interacted with the PWA. We ignore if the user spent more session time viewing the archived web pages after the last interaction or used part of the time doing parallel tasks [19]. We assigned a 0 minutes duration to sessions composed by only one query.

The large majority of sessions ended quickly as shown in Table 2. Around 60% of the full-text sessions lasted less than 1 minute and 89% less than 10 minutes. Only around 3% of the sessions had a longer than an half hour duration. Each session took in average 4 minutes and 8 seconds. URL sessions took even less time than full-text sessions. In average, each session took 1 minute and 14 seconds. Around 81% of the sessions lasted less than 1 minute and only 6% took longer than 5 minutes.

# queries	% full-text sessions	% URL sessions
1	64.98%	72.10%
2	12.53%	15.57%
3	7.48%	6.21%
4	5.00%	3.06%
5	2.72%	1.11%
6	1.65%	0.56%
7	1.12%	0.74%
8	0.68%	0.28%
9	1.26%	0.19%
≥10	0.58%	0.09%

Table 3: Number of queries per session.

# terms	% modified queries
≤-5	1.51%
-4	1.33%
-3	3.46%
-2	6.12%
-1	13.04%
0	32.21%
+1	25.64%
+2	10.12%
+3	3.11%
+4	2.13%
≥+5	1.33%

Table 4: Number of terms changed per modified full-text query.

5.1.2 Query distribution

Table 3 shows that the majority of the users only submitted one query. Around 85% of the full-text sessions had up to 3 queries and less than 3% had 10 or more queries. This last number can represent highly motivated users searching for special topics (e.g. sex) [16].

When users submitted URL sessions, 72% were composed by only one query, while 94% up to three queries. Only 2% had five or more queries. An URL query is a very specific query, where users know exactly what they are searching for. This can explain why users submitted less queries than in full-text sessions.

5.2 Query Level Analysis

5.2.1 Modified queries

Sometimes users submit sequences of queries as a way to refine or reformulate the search in a trial and error approach. We consider that two sequential queries submitted on the same session have the same information need if they share at least one term. In this case, we called the second query a modified query. We ignored the stopwords (too common terms) in this analysis. Thus, a modified query could be a specialization of the query (adding terms), a generalization (removing terms) or both at the same time.

We counted 44.53% of modified queries from all subsequent full-text queries. Looking at Table 4, we see that around 71% of the modified queries are the result of a zero or one change on the number of terms. A zero length change means that the users modified some terms, but their number remained the same. Users tend to add more terms in the modified queries rather than to remove them. We counted around 42% versus 25%. PWA's users tend to go from broad to narrow queries, such as in web search engines [3, 12, 22].

advanced operator	% advanced queries	% total queries
NOT	3.62%	0.94%
PHRASE	78.10%	20.20%
SITE	12.81%	3.31%
TYPE	5.48%	1.42%
total	100.00%	25.86%

Table 5: Advanced operators per full-text query.

5.2.2 Identical and New queries

A variety of reasons can lead users to repeat queries, such as a refresh of the SERP or SEVP, a back-button click or the submission of the same query more than once due to a network or search engine delay. When analyzing the full-text queries, we counted 20.35% of identical queries, where each query has exactly the same terms as the previous one made in the same session (see Table 1). We also counted the subsequent queries with the same terms, but written in a different order. For instance, a query *Web Archive* followed by a query *Archive Web*. Only a small number of subsequent queries, 3.75%, had the order of the terms swapped. Besides the modified and identical queries, the users also submitted 31.37% of subsequent queries with only new terms. This indicates that at most this percentage of subsequent queries were the result of a new information need.

We divided the subsequent URL queries in identical and new. 78.56% of the subsequent queries were new. The remaining 21.44% were the result of the same URL submission (see Table 1).

5.2.3 Advanced queries

In the PWA, users could use four advanced search operators: NOT, to exclude all results with a term in their text (e.g. *-web*); PHRASE, to match all results with a phrase in their text (e.g. *"web archive"*); SITE, to match all results from a domain name (e.g. *site:wikipedia.org*); TYPE, to match all results from a media type (e.g. *type:PDF*).

Table 5 presents the percentages of advanced queries (i.e. with at least one advanced search operator). It shows that 25.86% of the queries included operators. This is a significantly higher percentage when compared with studies over web search engines [9, 12, 22]. The reason is the PHRASE operator, which represents 78.10% of the choices. The PWA suggested a URL within quotes for each URL submitted, to inform the users that they could match the URL in the text. However, even when ignoring the URLs within quotes, the percentages are roughly the same. The second most used operator was the SITE, occurring in 12.81% of the advanced queries. The TYPE and NOT operators were insignificantly used when compared to the total number of queries.

5.2.4 Term distribution

The distribution of the terms per full-text query listed in Table 6 shows that the majority of the queries had 1 or 2 terms. This is also visible by the 2.84 average of terms per query (see Table 1). Around 87% of the queries had up to 5 terms and only 3% had 10 or more terms. These results indicate that the users tend to submit short queries. These values are useful, for instance, to optimize index structures [15] or to determine the adequate length of the input text boxes on the user interface [8].

# terms	% full-text queries
1	35.77%
2	24.99%
3	15.14%
4	7.54%
5	3.55%
6	4.47%
7	2.40%
8	1.92%
9	1.46%
≥10	2.77%

Table 6: Number of terms per query.

SERP viewed	% full-text queries
1	100.00%
2	14.44%
3	8.08%
4	5.29%
5	3.75%
6	2.88%
7	2.33%
8	1.72%
9	1.59%
≥10	3.79%

Table 7: SERPs viewed per query.

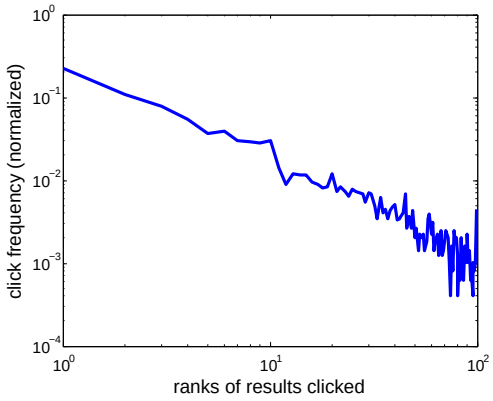

Figure 3: Distribution of ranks clicked on SERPs.

5.2.5 SERPs

The users saw on average about 1.44 SERPs per full-text query. All users saw the first SERP as expected, since the PWA always returned it after a query. Then, the users followed the natural order of the SERPs, but in a sharp decline (see Table 7). For instance, the second SERP was viewed in 14.44% of the queries. This indicates that prefetching the second SERP would not significantly improve web archive performance. On the other hand, the close percentages of the following SERPs indicate that prefetching them can bring improvements as shown in other studies [5].

5.2.6 Clicks on SERPs

About 66% of the clicks occurred on the first SERP from almost a click per query. The users clicked on 1.06 times per query to access an archived web page listed on the SERPs. We observed that users clicked on the rank of results following a power law distribution, with a 0.88 correlation (see Figure 3). These results are similar to web search engine studies, which also present a discontinuity in the last ranking position of each SERP (multiple of 10) [2].

5.2.7 Query frequency distribution

We ranked the full-text unique queries by their decreasing frequency and verified that their distribution fits the power law with a 0.96 correlation. This finding was also observed in web search engines [1, 5]. It means that a small number of queries was submitted many times, while a large number of queries were submitted just a few times. Figure 4 depicts the cumulative distribution of queries. For instance, by caching around 27% of the most frequent queries, the PWA could respond to 50% of the total query volume.

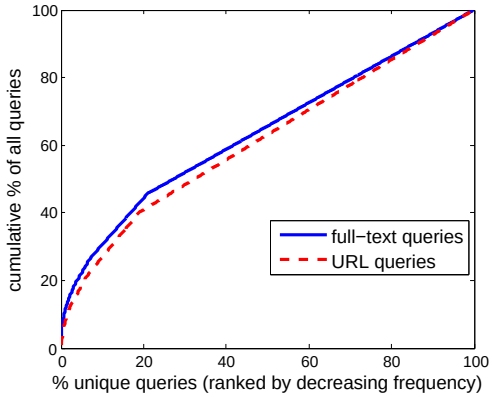

Figure 4: Cumulative distributions of queries.

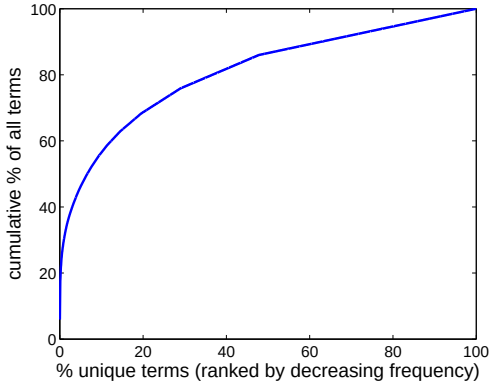

Figure 5: Cumulative distribution of full-text terms.

We also ranked the URL unique queries by their decreasing frequency and verified that their distribution, once again, fits the power law with a 0.96 correlation. By caching around 32% of the most frequent URL queries, the PWA could respond to 50% of the queries. Although satisfactory, the percentage of queries cached are much superior than in previous studies [3]. This is likely due to the small number of sessions analyzed, which leads to a reduced repetition.

As a consequence of the users' queries and clicks following a power law distribution, the archived pages seen by the users also follow a power law distribution, with a 0.94 correlation. This applies to both full-text and URL sessions.

5.3 Term Level Analysis

5.3.1 Term frequency distribution

Analogous to the query frequency distribution, we ranked the full-text unique terms by their decreasing frequency. Their distribution fits the power law with a 0.97 correlation. As depicted in Figure 5, the cumulative distribution shows that it is necessary to cache just around 6% of the most frequent terms to handle 50% of the queries. Much less RAM is necessary to cache terms than queries for a similar hit rate. These results are consistent with others presented for web search engines [1, 3]. However, caching the terms instead of the queries, adds extra processing over the posting lists of the inverted index, to evaluate the documents matching the query. A proper trade-off must be found.

restriction	% full-text queries	% URL queries
start date	1.64%	1.34%
end date	23.55%	30.16%
start & end date	12.98%	4.88%

Table 8: Queries restricted by date.

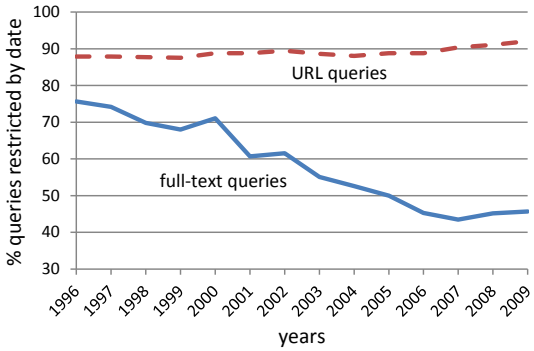

Figure 6: Distribution of years included in queries restricted by date.

5.4 Temporal Level Analysis

5.4.1 *Queries restricted by date*

The users restricted by the end date 23.55% of the full-text queries, while only 1.64% by the start date. The start and end dates were both changed in 12.98% of the queries. The same pattern exists in URL queries as shown in Table 8, where the start date was changed almost only when the end date also was. This indicates that users are more interested in old documents. The idea is reinforced by the distribution of the years included in the full-text queries restricted by date. As it can be seen in Figure 6, the older the years, the more likely they are of being included in queries. However, the URL queries have an almost constant rate.

5.4.2 *Clicks on temporal versions*

Documents tend to have just a few years with archived versions, thus segmenting the number of clicks per year would likely bias the results. Instead, we computed for all URL queries, the percentage of clicks in each year y_i with at least one version. We measured as $\frac{clicks(y_i)}{times(y_i)}$, where the denominator represents the number of times the year y_i was displayed to the user, and the numerator the number of clicks in y_i. For instance, the first year y_1 is 1997 if there is no archived versions for that URL in 1996. Otherwise, y_1 is 1996.

In Figure 7 it is visible that users clicked much more on the first year with archived versions than on the remaining years. The first year was clicked in 55% of the times, while all the others were clicked at most 20%. With exception of the eighth year, the first three years had the higher percentages. This shows a preference for the older documents.

5.4.3 *Implicit temporal queries*

We counted the number of queries with temporal expressions, since they represent a temporal dependent intent. We started by experimenting named-entity recognition tools for Portuguese. However, queries are not grammatical, so the tools presented a small precision. Instead, we used a simple match of all the queries with years, months and day patterns. Then, we classified a random subset of 1,000 queries to validate our detection patterns. Surprisingly, they worked very well. The patterns achieved a precision, recall and accuracy, of 89%, 100% and 98%, respectively. The patterns created some false positives, but unexpectedly no false negatives. This was mostly, because there were no temporal expressions in the logs without date patterns (e.g. last decade).

All matches were manually validated, from which we excluded the false positives. In the end, we counted 3.49% of queries with temporal expressions. Almost all are related with past events, such as *world cup 2006*. This is a small percentage in line with the 1.5% of temporal expressions found in the logs of the AOL web search engine [18].

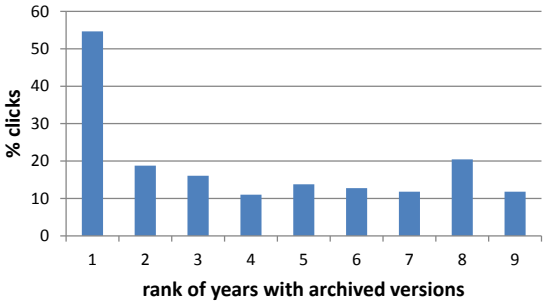

Figure 7: Clicks on years with archived versions (from oldest to newest).

6. DISCUSSION AND CONCLUSIONS

Search patterns from users of web archives and web search engines are contrasted in Table 9. Web archive users submit more single query sessions, which reflects in a smaller number of queries per session. In a nutshell, web archive users iterate less. This can be explained by most of the information needs of web archive users being navigational, contrary to the needs of web search engine users [4]. Web archive users search for known-items using names, titles and URLs, some within quotes, that give good clues of the desired information. Another explanation is that web archive users submit longer queries, which could lead to better results.

On the other hand, the single term queries, the SERPs viewed per query and the topic most seen, are in conformity with web search engine results [3, 10, 11, 12, 16]. The classification of searched topics in web archives followed a different taxonomy, so they are not directly comparable. Still, Commerce is the most searched topic for navigational queries and People for informational queries [4].

Overall, the search patterns of the users of both types of systems show no evidence precluding the adoption of web search engine technology for web archive search. This was a surprise to us, because users from both systems have different information needs. For instance, users said they wanted to see the evolution of a page throughout time, but they tend to click on one or two versions of each URL. All information needs of the users are focused on the past, but most of the user queries are not restricted by date, neither contain temporal expressions. Users search as in web search engines. This behavior may be the consequence of we having offered a similar interface, leading them to search in a similar way.

IR system	web search engine			web archive
world region	U.S.	Europe	Portugal	Portugal
name	**Excite** [11, 24]	**FAST** [11, 24]	**Tumba!** [3]	**PWA** [4] & this study
single query session	55%-60%	53%-59%	41%-50%	65%
queries per session	2.3	2.9	2.5 - 2.9	2.2
single term queries	20% - 30%	25% - 35%	40%	36%
terms per query	2.6	2.3	2.2	2.84
advanced queries	11% - 20%	2% - 10%	11% - 13%	26%
SERPs viewed per query	1.7	2.2	1.4	1.4
topic most seen	Commerce, Travel	People, Places	Commerce, Travel	Commerce & People

Table 9: General comparison between users from web search engines and web archives.

Hence, new types of interfaces must be experimented, such as the temporal distribution of documents matching a query or timelines, which could create a richer perception of time for the user and eventually trigger different search behaviors.

Nevertheless, the identification of the users' specificities might contribute to the development of better adapted web archives. We observe a strong preference in searching and seeing the oldest documents over the newest. This finding can be used in ranking results, when no other temporal data is given. The ranking should also be tuned for navigational queries when the query type is unknown. Queries, terms, clicked ranks and seen archived pages follow a power law distribution. This means that all have a small fraction that is repeated many times and can be explored to increase the performance of web archives.

The PWA is still experimental and has a much smaller user base than commercial web search engines. Still, we believe that the obtained results are general, but studies over larger datasets and from other web archives are necessary to confirm this. Our future work will use these results to improve the architecture and retrieval algorithms of the PWA.

7. ACKNOWLEDGMENTS

This work could not be done without the help and infrastructure of the PWA team. We thank Michel da Corte for her review of the paper and FCT (Portuguese research funding agency) for its Multiannual Funding Programme.

8. REFERENCES

[1] R. Baeza-Yates, A. Gionis, F. P. Junqueira, V. Murdock, V. Plachouras, and F. Silvestri. Design trade-offs for search engine caching. *ACM Transactions on the Web*, 2(4):1–28, 2008.

[2] R. Baeza-Yates, C. Hurtado, M. Mendoza, and G. Dupret. Modeling user search behavior. In *Proc. of the 3rd Latin American Web Congress*, page 242, 2005.

[3] M. Costa and M. J. Silva. A search log analysis of a Portuguese web search engine. In *Proc. of the 2nd INForum - Simpósio de Informática*, pages 525–536, 2010.

[4] M. Costa and M. J. Silva. Understanding the information needs of web archive users. In *Proc. of the 10th International Web Archiving Workshop*, pages 9–16, 2010.

[5] T. Fagni, R. Perego, F. Silvestri, and S. Orlando. Boosting the performance of web search engines: Caching and prefetching query results by exploiting historical usage data. *ACM Transactions on Information Systems*, 24(1):51–78, 2006.

[6] D. Gomes, A. Nogueira, J. Miranda, and M. Costa. Introducing the Portuguese web archive initiative. In *Proc. of the 8th International Web Archiving Workshop*, 2008.

[7] A. T. W. Group. Use cases for access to Internet Archives. Technical report, Internet Preservation Consortium, 2006.

[8] M. Hearst. *Search User Interfaces*. Cambridge University Press, 2009.

[9] C. Hölscher and G. Strube. Web search behavior of Internet experts and newbies. *Computer networks*, 33(1-6):337–346, 2000.

[10] B. Jansen and A. Spink. An analysis of Web searching by European AlltheWeb.com users. *Information Processing and Management*, 41(2):361–381, 2005.

[11] B. Jansen and A. Spink. How are we searching the World Wide Web? A comparison of nine search engine transaction logs. *Information Processing and Management*, 42(1):248–263, 2006.

[12] B. J. Jansen, A. Spink, and T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the Web. *Information Processing and Management*, 36(2):207–227, 2000.

[13] R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proc. of the 17th ACM Conference on Information and Knowledge Management*, pages 699–708, 2008.

[14] D. Kelly. *Methods for evaluating interactive information retrieval systems with users*, volume 3 of *Foundations and Trends in Information Retrieval*. Now Publishers Inc., 2009.

[15] C. Lucchese, S. Orlando, R. Perego, and F. Silvestri. Mining query logs to optimize index partitioning in parallel web search engines. In *Proc. of the 2nd International Conference on Scalable Information Systems*, pages 1–9, 2007.

[16] K. Markey. Twenty-five years of end-user searching, Part 1: Research findings. *American Society for Information Science and Technology*, 58(8):1071–1081, 2007.

[17] A. Ntoulas, J. Cho, and C. Olston. What's new on the web?: the evolution of the web from a search engine perspective. In *Proc. of the 13th International Conference on World Wide Web*, pages 1–12, 2004.

[18] S. Nunes, C. Ribeiro, and G. David. Use of temporal expressions in web search. In *Proc. of the Advances in Information Retrieval, 30th European Conference on IR Research*, pages 580–584, 2008.

[19] S. Ozmutlu, H. Ozmutlu, and A. Spink. Multitasking Web searching and implications for design. *American Society for Information Science and Technology*, 40(1):416–421, 2003.

[20] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *Proc. of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 239–248, 2005.

[21] M. Ras and S. van Bussel. Web archiving user survey. Technical report, National Library of the Netherlands (Koninklijke Bibliotheek), 2007.

[22] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. In *ACM SIGIR Forum*, volume 33, pages 6–12, 1999.

[23] F. Silvestri. *Mining query logs: Turning search usage data into knowledge*, volume 4 of *Foundations and Trends in Information Retrieval*. Now Publishers Inc., 2010.

[24] A. Spink, S. Ozmutlu, H. C. Ozmutlu, and B. J. Jansen. U.S. versus European Web searching trends. *SIGIR Forum*, 36(2):32–38, 2002.

[25] J. Srivastava, R. Cooley, M. Deshpande, and P. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *ACM SIGKDD Explorations Newsletter*, 1(2):23, 2000.

[26] UNESCO. Charter on the Preservation of Digital Heritage. Adopted at the 32nd session of the General Conference of UNESCO, October 17, 2003. `http://portal.unesco.org/ci/en/files/13367/10700115911Charter_en.pdf/Charter_en.pdf`.

[27] I. Weber and C. Castillo. The demographics of web search. In *Proc. of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 523–530, 2010.

Changing Vision for Access to Web Archives

Zeynep Pehlivan
LIP6
University P. and M. Curie
Paris, France
zeynep.pehlivan@lip6.fr

Anne Doucet
LIP6
University P. and M. Curie
Paris, France
anne.doucet@lip6.fr

Stéphane Gançarski
LIP6
University P. and M. Curie
Paris, France
stephane.gancarski@lip6.fr

ABSTRACT

Since late 90s, there has been a large investment in web archiving. Accessing these huge information sources is getting more and more attention. Web archive users profiles differ from casual web users profiles. Archive users need to analyze, evaluate and compare the information which requires complex queries with temporal dimension. These queries can not be performed by currently proposed access methods: wayback machine, full-text search and navigation. In this paper, we address this requirement by proposing a data model and a temporal query language for web archives which take into account different topics in web pages and the issues related to web archiving.

In our approach, a captured web page is visually segmented into semantic blocks. A concrete block notion is introduced to represent these different semantic blocks. A concrete block is a triplet: frame block which keeps properties of a block, the content (textual and:or non-textual) and the importance accorded to a block. Each of them is timestamped with a period called *validity*. A web page, identified with an url, is a set of concrete blocks and a web site is a set of pages. Pages and sites are generated dynamically by manipulating concrete blocks when needed. Operators for data manipulation, navigation and ranking are also proposed.

Keywords

Web Archiving, Access to web archives, Temporal Query Languages, Temporal Data Model

1. INTRODUCTION

Web archives, in short *WAC*s, aim to preserve the history of large portions of the web. They represent thus a huge information source, potentially greater than the web itself. This makes web archiving an active research area with numerous opened issues like crawler optimization, storage models, etc. An overview of main issues is presented by Masanès [21]. It is interesting to notice that the researches on accessing web archives focus on extension of existing access methods to web.

For instance, the "Wayback Machine" [27] which allows users to see captured versions of web pages over time, is the best known access method to web archives. The others methods currently provided are well-known web access methods: full-text search and navigation within the requested time range. These methods are powerful for casual users, who search the web for general information and represent the largest proportion of web users. According to the "Web Archiving User Survey" [24], the web archive users profiles consist of historians, journalists, lawyers, students etc. more than casual users. It is also underlined that the main reason for using web archives is research activity. Web archive users need to analyze, compare and evaluate the information. In order to achieve this, web archive systems must provide tools to execute complex queries.

Consider a researcher who studies how French media covered the event "earthquake at Haiti in 2010" over last year. At the very beginning of her research she would like to know the number per month of web pages, in domain *.fr*, referring to earthquake. This kind of queries can not be performed by wayback machine, full-text search or navigation methods but with an efficient query language. A number of possible user scenarios over web archives are listed and illustrated with technical requirements in "Use cases for access to Internet Archives" [6]. A significant number of them requires complex query capability to be handled.

Multi-topics and noisy information on a web page affect the search performance. In recent years, there has been an increasing interest in web based searching by taking these different topics as units of retrieval and by eliminating noisy information like copyrights, navigation bars etc. (e.g [12, 15, 28]). To achieve this, different page segmentation algorithms are proposed (e.g [20, 14, 16]). They partition web pages into non-overlapping hierarchical blocks where each block deals with a different topic or only contains noisy information. As web archives are temporal collections of web pages, block-based approach needs to be extended with temporal dimension in order to be used in web archive search.

The role of search engines for web users is non-negligible. On the other hand, as seen in use cases [6], most of the time, they are not sufficient to meet the need of web archive users. New approaches to explore web archives with complex queries are needed. An appropriate query language, that enables temporal search besides content queries and structural queries (not only for the hypertext structure, also for web page's internal structure) is especially required for formulat-

TWAW 2011, March 28, 2011, Hyderabad, India.

ing complex queries. There are a number of query languages proposed for querying web data (e.g WebSQL, W3QL, WebLog, WebOQL, etc.) but most of them are not suitable for web archives due to a lack of temporal dimension and a lack of handling challenges related to web archives. Actually, our aim is to integrate the database approach and the IR approach.

In this paper, we present a conceptual model for web archives and basics of a query language based on this model. In our approach, visual blocks, which are extracted from a page, are used as unit of retrieval rather than a whole page. Each element of the model is timestamped with a period called *validity*. Operators for data manipulation, navigation and ranking are proposed. The query language that we propose for web archives:

- Enables block-based search
- Takes into account incompleteness
- Eliminates duplicates
- Enables temporal ranking and grouping
- Is user-friendly

We organize our paper as follows. In Section 2, we present the features of the query languages for web archives. In Section 3, we present the related works. In the following section, conceptual model is presented with related operators. Before the conclusion in Section 5, two use cases are explored in Section 5.

2. FEATURES OF WAC QUERY LANGUAGE

The overarching design goal of our approach is to offer a query language for web archive users. In this section, we describe the features of this query language and the rationale behind them.

Block-Based Search: Today, web pages contain various topics. A typical example is the web pages of newspapers/TV channels like www.bbc.co.uk/news (Figure 1) where multiple blocks with unrelated topics are marked with different colors. A web page with matched query terms in the same region is more relevant than a web page with matched terms distributed over the entire page. Besides increasing keyword search performance, the segmentation gives a structure to the web page for structural queries.

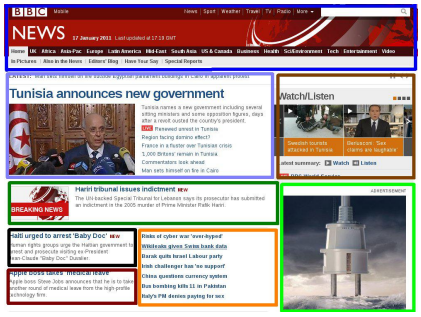

Figure 1: Multi-topics in a web page

Previous works [20, 14] show that a page can be partitioned into multiple blocks and, often, the blocks in a page have a different importance. The importance weights are accorded to different blocks inside a web page according to their location, area size, content, etc. Using a web page as unit of retrieval does not take into account these different regions and their importance. In our model, we add this notion to enrich our query language. With the block based search and the importance, we are able to answer queries like "Find pages mentioning Obama and Sarkozy in same block" or "Find pages mentioning Obama in their most important blocks" which returns more relevant information and also helps to reduce the number of results.

Incompleteness and Temporal Coherence: Due to the limited sources, all web archives are incomplete (i.e they do not contain all possible versions of all the pages on the web) and we should query them as they are. If a user asks for a version at t, and if the archive does not have versions at t but it has the versions at *t-2* and at *t+2*, the access model should decide or support different choices to get the closest version to t.

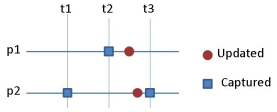

Figure 2: Temporal Coherence

Temporal coherence is another issue in web archiving. The main reason is the dynamic structure of the web. It changes continuously in an unpredicted and unorganized manner. The web sites politeness constraints and the limited allocated resources do not allow archiving a whole web site at once, at the same moment. For example, in Figure 2, for a temporal navigation starting from the version of p1 at *t2*, p2 at *t1* is coherent, while p2 at *t3* is incoherent. The access model should be able to find the most coherent version.

Temporal coherence and incompleteness lend to broken or defected links which disable a complete navigation and bring access to a standstill. An access model for web archives should take into account these issues.

Duplicates: Web archives has much more duplicated contents than the web itself. In web archives, that kind of duplicates can occur in two different cases: two versions of the same URL crawled at different times or the same content is pointed by several different URLs. According to [18], 25% of the documents in web archives are exact duplicates. Duplicates complicate the search and the result visualization.

Temporal Ranking and Grouping: Ranking and grouping are most common ways to deliver query results for the large-scale data. For web archives, temporal dimension must be included in both ranking and grouping process. Ranking has also a dynamic nature in WACs. For example, a page archived mentioning "Obama" at 1999 should not have the same ranking result when querying at 2000 and at 2010.

Temporal Logic: Support for temporal logic operators enriches the language. For example, a researcher who wants to analyze the effects of the arrest of Julian Assange can execute a query: "Find pages linked to wikileaks.org after Julian Assange's arrest in London".

User-Friendliness: This query language will be used by researchers and casual web users. So it should enable users to find information without long-term training. Simple queries (like keyword search) should be expressed straightforwardly. Complex syntax should be used only to express more complex queries. We believe that with an advanced GUI, users can avoid from writing "codes".

3. RELATED WORKS

In this section, we briefly summarize related works in three different areas concerning the access to web archives: Web archiving, query languages for web and block-based search

Web Archiving

To explore web archives, traditional access methods, i.e navigation and full-text search are proposed by web archiving initiatives [3, 4, 5]. In fall of 2001, Internet Archive(IA) [3] launched its collaborated project with Alexa Internet called "Wayback Machine" [27]. It allows users to go back in time and view earlier versions of a web page for a given URL. The inconvenience of this method is the necessity of knowing exact URLs. There is another way of navigating in web archives : navigation between different versions. Some web archive initiatives propose a navigation tool like UK Web Archives(UKWAC) [5] as seen in Figure 3 to facilitate the navigation between versions. By using the cursor, users can browse different versions easily.

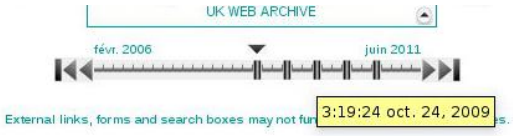

Figure 3: UKWAC navigation tool

The increasing number of national web archives, diversity of existing works led to the establishment of the International Internet Preservation Consortium (IIPC) [2] in Paris at 2003. The aim is to develop common standards, tools and techniques for web archiving. One of the current projects of IIPC, called WERA, is an archive access solution for searching and navigating web archives. It allows a full-text search besides wayback machine style search and it is based on the NWA(Nordic Web Archive) toolset [19] and the NutchWAX [26] full-text indexer. NutchWAX is extension of Nutch (an open source search engine based on Lucene java for searching and indexing) for searching web archive collections.

Today, most of web archive initiatives use the wayback machine to support URL indexing and search. NutchWAX is used to enable full-text indexing and search. Our motivation is focused on enabling complex queries which can not be performed by existing methods.

Query Languages for Web

A number of Web query languages have been developed in the past (e.g WebSQL, W3QL, WebLog, WebOQL, etc.) [17]. All those languages are intended for online queries on the web. From the perspective of web archiving, the most important inconvenience of those query languages is their lack of temporal dimension and the lack of handling challenges related to web archives like temporal coherence, incompleteness etc.

WebBase [23] project at Stanford University is a web repository project that aims to manage large collections of web pages and to enable web based search. A web warehouse is interpreted simultaneously as a document collection, as a directed graph and as a set of relations. For web based search, a query language with the notions of ranking and ordering is proposed. However, one copy of each page at a time is archived, thus, no temporal dimension is provided in that project.

A web warehousing system called WHOWEDA(Warehouse of Web Data) [10] proposed by Web Warehousing and Data Mining group at the Nanyang Technological University in Singapore aims to store and manipulate web information. It stores extracted web information as web tables and provides web operators to manipulate those tables. Its data model is based on nodes (pages) and links (hyperlinks) objects. Links do not have any time-related attribute. Any change in the last-modified time of the web document results in a new node. Content of a web document is represented as "node data tree". For HTML documents, it is a HTML DOM tree. The flexibility of HTML syntax might cause mistakes in DOM tree structure. In addition, however, DOM tree is powerful for presentation in the browser, it is not introduced for description of the semantic structure of the web page. In WHOWEDA, the internal semantic structure of a page is not modeled, thus it only allows queries to be specified on the whole content of the document. WHOWEDA only focuses on user interested web sites which constitute a much smaller scale than web archives.

Block-Based Search:

Using semantic blocks in web pages as an unit of information retrieval is an active research area. The vector space model customized with importance and permeability (the indexing of neighbors blocks) is proposed in [11] without temporal dimension. In [13], after segmenting each page into non overlapping blocks, an importance value is assigned to each block which is used to weight the links in the ranking computation. A block-based language model is proposed in [28]. As far as we know, there is no approach with temporal dimension in block-based search, ranking and language modeling. Our approach is based on visual page segmentation of web pages [22] and uses an importance model proposed in [25].

In conclusion, wayback machine, full-text search and navigation are the only applied solutions to access to web archives. Most of the web query languages do not contain temporal dimension for querying historical data. Different topics in a web page are not handled in search except recent works like [12, 15, 13] which suffer from lack of temporal dimension to be used for web archives. In our approach, we propose a data model which takes into account different regions of a web page with associated importance and temporal dimension.

4. WAC QUERY LANGUAGE

The details of WAC query language are described in this section. After briefly introducing the interpretation of temporal dimension in Section 4.1, we describe our data model

in Section 4.2 and list the operators in Section 4.3.

4.1 Temporal Dimension

4.1.1 Time on the Web

Integrating time into data models, query languages and database management system implementations still attracts the attention of researchers. We focus here on two questions: which time to use and how to represent it.

In the context of web archiving, there are different types of temporal information: time in content, HTTP headers, crawled time.

Temporal expressions can be found embedded in the content of a web page. Those expressions are explained in three categories in [8]: Explicit, Implicit and Relative. Temporal expressions that can be converted directly to chronons are explicit expressions (e.g "December 24,2010..."). Names of holidays or events which can be anchored in timeline are considered as implicit expressions like "Christmas Day 2010". Relative expressions can not be anchored in timeline directly like "today", "on Wednesday" etc. Extraction of temporal expressions from documents are active research field. Temporal information in content is excluded from our model.

The temporal fields in HTTP headers contain `Date` , `Last-Modified` and `Expires`. Servers send `Last-Modified` header with date time value when the content was changed last time. `Last-Modified` header is mostly used as a weak validator. According to the definition of HTTP specification, a validator that does not always change when the resource changes is a weak validator. The meaning of `Last-Modified` depends on the implementation of the origin server and the nature of the original source (files, database gateways, virtual objects, etc.) [1]. The header `Expires` indicates when a document stops being fresh. If it is used correctly, this header indicates a validity period for the document and an exact date for recrawling. But it is an unreliable tool: many servers do not provide the header or provide with zero or with low expiration delay.

HTTP header `Date` represents the date and time at which a web page is returned from a HTTP server upon request by the HTTP client. Crawled time is the time that crawlers capture the snapshot of a page. Web crawlers set the value of the `Date` field in crawled documents to the date that a document is crawled, unless they are configured otherwise. It is the most trustworthy time because it does not depend on host servers configurations. In our model, crawled time is used as the basic time dimension but we should underline the fact that our model supplies also other choices.

Our model uses Allen's interval-based representation of temporal data [7] where intervals are addressed as primitive time elements. Each element is temporally stamped by a period, called *validity*, defined as a time interval $[t_s, t_e)$ where t_s represents a starting time point and t_e is an ending time point. *now* represents the current time and is assumed to be larger than any timestamps.

4.1.2 Time in queries

There are two kinds of temporal queries in temporal information retrieval: time-point and time-interval. In time-point queries, a time point t is given in the query and the results should contain data whose validity contains t. In time-interval queries, an interval [t1,t2) is given in the query and data whose validity overlaps with [t1,t2) are returned as result. A time-interval query is also called a time slice query.

In our model, all queries are treated as time interval queries. For example, a time point query like "2010/08/19" is converted to time-interval [2010/08/19,2010/08/20).

4.2 Conceptual Model

In our model, each concept has its temporal and non-temporal definition. Non-temporal definitions do not contain temporal attributes and are denoted by $concept^-$. We follow the example in Figure 4 to illustrate the main features of the model. In this example, we have a web page crawled at t1,t2 and t3 without structural changes. The page is segmented in three blocks marked with green, blue and pink. In our approach, visually segmented blocks are called *concrete blocks*. Each concrete block has a frame block, content and importance. A page is a set of concrete blocks and a website is a set of pages.

Frame Block

Web page segmentation returns a set of non-overlapping hierarchical blocks. Frame Block (fb) keeps properties of a block: the url to which the block belongs, a Dewey identifier that indicates its place in the page structure and its validity interval. If a frame block disappears and reappears later, it is considered as a new frame block. A frame block is defined as follows:

$$fb = (URL, DeweyID, [fbt_s, fbt_e))$$

$$fb^- = (URL, DeweyID)$$

For the in Figure 4, frame blocks are:
$blue : (www.bbc.co.uk/news, 1, [t1, now))$
$pink : (www.bbc.co.uk/news, 2.1, [t1, now))$
$green : (www.bbc.co.uk/news, 2.2, [t1, now))$

Content

There are two kinds of contents: non-textual and textual. Non-textual content are images, videos etc. in a web page. In our model, treating non-textual content is out of our scope, thus, if the content has non-textual content, *binary* is introduced to provide support for further works. Textual content is treated as a bag of words (after elimination of stop words, stemming etc.) in a block. A *validity* attribute allows to trace the content changes over its frame block. Its validity should be included in the validity of its frame block. A block b can contain only one textual content but several non-textual contents. A content of a block b is defined as:

- if b has only textual content

$$c = (text, [ct_s, ct_e))$$
$$c^- = (text)$$

- if b has only non-textual content

$$c = \{(binary_1, [ct_s, ct_e)), ..., (binary_n, [ct_s, ct_e))\}$$
$$c^- = \{(binary_1), ..., (binary_n)\}$$

- if b has textual and non-textual content

$$c = \{(text, [ct_s, ct_e)), ..., (binary_n, [ct_s, ct_e))\}$$
$$c^- = \{(text), (binary_1), ..., (binary_n)\}$$

Figure 4: Example

In the example in Figure 4, contents are listed as followed:

$$\left.\begin{array}{l}((News),[t1,now))\\(binary,[t1,now))\end{array}\right\}blue$$

$$\left.\begin{array}{l}((Obama,America,gun,laws...),[t1,t2))\\((binary,[t1,t2)))\end{array}\right\}pink$$

$$\left.\begin{array}{l}((French,hostage,Niger...),[t1,t2))\\(binary,[t1,t2))\end{array}\right\}green$$

$$\left.\begin{array}{l}((Sarkozy,Obama,US,French),[t2,t3))\\(binary,[t2,t3))\end{array}\right\}pink$$

$$\left.\begin{array}{l}((Woody,Allen,ageing...),[t2,now))\\(binary,[t3,now))\end{array}\right\}green$$

$$\left.\begin{array}{l}((Sarkozy,Carla,Woody,Allen...),[t3,now))\\(binary,[t3,now))\end{array}\right\}pink$$

Importance

The blocks in a page have different importances as explained in Section 2. We define the importance as:

$$i=(alpha,[it_s,it_e))$$

$$i^-=(alpha)$$

The importance of a block, denoted *alpha*, depends on its location, area size, content, etc. It is calculated according to the model proposed in [25]. Validity of importance is not equal to the validity of the content in the same block. Although its content stays unchanged, the importance of a block can change by adding / deleting links or images, or by updating blocks size in the page.

For the example in Figure 4, Importances are listed as followed:

$(0.4,[t1,now))$ blue	$(0.6,[t2,now))$ pink
$(0.2,[t1,t2))$ pink	$(0.3,[t2,t3))$ green
$(0.1,[t1,t2))$ green	$(0.4,[t3,now))$ green

Concrete Block

Concrete Block is a region in a web page. It is defined as follows:

$$cb=(fb,\{c\},\{i\})$$

In this triplet, *fb* is a frame block, $\{c\}$ is a set of its content ordered by validity attribute and $\{i\}$ is a set of its importance ordered by validity attribute.

In Figure 4, Concrete Block for the green block is described as followed:

$$\left(green,\left\{\begin{array}{c}(French,hostage...,[t1,t2))\\(binary,[t1,t2))\\(Woody,ageing...,[t2,now))\\(binary,[t3,now))\end{array}\right\},\left\{\begin{array}{c}(0.1,[t1,t2))\\(0.3,[t2,t3))\\(0.4,[t3,now))\end{array}\right\}\right)$$

If we want to eliminate temporal nesting in data, for example to find different versions, T-FLAT operator can be used. Concrete block after applying T-FLAT operator is called *temporally flattened concrete block*, and denoted as *čb*. It has non-temporal triplet of frame block, its content and its importance and a validity which is equal to the intersection of validities of elements in triplet.

$$čb=\left((fb^-,c^-,i^-),[čbt_s,čbt_e)\right)$$

For an example above, T-FLAT operator returns temporally flattened concrete blocks as follows:

$$\left(green,\left\{\begin{array}{c}(French,hostage...)\\(binary)\end{array}\right\},0.1,[t1,t2)\right)$$

$$(green,(Woody,ageing...),0.3,[t2,t3))$$

$$\left(green,\left\{\begin{array}{c}(Woody,ageing...)\\(binary)\end{array}\right\},0.4,[t3,now)\right)$$

A snapshot of a concrete block at a given time t is a non temporal triplet valid at t.

$$cb^t=(fb^-,c^-,i^-)$$

Page

A page is a set of concrete blocks. It is built dynamically from concrete blocks for a given URL when needed. Validity of a page is the union of all concrete blocks' validity. A page is defined as:

$$p_{url}=\{cb1,cb2,cb3...\}$$

In our example (Figure 4), we have one page with three versions. This page is built as follows:

$$p_{url}=\left\{\begin{array}{l}(blue,(News,[t1,now)),(0.4,[t1,now))),\\\left(pink,\left\{\begin{array}{c}(((Obama,gun,laws...),[t1,t2))\\((binary,[t1,t2)))\\((Sarkozy,Obama...),[t2,t3))\\(binary,[t2,t3))\\((Sarkozy,Woody,...),[t3,now))\\(binary,[t3,now))\end{array}\right\},\left\{\begin{array}{c}(0.1,[t1,t2))\\(0.3,[t2,t3))\\(0.4,[t3,now))\end{array}\right\}\right)\\\left(green,\left\{\begin{array}{c}(French,hostage...,[t1,t2))\\(binary,[t1,t2))\\(Woody,ageing...,[t2,now))\\(binary,[t3,now))\end{array}\right\},\left\{\begin{array}{c}(0.1,[t1,t2))\\(0.3,[t2,t3))\\(0.4,[t3,now))\end{array}\right\}\right)\end{array}\right\}$$

A snapshot of a page is a set of concrete block snapshots for a given URL valid for a requested time and defined as follows:

$$p_{url}{}^t=\{cb1^t,cb2^t,cb3^t...,cbn^t\}$$

Site

A website is a set of web pages that are addressed relative to a common URL. In our approach, websites, as well as pages, are built dynamically by using regular expressions to find out which pages belong to which websites.

$$s_{regex} = \{p_{url1}, p_{url2}, p_{url3}...p_{urln}\}$$

A snapshot of a website at t is a set of pages valid at t.

$$s_{regex}{}^{t} = \{p^t_{url1}, p^t_{url2}, p^t_{url3}..., p^t_{urln}\}$$

Links

Link represents the hyperlink in the page. It points to a page from a frame block and defined as follows:

$$l = (label, type, from, to, [lt_s, lt_e))$$

where:

- *label* is a link label as shown here: $< ahref = "URL" > linklabel < /a >$.
- *type* is used to distinguish global, internal and local hyperlinks. A hypertext link in an HTML document is said to be:
  - interior if the destination document coincides with the source document (ex:href="#anchorname")
  - local if the destination and the source documents are different but in the same domain (ex:href="/news/art.html")
  - global if the destination and the source documents are located on different servers.
- *from* attribute corresponds to a frame block
- *to* attribute corresponds to an URL.

In our example (Figure 4), links are listed as followed:

("Mobile", local, blue, "/news/mobile", [t1, now))
("News", interior, blue , "#", [t1, now))
("Why America's gun laws won't change", local, pink, "/news/politics-25698422", [t1, t2))
("US-French push for Iran sanctions", local, pink, "/news/politics-2457913", [t2, t3))
("Bruni to star in Wood Allen film", local, pink, "/news/cinema-18698422", [t3, now))
("Two French hostages in Niger", local, green, "/news/world-4598422", [t1, t2))
("Woody Allen on ageing and death", local, green, "/news/cinema-2659874", [t2, now))

4.3 Operators

The proposed language consists of classical set operators with their temporal extensions, relational operators, navigation operators, interval-based temporal logic operators, aggregate operators, ranking and grouping operators. Due to the lack of space, only query operators related to web archive issues are detailed in this section. Table 1 lists the suite of operators.

InBlock

InBlock is one of our logical full-text operators like or-select(OR), and-select(AND), not-select(NOT) etc. It finds matches that satisfy full-text selection in the same block. It is used combined with other logical full-text operators.

For example, assume that we are looking for information about Woody Allen's new movie where Carla Bruni Sarkozy participates (Figure 4). A query like "Sarkozy and Allen" will return version at t2 and version at t3. In fact, information in version at t2 is not relevant. But a query like "Sarkozy InBlock-AND Allen" will return only version at t3 which has more relevant information.

Wayback

It returns all the versions of a page identified by its URL for a given period.

$$WAYBACK(URL, [ts, te))$$

Fixdate

This operator is used at the beginning of the queries to fix the date interval for all queries in a session. For example, if the user wants to work over the data of January 2000, after calling FIXDATE($[2000-01-01, 2000-02-01)$), she can call other operators without specifying the temporal attributes (*e.g* WAYBACK(url)).

Nearest/Recent/Both

These operators are used to deal with incompleteness explained in Section 2. For example, in Figure 4, if it is asked for the version at t where $t1 < t < t2$, we need to make an assumption over the version at t. Three different operators are proposed:

- NEAREST: it returns the nearest time by minimizing $|t - tx|$
- RECENT: it returns the closest time before t. It is the default operator, if the user does not specify another one.
- BOTH: it returns a time interval constructed with the most closest time before t and after t.

For Figure 4, a query WAYBACK(URL,t) will be executed:

- with NEAREST as WAYBACK(URL,t2), assume that $|t - t2| < |t - t1|$
- with RECENT as WAYBACK(URL,t1)
- with BOTH as WAYBACK(URL, t1) ∪ WAYBACK(URL, t2)

Navigation

Operator out_b finds the set of pages pointed in one step from the set of concrete blocks (CB) by following any of the links valid at a given period. Operator *out* uses out_b to find the set of pages reachable in one step from the set of pages (P) at a given period. For that, it finds the set of concrete blocks foreach page in P and calls out_b foreach CB.

Operator *in* finds the set of concrete blocks that points to a set of page by links valid for a requested period.

Operators $jump^+$ and $jump^-$ return a set of pages reachable, respectively, incoming and outgoing direction in one to n steps by following links valid at a given period. It is a combination of *in* and *out* operators with iteration.

Collapse/Expand COLLAPSE, also referred in literature as coalesce, combines the tuples, which have the same non-temporal values and consecutive or overlapping validities, into one tuple with validity that is the union of the constituent validities.

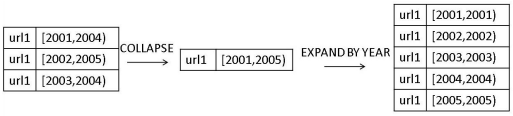

Figure 5: Collapse/Expand

EXPAND expands a tuple into several tuples by splitting its validity into consecutive validities in a given scale. In our approach this scale can be following keywords: YEAR, MONTH, DAY. These operators can be combined with *IN period* to limit the range. In Figure 5, EXPAND BY YEAR IN [2001,2004) returns the first three tuples.

5. USE CASES

In this section, to illustrate how the different operators work in our approach, we use two examples and construct the corresponding queries.

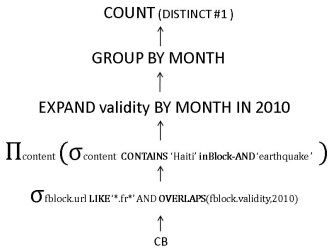

Figure 6: Query Example 1

Example 1: We extend the example that we gave in the introduction. A social researcher who studies how French media covered the event "earthquake at Haiti in 2010" over last year wants to know the number per month of *different regions* in web pages in domain .fr referring to earthquake by eliminating duplicates. In that case, if at t, there were two different articles in "lemonde.fr", it is counted as 2 instead of 1 (whole page). Figure 6 shows the query graph for this example.

First, we need to find all concrete blocks in domain .fr which are valid at 2010. LIKE is used to make a string comparison. Then, with CONTAINS operator, we find contents which mention given keywords (Haiti, earthquake). EXPAND operator is used to group by month over validity of content. COUNT and DISTINCT operators are used to find out the number of different regions. By using GROUP BY operator with url, we can limit the count to web pages.

Example 2: Our second example is based on finding broken links from a given url X in a given period (2000 in our example). Figure 7 illustrates the query graph for this example. We need to underline the fact that these broken links are not the result of incompleteness but HTTP 404 error while crawling.

6. CONCLUSION AND FUTURE WORKS

In this paper, we addressed the problem of accessing information in web archives. We presented a conceptual model as the basis of a query language for web archives. The operators to support queries are also described. In our model,

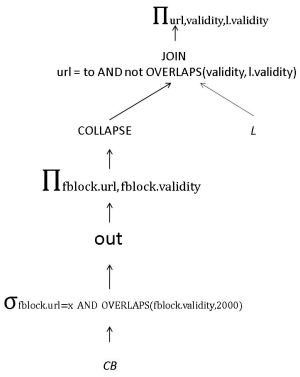

Figure 7: Query Example 2

we take into account different topics in web pages by using visual blocks as an unit of retrieval with accorded importance. Block-based approach is used for information retrieval on the web, however, as far as we know, it is never used with temporal dimension. Navigation operators with temporal dimension let users to execute queries over web archives temporal hyperlink structure. The model and operators enriched with the temporal dimension allow querying web archives powerfully.

Our approach is in the early stage of development. Our first priority is to express the language in algebraic form. Next steps will be the implementation with an appropriate user-friendly syntax. We want to underline the fact that in this paper we clarify the requirements of WAC query language formally. It can be implemented as a new query language or as an extension of an existing query language. We will also work on ranking functions which take into account the block-based structure and temporal dimension. By using the existing temporal indexing [9] and block-based indexing approaches [11], we intend to propose a hybrid indexing model. Once the proposed query language will be fully implemented, our attention will focus on query optimization strategies.

7. REFERENCES

[1] Html protocol, http://www.w3.org/protocols/rfc2616/rfc2616-sec14.html.
[2] Iipc, international internet preservation consortium, http://netpreserve.org/about/index.php.
[3] Internet archive, http://www.archive.org/index.php.
[4] Pandora, australia's web archive, http://pandora.nla.gov.au/.
[5] Uk web archive, http://pandora.nla.gov.au/.
[6] Use cases for access to internet archives. Use case report, IIPC, 2006.
[7] J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26:832–843, Nov. 1983.
[8] O. Alonso, M. Gertz, and R. Baeza-Yates. On the value of temporal information in information retrieval. *SIGIR Forum*, 41:35–41, Dec. 2007.
[9] K. Berberich, S. Bedathur, T. Neumann, and G. Weikum. A time machine for text search. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 519–526. ACM Press, 2007.
[10] S. S. Bhowmick, S. K. Madria, and W. K. Ng. *Web Data Management*. Springer, 1 edition, Nov. 2003.
[11] E. Bruno, N. Faessel, H. Glotin, J. L. Maitre, and M. Scholl. Indexing by permeability in block structured

OPERATOR	DESCRIPTION
Select	corresponds to well-known select operator in the relational algebra. It returns a subset of data which satisfies given predicates. Temporal select operation is the same as the non-temporal selection but it is extended with additional predicates for temporal comparison
Project	corresponds to its well-known non-temporal version in the relational algebra
Coherent	applies an estimation function over the archive and finds more coherent version for a given page
Rank	applies a ranking function over a bag
Cartesian Product	computes the Cartesian product of two bags
Temporal Cartesian Product	returns a temporal Cartesian product of two bags
Union	returns the union of two bags
Temporal Union	returns the union of temporal elements where their validities overlap
Intersection	returns the intersection of two bags
Temporal Intersection	returns the intersection of temporal elements where their validity overlap
Difference	returns all the elements of the first bag which are not in the second bag
Temporal Difference	in addition to non-temporal Difference, it checks and removes overlapping temporal parts from bags
Join	performs an inner-join on two non-temporal bags based on predicates
Temporal Join	performs an inner-join on two bags based on predicates
Distinct	removes duplicates for non temporal bags
Temporal Distinct	in addition to non-temporal Distinct, it removes duplicates by taking into account overlapping temporal parts
GroupBy	creates groups of tuples sharing some attribute value
Aggregate	COUNT, MIN, MAX are supported
Forward / Backward	for a given element returns the version after/before
T-FLAT	eliminates the temporal nesting in a bag
Logical Full-Text Operators	OR, AND, MILD-NOT, NOT, ORDERED, IN-BLOCK are supported
Contains	realizes keyword search over index servers
OrderBy	sorts a set into a specific order
Temporal Operators	Allen's thirteen temporal operators [7] are supported
Diff	finds differences between two versions of a web page

Table 1: List of Operators

web pages. In *Proceedings of the 9th ACM symposium on Document engineering*, DocEng '09, pages 70–73, New York, NY, USA, 2009. ACM.

[12] E. Bruno, N. Faessel, J. L. Maitre, and M. Scholl. Blockweb: An ir model for block structured web pages. In *Content-Based Multimedia Indexing, 2009. CBMI '09. Seventh International Workshop on*, pages 219–224, 2009.

[13] D. Cai, X. He, J. Wen, and W. Ma. Block-level link analysis. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 440–447, New York, NY, USA, 2004. ACM.

[14] D. Cai, S. Yu, J. Wen, and W. Ma. VIPS: a vision-based page segmentation algorithm. Technical Report MSR-TR-2003-79, Microsoft Research, 2003.

[15] D. Cai, S. Yu, J. Wen, and W. Ma. Block-based web search. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 456–463. ACM Press, 2004.

[16] Y. Cao, Z. Niu, L. Dai, and Y. Zhao. Extraction of informative blocks from web pages. In *Proceedings of the 2008 International Conference on Advanced Language Processing and Web Information Technology*, pages 544–549, Washington, DC, USA, 2008. IEEE Computer Society.

[17] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the World-Wide web: A survey. *SIGMOD RECORD*, 27:59–74, 1998.

[18] D. Gomes, A. L. Santos, and M. J. Silva. Managing duplicates in a web archive. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 818–825, New York, NY, USA, 2006. ACM.

[19] T. Hallgrimsson and S. Bang. Nordic web archive, 2004.

[20] M. Kovacevic, M. Diligenti, M. Gori, M. Maggini, and V. Milutinovic. Recognition of common areas in a web page using visual information: a possible application in a page classification. In *in the proceedings of 2002 IEEE International Conference on Data Mining (ICDM'02*, page 250. IEEE Computer Society, 2002.

[21] J. Masanés. *Web Archiving*. Springer Berlin Heidelberg, 2006.

[22] Z. Pehlivan, M. Ben-Saad, and S. Gançarski. Vi-DIFF: understanding web pages changes. In *Database and Expert Systems Applications*, volume 6261 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin / Heidelberg, 2010.

[23] S. Raghavan and H. Garcia-Molina. Complex queries over web repositories. In *Proceedings of the 29th international conference on Very large data bases - Volume 29*, VLDB '2003, pages 33–44. VLDB Endowment, 2003.

[24] M. Ras and S. van Bussel. WEB ARCHIVING user survey. Technical report, National Library of the Netherlands (Koninklijke Bibliotheek), Netherlands, 2007.

[25] R. Song. Learning block importance models for web pages. In *In Intl. World Wide Web Conf. WWW*, pages 203–211. ACM Press, 2004.

[26] M. Stack. Full text search of web archive collections. In *In IWAW*, 2006.

[27] B. Tofel. Wayback for accessing web archives. In *IWAW'07*, Vancouver, British Colombia, Canada, 2007.

[28] Y. Zhang, K. Tanaka, J. X. Yu, S. Wang, M. Li, S. Li, S. Huang, G. Xue, and Y. Yu. Block-Based language modeling approach towards web search. In *Web Technologies Research and Development - APWeb 2005*, volume 3399 of *Lecture Notes in Computer Science*, pages 170–182. Springer Berlin / Heidelberg, 2005.

Author Index