Content curation and characterization in communities of a place

Giuseppe Scavo

PHD THESIS
UNIVERSITY PIERRE ET MARIE CURIE

Spécialité : Ingénierie

École doctorale : "EDITE"

réalisée

à l'Inria et Nokia Bell Labs

présentée par

Giuseppe SCAVO

pour obtenir le grade de :

DOCTEUR DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Content Curation and Characterization in Communities of a Place

soutenue le 15/12/2016

devant le jury composé de :

M.	**Marcelo DIAS DE AMORIM**
M.	**Krishna GUMMADI**
M.	**Dan VODISLAV**
M^{me}	**Renata TEIXEIRA**
M.	**Zied BEN-HOUIDI**

Contents

Chapter 1

Introduction

The amount of information in the Internet overwhelms most users. Discovering relevant information (e.g. news to read or videos to watch) is time consuming and tedious. As Laudon et al. [1] report, discovering the right information is part of the daily job of at least 80% of the employees in North America. This percentage corresponds to the set of knowledge workers, whose daily job is to digest and transform information. Search engines have solved part of this information overload. Searching for something specific however, is only a fraction of users' interactions with the web. For instance, knowledge workers spend around 60% of their web time either browsing with no specific goal in mind or gathering information from known websites [2].

Several information filtering systems for the web can ease this task for users. For example, users who want to find relevant news can visit news aggregators like Google News; users who want to find interesting web pages can visit Reddit and receive recommendation from other users. The general idea behind such information filtering systems is that on one side there is an enormous amount of information and on the other the user. The information filtering system seats between the two and suggests to the user only the interesting content, filtering out the noisy information. We use the general word "information" deliberately, as one can use information filtering systems in multiple contexts. For example to pick the most interesting web pages (e.g. Delicious) or to chose which movie to watch next (e.g. Netflix). The first appearance of the keyword "information filtering" in the context of web dates to the beginning of the 2000s. For example Widyantoro et al. [3] use this word to describe their system that suggests web news, and Jung et al. [4] use it for their system implementing document search. Many different systems exist with the goal of filtering different kinds of information and targeting different audiences. In a more general context (i.e. not web specific), information filtering systems refer to any system that removes noisy informations (e.g. spam filters).

Information filtering systems fall in two families. The first gathers the tools that suggest information in a "pull mode". In this case, the users look for information intended to satisfy a particular need. Examples are search engines that suggest information matching users' queries. Some researchers refer to this family as "information retrieval systems" [5]. The second family gathers the tools that propose information in "push mode". In this case, users do not try to satisfy particular needs. The system finds interesting information and suggests it to the users. As the users are not involved in the discovery of new information, we refer to these tools as *passive information filtering systems*. All passive information filtering systems for web

content require some kind of user activity. Most of the time, users contribute to the system adding new content or scoring pre-existing items (see Chapter 2 for a detailed classification of such systems). For example, recommender systems need users to rate items. As another example, social networks and social rating websites have been shown to be "naturally" efficient information filtering systems [6, 7, 8]. The sum of the efforts of their users who create, find, submit, relay, and rate content turns these into efficient *information filtering systems*. User engagement in such a setting is the key of the success of such tools. Unfortunately, it is hard to obtain. The lack of user engagement translates into cold start and data sparsity. Internet-wide systems like Reddit and Digg suffer from this problem [9, 10].

The lack of user engagement is even worse in the case of *communities of a place*. Communities of a place group people who live, study, or work in the same place. Examples of communities of a place are: (i) the students of a campus, (ii) the people living in a neighbourhood or (iii) researchers working in the same site. Members of communities of a place share interests. For example, the students of a campus need to share information about university. People living in a neighbourhood need to share local informations like local shops or important events. Unfortunately, they either do not know each other or fail to actively engage in sharing content. Luckily, the members of communities are often connected to the same network. This creates the opportunity to study their online behaviour from a single point of view. For example, from the network log traces or from the network proxy.

In this thesis, we leverage the passive observation of HTTP logs of entire communities of users. We obtain a new rich source of input that we use to (i) design WeBrowse, a content curation tool for communities of a place, and to (ii) characterize news consumption in communities of a place.

1.1 Existing Information Filtering Systems

In the design space of information filtering systems that push content to users, we identify two families. The first comes more from "technology" and group the content curations tools, the second is rooted in research, with a lot of applications in real-world systems and group the recommender systems. Some refer to the first family as content curation tools [11, 12]. Content Curation is the act of using manual or automated resources to assist users in discovering relevant information in the web. The term has recently gained popularity in the web landscape, witnessed by an explosion of commercial tools (see [13] for a list of the most important ones). More generally, we classify such tools into: (i) those using users as curators and (ii) those using automated resources. In the first category, we distinguish between: social rating (e.g. reddit, hackernews) based mainly on the user actions *submit* and *rate*, social bookmarking (e.g. Pinterest, Delicious) based mainly on *submit*, *categorize* and *follow*, and finally even social networks which are based on the actions *submit*, *follow* and *share*. Such tools act as information filtering mechanisms, although it was not necessarily their primary goal [6, 7, 8]. In the second category, we find mainly news and blog aggregators (e.g. Google News) which actively crawl the web looking for new pages to suggest, feed aggregators which aggregate feeds from different sources including social networks (e.g. Flipboard), and finally trending systems (e.g. Google Trends [14]). Most of the research work on such systems has been focused on modelling and understanding their (e.g. filtering) dynamics [15, 6, 7, 8, 16, 17, 18, 19].

Recommender systems have gained attention from research [20, 21] with a dominance

of collaborative-filtering [22, 23] methods. Collaborative-filtering methods take as input a user-item matrix that encodes users' preferences about a collection of items, and aim to predict user preference for a new item. One important metric to assess such solutions was often accuracy: to what extent the prediction corresponds to the real preference.

Both approaches are complementary, as recommender system algorithms can apply on top of the user input collected by content curation tools [24]. Some social networks like Facebook and Linkedin already do it. But they have also similarities: recommender systems find their historical origins in manual collaborative filtering [21], something at the heart of the user-based content curation tools enumerated above. For instance, while user-based collaborative filtering techniques infer similarity between users based on their past preference, users of some content curation tools provide such information explicitly by following each other.

We think that several factors explain the place that content curation has in practice despite the advent of more and more accurate recommendation algorithms. First, due to the lack of user input, recommender systems were traditionally applied only to a single service at a time (e.g. only movies). The more diverse user input obtained in content curation tools favours serendipity (e.g. providing unexpected suggestions) and diversity. It is worth noting that after a long algorithmic quest for accuracy, encouraged by contests like the Netflix prize [25], the recommender community started to target new metrics [26] like serendipity and diversity. The second factor is trust and the fact of taking into account explicitly user similarities. In the next section we talk about one of the biggest challenges in information filtering systems: the lack of user input.

1.2 Lack of User Input in Communities of a Place

The vast number of existing Information Filtering Systems is the answer to the huge amount of Internet content that users have to go through daily. However, the tools presented in the previous section are tributary to user input. Information filtering systems require users actions to work. For example, inputting news content in the system, scoring items, commenting others' content, or saving favourite items. The scarcity of user input translates into cold start and data sparsity problems [27]. Reddit for instance suffered from cold start in its early days. Today, it still suffers from the lack of enough users outside the US, and from the inactivity of most users. The lack of users input can cause these communities to become closed and self-referencing [15].

The lack of user input is due to under engagement of the users in the systems. Users who do not actively engage in a system are called "freeloaders". When the percentage of freeloaders is high, the systems do not work well. Take again the example of Reddit. Gilbert [16] shows that more than half of the interesting content of the system is lost due to the lack of users' active participation. To counter this problem, research has focused (mainly in the recommender systems space) on augmenting user input, either by means of passive techniques (for example exploiting implicit feedback) [28] or by using social information data [29]. In the next section we explore more the idea of augmenting the user input by means of passive techniques. A large body of research on community or social-based recommendations [30, 31, 32, 33, 29] has shown that in general social-based recommendation leads to better results than recommendation based on users' preference similarity. However, as pointed out by a comprehensive survey [34] on social-based recommendation systems, such approaches remained theoretical, (i.e. there not exist any deployed system that integrates social information

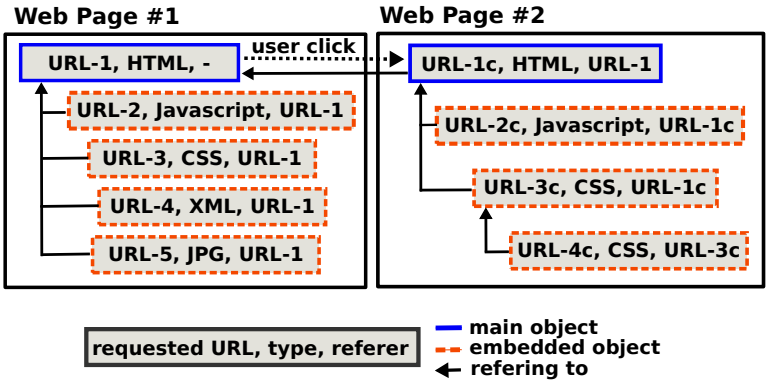


Figure 1.1: Sequence of HTTP requests when accessing some standard webpages.

in the computation of recommendation), and were not tested on online systems. Indeed, augmenting recommender systems with social-based information would require either mining such information, which is not always available or applying community detection mechanisms to infer groups of preference.

The lack of user input becomes even more pronounced in the case of information filtering systems for communities of a place. In this case, the population is much smaller than other country-wide systems like Reddit or even world wide like Facebook. Data we obtain from a large corporation's social network shows that, on average, only 0.3% of workers contribute daily, and 10% monthly. Nevertheless, information filtering systems for communities of a place are more and more considered useful tools. Several enterprises and universities encourage [35, 36, 37] the deployment of internal social networks [38, 39], blogs [40], wikis [41] and social bookmarking [42] to foster knowledge discovery. Friedman et al. [43] report the struggle to obtain user engagement to start such information filtering systems within an enterprise.

In this work, we aim to solve the problem of the lack of user input for information filtering systems in communities of a place.

1.3 Implicit Input from HTTP Logs

The typical approach, for an information filtering system, to obtain user input when users are inactive is to use implicit feedback. Implicit feedback is the fact of using user interaction with a given presented item to infer his preference about this item. Different kinds of actions can help inferring preferences (e.g. using query reformulations, clicks or time spent in front of a product, and scrolling actions). Users do not need to know that they are contributing to improve the quality of the system. An example of a system using implicit feedback is Amazon. Amazon uses the information that users bought a book as implicit feedback.

In this thesis, we exploit users' clicks on webpages as user input. We virtually emulate a mixture between a social rating system (submit and rate) and a social network (connecting otherwise disconnected users of the same community). Intuitively, (1) the first click on a URL is a submission of the corresponding page to the system and (2) the more users' click on this URL, the higher the rating and thus the interest in the corresponding page. Assuming people in the community share common interests, and that the most popular pages are discovered only by a small part of the community, the resulting promoted content should satisfy their needs in

discovering relevant, often new, web content.

This approach sounds similar to implicit feedback. A key difference is that, in our case, items are not known in advance. Indeed, a click in our system is not only a feedback about an existing item (e.g. an Amazon product), it is also a submission of a new *discovered* item to the system. Another difference is that, by collecting the clicks of a community of users, we implement in practice the theoretical [34] findings that showed that social-based recommendations outperform traditional collaborative filtering methods [30, 31, 32, 33]. In practice, this approach implement *passive crowdsourcing*: using the efforts of users (for example submitting and rating) without their intentional contribution. The idea of feeding an information filtering system with passively observed data sounds appealing. It comes, however, with a number of challenges and open questions that we propose to answer in this thesis.

First, to extract users' clicks is challenging because webpages have become considerably more complex in the last decade [44]. When a user visits a page, the browser first issues an HTTP request to the main URL. In this thesis, we call this webpage, which the user explicitly visits, *user-URL*. Webpages typically embed diverse types of objects such as HTML, JavaScript, multimedia objects, CSS, or XML files. Ajax, JavaScript, and Flash objects may dynamically fetch other objects. Each of these objects (both static and dynamic) is referenced with another URL and may be served from different (maybe third-party) servers. Thus, the browser issues individual HTTP requests for each object. Although fetching all these objects is necessary to render the page, the user does not directly request these objects. We call all these URLs *browser-URLs*. Take the example of the user visiting 'Web page #1' in Fig. 1.1. The user visits URL-1, which is a user-URL. The browser then requests the URLs of the four embedded objects. This example also shows the type of each requested object (e.g., the main object is an HTML file). Take again the example in Fig. 1.1. After visiting 'Web page #1', the user clicks on one of the links in the page (in this example, URL-1c) to visit 'Web page #2'. The figure also presents the referer field of each HTTP request. The *referer* field contains the URL of the page that originated the request. In this example, the referer of the HTTP request to URL-1c is URL-1.

Second, the goal of an information filtering system is to promote content to users. Not all user-URLs are interesting for all the users. For example the website of a bank or a website of a search engine are not worth being recommended. We call URLs worth being promoted *candidate-URLs*. A challenge is how to detect them.

Third the passive crowdsourced system must not violate user privacy, e.g. by promoting private information or allowing to identify who visited a given web page. Fourth, although appealing on the paper, it is not clear if users would appreciate the content such a system promotes. Fifth, the relatively small number of users composing a community of a place might be an issue, as it might limit the crowdsourcing effect. For example, Reddit has millions of users contributing with 200,000 submissions per day [45], while communities of a place typically consist of few thousand users. Hence, the question for us is whether passive crowdsourcing is feasible for such small communities of a place, and starting from which number of active users. Finally, one of our assumptions is that people in the same community of a place share common interests. One question for us is thus to understand if they really do.

1.4 Contributions

In this thesis, we design methods to passively extract users' interests in web content. We leverage these methods for content curation and to characterize news consumption in communities of a place. We summarize our contributions as follow.

1.4.1 Identifying Users' Clicks

The first contribution of this thesis is a set of algorithms that allow us to infer users' clicks starting from HTTP log traces. This task is challenging because the logs contain much more information than the visited URLs. We design several heuristics in a modular way, such that we can mount one on top of the other and thus test different combinations. This allows to find the appropriate combination of filters that allows us to maximize precision and recall.

As the heuristics to detect user-URLs are intended to be used live in the network, we design a scalable algorithm that works online. In Sec. 3.4, we test it and compare it to the state of the art. We find that our algorithm can reach a precision of 90.53% and recall of 82%. However, as we want to be sure to detect every relevant content, for us recall is more important than precision. We are able to push our recall to 96.51% losing 27% of precision. Finally, our comparison with the state of the art in clicks extraction tools, shows that for similar values of precision and recall our algorithm is faster.

1.4.2 A Passive Crowdsourced Information Filtering System

WeBrowse is an information filtering system for communities of a place. Differently from the other information filtering systems, WeBrowse does not need user engagement. Fig. 1.2 presents an overview of WEBROWSE. WEBROWSE takes as input HTTP requests from a raw data extraction layer and outputs a list of sorted URLs to a presentation layer, which can be a website or an API service for mobile apps. In this thesis, we focus on the WeBrowse layer. The WeBrowse layer has four sub-blocks as depicted in Fig. 1.2. The *user-URL filter* (Sec. 3.1) identifies the HTTP requests corresponding to actual users' clicks. It eliminates the vast majority of HTTP requests that the browser automatically generates. The *candidate-URL* (Sec. 4.2.1) and the *content versus portal* modules (Sec. 4.2.2) together select the set of user-URLs that are worth sharing with other users. Finally, the *promotion* module (Sec. 4.2.3) takes as input this set of URLs (together with their timestamp) and decides which ones to output to the presentation module.

Our contributions are:

- We find that passive crowdsourced content curation is technically feasible, at scale, while respecting user privacy. Sec. 4.2.2 describes scalable online algorithms to detect on the fly recommendable URLs from raw HTTP logs, Sec. 4.2.3 designs privacy-preserving promotion mechanisms for WEBROWSE and Sec. 4.3 evaluates the scalability of the system.

- We rely on explicit feedback from 115 users to evaluate the promoted content. Almost 70% of them rated the content WEBROWSE promotes from very to extremely relevant (Sec. 4.4.2).

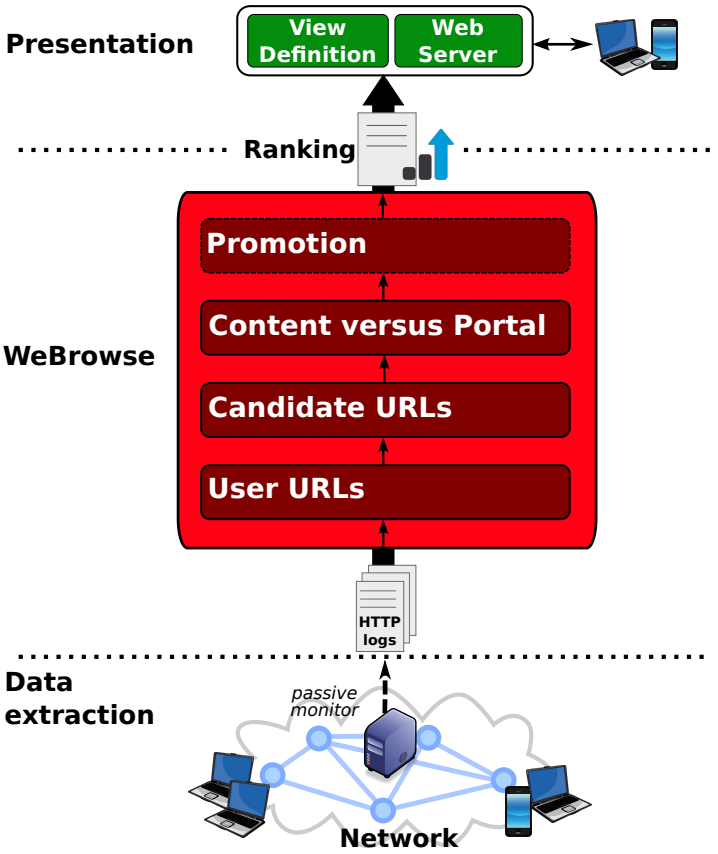


Figure 1.2: An overview of WeBrowse, our proposal for an information filtering system that works passively. WeBrowse is composed of three layers. The data extraction layer extracts HTTP logs from the network. The WeBrowse layer extracts the clicks and select the pages to promote. The presentation layer is the interface where users can find the promoted pages.

- Studying the effect of the crowd size, we show that passively observing approximately one thousand active users is sufficient to identify enough interesting content to promote (Sec. 4.4.1).

- Finally, analysing popular topics in different communities of a place, we find that people in the same community of a place share common interests. People in the same neighbourhood share common interests, but less common interests compared to people in the campus, and more compared to people in another city (Sec. 4.4.3).

1.4.3 News Consumption Characterization

The interest in the exchange and consumption of news is one of the oldest human habits [46]. What has changed over time is the medium to disseminate news: from word-of-mouth, to written, to printed, and finally to online publishing. The shift from print to online offers new opportunities to study people's consumption of online news. The results of such studies are enlightening for social scientists, but also for news organizations as it helps them to better allocate their journalistic resources as well as adapt their content to the needs of their readers.

The literature is rife with studies of online news consumption (which we review in Chapter. 2). These studies have characterized different aspects of online news consumption, for example, the virality of news [47], news coverage in social media compared to traditional media [48], and user preferences compared to editors' suggestions [49]. In general, prior work studied users' activity when browsing a single journal's website or share in one given social networking site, but individual users may visit different journal sites and share news using different means from email to social networks. We lack a holistic view of online news consumption of individual users.

In this work, we bring novel datasets and methods to shed light on online news consumption of a population of users. Similar to prior studies [50], we analyze online *news visits* (or when users' click on web links to news articles) as an indication of news consumption. In contrast to prior work, we aggregate all online news visits of individuals in a population of users across any online journal sites they visit when connected to a given network. In particular, we observe online news visits in a network. Thus, we can observe all news visits of users connected to the Internet through the link we monitor. We perform our analysis on data collected for 16 months at four different network links in Italy: three links in the network of a large residential Internet Service Provider (ISP), two of them connect around 11,00 households the other around 1,500 households; and one in a campus network with $20,000$ users. During this period we observe 80 million online news visits across the four datasets. We note that we can only monitor users for part of their day (in the ISP datasets, when users are at home, and in the campus dataset, when they are in the campus). Despite missing news visits when users are away from the places we monitor, when they are connected via one of the links we monitor we can view *all* their news visits. The main findings of our study are:

- We confirm previous results that show that web journals are still the main source of news visits. Only 16% of news visits in our datasets comes from social networks. This result implies that studies that focus only on news shared on social media are capturing a small fraction of news visits.

- People in the same city have closer tastes compared to people in different cities. We estimate that approximately 9% of news visits are local news.
- Our comparison of headlines chosen by editors and news visits confirm prior work [49] that showed a mismatch between what editors think interesting and what users really visit.
- Although most news aggregators recommend mainly fresh news, we find that 44% of the visited news are one to several days old.
- Our clustering of online newspapers, based on co-visits, suggests that people who read newspapers with strong opinions (e.g. clearly right wing) tend to read also the newspapers from the opposite opinion, suggesting the absence of selective exposure (i.e. the act of selecting specific aspects of information) for such a category of users.

Chapter 2

Related Work

In this chapter, we first describe the related work about information filtering systems. As there exists a large number of tools and systems with no consistent terminology, we elaborate an unifying functional taxonomy of existing systems. Second, we discuss prior work in the field of mining of weblogs. In particular, we show other systems that use similar approaches to ours. Third, we present tools that promote content in communities. Finally, we turn our attention to previous work in characterizing online news consumption.

2.1 Information Filtering Systems for the Web

Information Filtering Systems for Web content are tools that seat between users and the huge amount of information present today in the Internet. Their role is to filter out noise and provide users with only relevant information. The last years witnessed the proliferation of such tools. In this section, we first classify the existing information filtering systems, introduce a taxonomy of these systems. We discuss some open problems in this field such as cold start of systems and sparsity of data. Then, we discuss previous work that have characterized individual systems. We continue presenting the tools specific for communities of a place. Finally, we review work dealing with the problem of evaluating the quality of the filtered items.

2.1.1 Categories of Information Filtering Systems

In this section, we list the most important families of information filtering systems for the web. For each of them, we identify some of their features and provide examples. We will later describe the families.

- As defined by Boyd at al. [51], a **Social Network** website is a service that allows users to: (i) create a personal profile, (ii) connect their personal profile with other users' profiles, and (ii) view and traverse their list of connections within the system. Examples of Social Network websites include Facebook, Twitter, and Google+. Users have a personalized view of the content according to their network of contacts. This means that the system filters out the content not relevant for a group of users.
- A **Social Rating** system allows users to submit content and let others rate it. The system promotes the content with highest rating and discards the rest. Examples are Reddit, Digg, and Hackernews.

- **Social Bookmarking** Systems or Collaborative Tagging Systems allow users to tag web documents using a list of keywords [52, 53, 54]. The users can share the tagged documents with other users. Examples include Pinterest and Delicious.

- **News Aggregators** actively scrape news from a predetermined set of source websites. Examples are Google News and Yahoo News. News Aggregators can be **personalized** or not.

- **Trending Systems** measure the popularity of content (e.g. web pages, videos) and show ranked lists to users. Examples are Google Trends and the section "trending" on YouTube.

- **User-based Recommender Systems** track users preferences to recommend personalized content. User-based recommender systems use the idea that two similar users have similar tastes. Thus the knowledge acquired about one user can be used to recommend content to users with similar taste. Last.fm, a website promoting music, is an example of such tools.

- **Item-based Recommender Systems** first calculate similarity between items. Then, they learn the preferences of the users and propose items similar to those they rated with high score. Amazon is an example of Item-based recommender systems. Linden et al. [55], working at Amazon, proposed the first type Item-based recommender system.

- **Content-based Recommender Systems** need the description of the item and the history of the users. The systems use the history of the users to select the items to propose.

  They combine these information to propose items that fit users' profile.

Table 2.1 shows examples for each family. Note that some tools belong to more than one of the listed families. For example, Pinterest is a social bookmarking tool because it allows users to tag web content and it is also a social network because it allows users to connect with each other. Also Twitter belongs to two families. It is a social network because it allows the creation of connections between users and it is also a trending system because it shows the trending topics of the moment.

2.1.2 Functional Taxonomy

We describe different features that allow to classify the families of systems we introduced in the previous section according to their functionalities. We are aware that there could be other ways to classify existing systems. (We present other classifications in Sec. 2.1.4). We chose this particular classification to show the features that are specific to WeBrowse and those that are similar to other systems.

Vantage Points and Breadth

The first feature that allows to differentiate the tools is the **Vantage Point**, which is, where does the input of the system come from. Depending on the system, the vantage point can be:

- **Server Logs**. The systems use historical information of the users' activity interacting with their website (e.g. YouTube using number of visits).

Table 2.1: Information filtering mechanism and examples

Information filtering mechanism	Example of services
Social Network	Facebook Timeline Twitter
Social Rating	Reddit Digg Hackernews
Social Bookmarking	Pinterest
News Aggregators	Google News Yahoo News MSN
Personalized News Aggregators	Google News
Trending systems	Google Trend Twitter Trend YouTube Popular
Item Based Recommenders	Amazon
User Based Recommenders	Last.fm
Content Based Recommender	Netflix
WeBrowse	WeBrowse

- **Scraping**. The systems have scraping softwares that actively look for new information.
- **Web App**. The systems discover new information thanks to explicit user submissions.
- **Browser Extension**. The systems augment the browser functionalities allowing the users to input new information or rate items (e.g. the "pin it" button of Pinterest).
- **Network or Proxy**. The systems access the HTTP traffic from the Proxy logs or from the network directly. To the best of our knowledge, WeBrowse is the only system using input from the network.

The second feature we consider is the **Breadth** of the system's content. It tells about the type of information that the systems deal with. This information can be:

- **Service Specific.** In this case the promoted content belongs to the same tool. An example is YouTube's most popular recommendations. It recommends only videos coming from YouTube itself. Another example is Amazon suggesting other products from its platform.
- **Web Wise.** In this case, the promoted content comes from various sources. For examples, Google News groups articles from different editors. A Facebook feed has links from various sources.

Table 2.2 maps the families of information filtering systems to the corresponding vantage point type and breadth. As they need explicit user input, Social Networks, Social Rating, Item Based Recommenders, User Based Recommenders, and Content Based Recommender use internal vantage points. All these systems have a notion of users and need to learn their

Table 2.2: Vantage Point and Breadth of Information Filtering Systems

	Vantage Point					Breadth	
Service	Server Log	Scraping	Web App	Browser Extension	Network or Proxy	Web Wise	Service Specific
Social Network			✓			✓	
Social Rating			✓			✓	
Social Bookmarking				✓		✓	
News Aggregators		✓					✓
Personalized News Aggregators		✓					✓
Trending systems	✓						✓
Item Based Recommenders			✓				✓
User Based Recommenders			✓				✓
Content Based Recommender			✓				✓
Webrowse					✓	✓	

preferences to work. Social Bookmarking systems have browser extensions that allow users to save and tag web pages they like. News Aggregators and Personalized News Aggregators actively scrape a predetermined list of website. They download from these pages the content to promote to users. For example, Google News Italy, that we used for our experiments, has a set of about 500 portals from where it collects the main articles. Trending systems use previously logged data. For example, Google Trends logs the query that users make to show afterwards them ordered by popularity. Finally WeBrowse's input is any source of clicks, thus it can be Networks or Proxies. The systems that do not allow users to input content have a Service Specific breadth. These are: News Aggregators, Personalized News Aggregators, Trending systems, Item Based Recommenders, User Based Recommenders, and Content Based Recommender. All the social systems allow users to input new content thus their breadth is Web Wise. WeBrowse's input is any webpage that users in the monitored network visit. Thus its breadth is Web Wise.

User Input

The second feature is the **User Input**. This is the user action that the systems use to perform the information filtering task. We have the following possibilities:

- **None**. Users don't need to perform any action on the system. They passively consume the content that the system suggests (e.g. read the news that publisher put in evidence.)
- **Follow**. Users follow other users with similar tastes.
- **No Need to Follow**. Some systems (e.g. Google News and Twitter Trend) do not require the users to follow other users.
- **Rate**. Users rate the promoted content, giving a positive or negative score.
- **Submit**. Users input new content in the system, thus suggesting it to others
- **Categorization**. Users categorize content by assigning labels.
- **Implicit Rating**. The rating happens implicitly. For example, clicking on a query result on Google or watching a video on YouTube.

- **Implicit Submission**. Users input new content in the system implicitly. For example, WeBrowse's inputs are the pages visited by the network users.
- **Implicit Categorization**. The system takes as input the labels that users edit for their personal use (i.e. they do not aim to contribute in improving the system).

Table 2.3: User input of Information Filtering Systems

	User Input								
Service	None	Follow	Rate	Submit	Categorizization	Implicit Rating	Implicit Submission	Implicit Categozization	No need to follow
Social Network		✓	✓	✓					
Social Rating			✓	✓	✓				
Social Bookmarking								✓	
News Aggregators	✓								
Personalized News Aggregators		✓							
Trending systems						✓			✓
Item Based Recommenders			✓			✓			
User Based Recommenders						✓			
Content Based Recommenders			✓			✓			
Webrowse						✓	✓		✓

In Tab. 2.3, we show how existing information systems collect users' input. Social networks become efficient information filtering mechanism thanks to the collective efforts of users who perform the actions of following, rating, and submitting. Following a person is considered as an explicit shared interest for the topics he's interested in. Users can submit new content in the social network, for example link to webpages or videos and other users can express their approval by rating this content. For example, in Facebook users can push the Like button to increase the rating of one item. The same thing happens in Twitter when users retweet. Differently from social networks, social rating systems do not require users to follow each other. They submit and rate content. In addition, they categorize the items of the system. For example, Reddit has categories, called Sub-Reddit, where all the items are stored to ease the search of other users. Example of categories one finds in Reddit are: movie, news, funny, jokes. The only input that social bookmarking systems use is the implicit categorization. These systems allow users to store their favourite web pages in personal folders. To ease the retrieval of the pages, they apply categories on them. Nevertheless, the users do not categorize webpages for the sake of the community, but rather for their personal benefit. For this reason, we call this operation implicit categorization. News aggregators do not need user input. As their goal is to create a front page with headlines coming from different websites, their website is the same for every user. In the case of Personalized News aggregators, the users need to express their preferences. This happens by following a specific topic. For example, Google News can be considered as a hybrid system. The front page shows the same headlines to every user. However, nowadays Google News hosts a personalized section. In this section, users find content matching their explicitly expressed preferences. Item Based Recommenders and Content Based Recommenders obtain user feedback by means of rating. The rating can be explicit or implicit. For example, Amazon users can rate each item using a five start method. The rating can be also implicit, for example, Netflix suggests movies that have similar topics to previously seen videos. In User based recommender systems, the input is obtained by implicit rating. Indeed, users of the systems look for users with similar tastes (e.g. creating links with people that like a given topic or live in a given area) but they do not need to know each other. Webrowse uses implicit rating and implicit submission. We assume that a click on a webpage is

a valid expression of interest (Liu et al. [56] make a similar assumption to personalize news articles). The submission happens the first time a content is visited in the monitored network. From this moment, WeBrowse starts to count the page popularity. Finally, WeBrowse does not require users to follow each other.

Level of Personalization and Suggestion Method

The Level of Personalization tells about the audience of the suggestions output by the systems. There are the following options:

- **None.** The system promotes content that is supposed to be interesting for a general public.
- **User.** The system promotes a different feed to each user to fit his preferences.
- **Community.** In this case the system promotes content to group of users (e.g. a community sharing a particular interest or a geographical area.)

In this section, we consider also another metric, the Suggestion Method. It describes the internal algorithms or methods that the systems use to produce suggestions. We identify the following possibilities:

- **Content Based**. These systems promote content to a user using his own history. The new suggested item are similar to the content previously consumed.
- **Based on Similar Users**. These systems promote to a user content suiting the taste of other users similar to him.
- **Based on Similar Items**. These systems promote to a user content similar to content he previous liked.

Table 2.4: Level of personalization and Suggestion Method of Information Filtering Systems

	Level of Personalization			Suggestion Method		
Service	None	User	Community	Content Based	Based On Similar Users	Based On Similar Items
Social Network		✓			✓	
Social Rating			✓			✓
Social Bookmarking		✓			✓	
News Aggregators	✓					
Personalized News Aggregators		✓		✓		
Trending systems	✓					
Item Based Recommenders			✓			✓
User Based Recommenders		✓			✓	
Content Based Recommenders		✓		✓		
Webrowse			✓		✓	

In Tab. 2.4, we show the Levels of Personalization and the Suggestion Methods in the different systems. Social Networks provide content personalized for each user. For example, every user on Facebook sees a different feed. To personalize the content, social networks

use the friendship relationships between users. Thus, the suggestion method is based on the assumption that friends share interests. Social rating tools present items grouped by categories. Users interested in particular topics visit the dedicated section of the website. Social bookmarking promotes personalized content. They suggest new items based on similar users. News aggregators and trending systems do not offer personalization. In the case of personalized news aggregators, each user has his own homepage and the systems suggests items based the content the users like. Item Based Recommender Systems suggest items similar to those that a user already liked. This means that all the people that like similar object (i.e. the community of people) that like similar objects, receive the same suggestions. User Based and Content Based recommender systems offer personalized suggestion at user level. The suggestion methods are based on similar users and on content features respectively. WeBrowse suggests content interesting for the community it monitors. The suggestion method is based on similar users, assuming members of the same community share common interests.

2.1.3 Lack of User Input: Cold Start and Sparsity of Data

In this section, we present the work about the open problems in designing information filtering systems. In particular we consider two of open issues: cold start and sparsity of data. The cold start problem happens when a system is too young to have enough data to infer what content to propose. To solve this problem Schein et al. [57] combine both collaborative filtering and content information to recommend unrated items in a collaborative filter based system. Lam et al. [58] approach this problem using information of the users. They base their work on the idea that people with the same features will also share the same interests. In the long term, the lack of content with user feedback is called sparsity of data. For example, in a system that requires users to explicitly evaluate items, most of the user can just free ride the system and never evaluate any item. To solve this problem Pazzani et al. [59] augment the rating information with personal information of the user (e.g. the age and the gender). The scientific community refers to this kind of approach as “demograhic filtering”. Huang et al. [60] model the problem as a neural network and use spreading activation algorithms to explore transitive associations among users. Another approach is used by Billsus et al. [61] and Sarwar et al. [62]. They reduce the original matrix containing the feedback using Singular Value Decomposition. Note that, as WeBrowse does not require user engagement, it is not affected by cold start problem and sparsity of data.

Finally, some work propose to augment the input of information filtering systems adding information abut the communities of the users. Notable examples of community based systems are [30, 31, 32, 33, 29], they demonstrate that in general social-based recommendation leads to better results than recommendation based on users’ preference similarity, in an heterogeneous set of taste-related domains (e.g. movie catalogues, club ratings, webpages and blogs).

2.1.4 Characterizing Existing Tools

A lot of work has been conducted on understanding and characterizing the information filtering systems that we exposed and analyzed. When users share content on social network they act as a filter for other users. Bakshy et al. study this effect in Facebook [63]. They relate the probability of sharing with other metrics like the number of comments an items has and messages a user received. Similarly, Kwak et al. [64] evaluate Twitter’s efficiency as a news propagation platform, and propose advanced criteria for generating user list recommendations.

Bakshy et al [65] predict the influence of users on twitter. They use for their predictor features the number of followers, number of tweets, date of joining etc. The scientific community has produced plenty of studies about Reddit. Singer et al. in [15] report the growing popularity of this website. They study five years of data and observe an exponential growth of the submissions. Gilbert [66] exposed the problem of underprovision in Reddit. In particular, this work shows that 52% of the most popular links are ignored the first time they are submitted.
Other works are more general and instead of focusing on single systems, they survey the systems similarly to what we do in Sec. 2.1.1. Adomavicius et al. [67] classify information filtering systems using their suggestions method. They divide the systems in Content-base, Collaborative and Hybrid. For each of these families they provide examples of two types: heuristic-based and model-based. Su et al. [68] survey the existing collaborative filtering techniques. They group the systems in memory based, model based and hybrid recommenders. For each group they cite some representative techniques and they list the main advantages and disadvantages.

Some of the existing tools (e.g. Reddit, Storify, Pearltrees) are crowdsourced. Crowdsourced means that their users share content on the tool. The tools use community votes to promote content. Although these are powerful tools, prior work highlighted some of their weaknesses. First, most users participating in platforms like Reddit are not active. This slows down information propagation in the community [69]. Second, although they target the web at large, such communities tend to become closed and self-referencing, with mostly user-generated content [15]. Third, they heavily suffer from "freeloaders", i.e. passive users relying on others actively sharing contents, and from a large amount of submitted contents which are not viewed [16]. Other tools like Facebook's Paper and Twitter Trends overcome these issues as they build on huge social networks to discover novel and popular content. Similarly, social bookmarking tools like Delicious and Pinterest, promote content based on users who organize collections of webpages.

2.1.5 Tools for Communities

All the tools described in the previous section share the problem of requiring users to be active at sharing content, which is difficult to obtain, in general [43, 70]. Differently, WeBrowse is a system for content sharing in communities of a place based solely on the passive observation of network traffic, and, to the best of our knowledge, it is the first system in its kind. WeBrowse's most similar system is perhaps Google Trends, which passively observes users' activity in Google Search (i.e., search queries) to extract trends about the most searched topics. The result however is different from the one produced by WeBrowse, as Google Trends returns the keys used for search queries, instead of links to contents.

Other proposals aim at personalizing the recommended contents to match users' interests [56], or to offer news based on a regional basis [71]. WeBrowse, in its current design does not offer personalized recommendations beyond the community-based aspect, but we plan to explore this aspect in future work.

Information filtering system for communities have also been deployed in workplaces. Zhao et al. [72] show the benefit of using Twitter at work. They show that the usage of Twitter in the workplace gives professional benefits (e.g. workers are aware of the work of other colleagues not nearby located) and personal benefits (e.g. workers develop a "sense of community" inside the enterprise). DiMicco et al. [37] show similar results. They design a social network for IBM and find that people use it to discover colleagues that worked on similar projects and that they

didn't know before. There are other systems that have been designed specifically for content promotion in communities-of-a-place scenarios such as campuses and neighbourhoods. For example, several enterprises and universities deploy internal social networks [37, 38, 39], create blogs [40, 36] and wikis [41]. Steinfield et al. [73] find that using a social network inside an enterprise increases the value of the social capital. The term social capital refers to the resources that derive from the relationships among people in varying social contexts and it is considered an asset by enterprises.

Finally, in systems that work in communities of a place, privacy is a sensible issue. Indeed, the set of users is in general small. For this reason, it can be easy to relate data present in system with their owner. A notable amount of work has been conducted investigating the relation between privacy and the usage of tool for communities. Some of these work are related to the design and deployment of WeBrowse. In fact, designing the system, we took into account the requirements of the privacy boards of the institutions hosting it (i.e. Inria and Politecnico di Torino). The seminal work about privacy perception is by Acquisti et al [74]. They use both surveys and information retrieved from the network to investigate the users' perception of privacy. They use as population for their studies a university campus. They find that undergraduate students underestimate the impact of the usage of social network in their privacy and also that some users have a unrealistic conception of the size of the network they share their information with. Similar surveys have been conducted also more recently [75, 76, 77].

2.1.6 Evaluating Suggestions

The traditional metric to evaluate suggestions is accuracy. For example, Netflix offered a prize of one million dollars to the BellKor's Pragmatic Chao team who was able to improve by 10% the accuracy of their algorithm [25]. To measure accuracy means to measure how the ranking proposed by a system differs from the ranking made by the user. Shani et al. [78] make an extensive description of prediction accuracy techniques and divide them in three classes. The first is measuring the accuracy of rating prediction. The second is measuring the accuracy of usage predictions. The third is measuring the accuracy of ranking items. Other work evaluate suggestions accuracy. Steck [79] tackles the problem of popularity bias. The popularity bias happens when users are more likely to provide feedback on popular items. In the case of WeBrowse, popularity bias happens because users click more on popular content generating a rich-get-richer effect. The author creates a model that gives more to less popular items in order to correct the bias. Accuracy is not the only evaluation metric available. McNee at al. [80] claim that using only accuracy as evaluation metric can hurt the systems. They propose other metrics such as: diversity of suggestion lists, serendipity and user needs and expectations. The lack of diversity in suggestion lists creates the effect that once a user consumes an item, he receives suggestions too similar. For example, a user buying a book on Amazon may receive as suggestion only books that are from the same authors. To solve this problem, Ziegler et al. [81] design a metric to measure the similarity of the proposed items. Serendipity is the experience of receiving unexpected input. Ge et al. [82] evaluate the relation between serendipity and accuracy. They show that an increase of serendipity means a decrease of accuracy and propose "explained suggestions" to mitigate this effect. Knijnenburg et al. [83] design a framework to evaluate the user experience using information filtering systems. They use questionnaires to correlate user experience with objective metrics.

2.2 Mining of Web Logs

WeBrowse is not the first system facing the task of mining HTTP logs for URL extraction. For instance, this idea was already proposed Jia et al. [84]. However, the result obtained with their approach, i.e., number of clicks to web portals, is not different from what services like Alexa already provide. Indeed, the main challenge behind this task resides in the detection of what we call user-URLs from the set of all URLs, and only a few other research studies address this problem [85, 86, 87, 88]. Among them, only StreamStructure [87] and ReSurf [88] are still adequate to investigate the complex structure of modern web pages. Neither of them, however, are appropriate for WeBrowse, as they rely on HTTP Content-Type, which is often unreliable [89], and makes traffic extraction more complex (because of the matching between requests and responses) and unsuitable for an online system as WeBrowse. Our experiments in Sec. 3.4 confirm this observation. Eirinaki et al.[90] survey the existing methods for web log mining. However, being dated 2003, this work is not up to date. Yang et al. [91] use some of the techniques we use in this work for web logs mining. For example, they detect the object composing a web page. However, as their goal is to improve caching and prefetching they do not need to reconstruct the page. More recently Neasbit et al. proposed ClickMiner [92]. ClickMiner has similarities with the click extraction part of WeBrowse. For example, they both reconstruct the pages using the referrer field of the HTTP request message. However, ClickMiner is based on a 2 steps procedure that makes it impossible to be used online. Neasbit et al. [93] use a completely different approach. They design WebCapsule. An extension for a browser that is able to log all the user activity. Naylor et al. [94] conduct a study about the diffusion of the HTTPS protocol. They find that it accounts for 50% of the total connections. They also state how the increase of its adoption will impact the network and web log mining. In particular they prospect the ending of the caching, compression, content optimization and filtering. Recently, Nelms et al. [95] proposed WebWitness. This tool tracks users' browsing activity that they call web paths. However, their goal differ from ours. Indeed, WebWitness is a tool that identifies threats (i.e. malware) to understand attack trends. Vassio et al. [96] propose an approach similar to ours for clicks extraction from HTTP logs. However, they focus on offline data and in improving recall an precision.

2.3 News Characterization

Since the introduction of online journals, researchers in different communities have seen the opportunity to study news consumption. Earlier studies [97, 98] focused on whether users would shift from traditional printed news to online news. These studies report an increasing trend on online news consumption, but news consumption in print is still significant. A part of this thesis focuses on the characterization of online news. This section briefly overviews studies on online news consumption.

User interaction with online news articles. Lagun et al. [99] study how long users interact in each part of a webpage. They identify four different user behaviors and design a model to predict user engagement that takes into account features of the page (e.g. the length of the text and the presence of an image). In this thesis, we have no metric of user engagement with news articles, so we focus on news visits. Lehmann et al. [100] study a particular kind of interaction of users with news. They identify a class of users that they call online content curator. They define an online content curator as "someone who continually finds, groups, organizes and

shares the best and most relevant content on a specific issue online". They also provide the features to identify these kind of users. They claim that this information is particularly useful for journalists and news editors.

News visit patterns and life cycle. A number of studies also focus on news visits. Dezsö et al. [50] study how users access news articles in a large news portal. Their analysis of the visits to news articles shows that most of the popularity of news articles decays after 36 hours. Castillo et al. [101] combine visit patterns and social media reactions to classify news articles into two classes. The popularity of breaking news articles tend to decay shortly after they are published; whereas in-depth news articles exhibit a longer shelf-life. They also show that social media reaction can help predict news visit patterns. Our analysis of news visits reappraises parts of these studies, but considering data collected across multiple online news journals and for a different population of users.

Topics of news articles. Zhao et al. [102] compare the topics in news shared in Twitter with those of the New York Times. They found that Twitter covers more personal life and pop culture and that Twitter users tweet less about world events but retweet a lot, which causes the news to spread. Zafar et al. [103] develop a novel method that selects the most relevant news articles related to a topic. They base this work on their previous observations [104], where they collected news article shared on Twitter labelled with a set of pre-defined topics. In contrast to previous work, Zafar et al. rely on trustworthy experts to extract news on specific topics from Twitter.

News on social media. In recent years, social media has risen as a means to share news. Kwak at al. [105] show that 85% of the topics in Twitter are related to news headlines. Lehmann et al. [106] use Twitter to identify transient crowd related to news article. A transient crowd is composed by users who tweet about a news article. Once they identify the transient crowd, they use it to propose to its members developments of the story they are interested in. Saez-Trumper et al. [107] identify three kinds of biases in news shared on Twitter: (i) Selection and coverage bias is correlated with geographical variables (news journals in an area select the same stories and write articles of similar length), (ii) statement bias is particular evident in social media, and (iii) social media tend to be much more focused on niche content. Wang et al. [108] study the news consumption from social media in China. They are able to characterize different types of audience for different types of news. Kourogi et al. [47] extract features from the headlines and text of tweeted news and use a SVM ranker to infer features that predict whether a news article will become viral. We show that only a small fraction of news visits in our datasets comes from social media.

Morgan et al. [109] show that users sharing news on Twitter present no bias based on their perceived ideology of the news journal, i.e., they see no bias due to selective exposure in their datasets. We also study selective exposure in our datasets with similar conclusions. Lee et al. [110] question the perceived noteworthiness of consumed news. They make a survey and show that only about one-third of the content produced by the mainstream news media is perceived as noteworthy. Boczkowski et al. [49] measure the gap between what publishers think is interesting and what users really read. Our analysis also contrasts the headline articles with news visits, or which articles readers actually read.

Selective exposure in news consumption. The selective exposure theory states that people expose themselves to information that strengthen they prior knowledge. This phenomenon is so strong that it can affect the decision making process [111]. Munson et al. [112], study the way users react to selective exposure. They set up an experiment proposing lists of news articles to

users. Each list has an amount of agreeable and disagreeable articles, meaning articles fitting the point of view of the user or not. They find that there two types of users: the diversity seeking ones and the challenge averse ones. They find that the satisfaction of the challenge averse users increases with the number of agreeable articles they see. While, for diversity seeking users, the satisfaction increases with the number of agreeable articles until it reaches a given value, after that it decreases. Bakshy et al. [113] show that a social media ranking algorithm (e.g. Facebook) reduces of 25% the amount of content coming from an opposite point of view with respect to a random ranking algorithm. However, they also show that at least 20% of the content one receive in his personal wall comes from an opposite point of view leaving room from ideological exposure. Munson et al. [114] design a browser extension that gives to users feedback about the lean of their news reading (i.e. conservative vs liberal). They show that after receiving the feedback, the users with a strong lean move to a more balanced reading. Park et al. [115] provide another solution to mitigate the selective exposure. They design NewsCube, a system the identify the salient aspect ad an event and group descriptions of such aspect from different sources. Doing this they augment the exposure of the users to multiple point of view.

2.4 Conclusion

In this Chapter, we presented the related work. First, we introduced information filtering systems. We noticed an explosion of tools coming from different horizons (the content curation family and the recommender systems family) with no clear link between such tools. Thus, we proposed a functional taxonomy that links all these tools. As the focus of this thesis is in communities of a place, we introduced the tools designed for communities of a place. We noticed that none of the existing solution is able to tackle with the biggest problem of systems for communities of place: the lack of user input. For this reason, in this thesis we design WeBrowse (see Chapter 4).

Second, we presented the tools that are able to mine web logs. We showed that none of these tools is fit to extract users' click from HTTP logs in an online fashion. This motivates the first contribution of this thesis that is the design of scalable online algorithms to extract user clicks from web traces. We present our solution in Chapter 3. In the same chapter, we compare our techniques of click extraction with those from the literature (i.e. we conduct an experiment to compare the performances of our techniques with the performance of ReSurf).

Finally, we focused on news consumption. We explored some branches of the literature: (i) user interaction with online news articles, (ii) news visits patterns and life cycle, (iii) the topics of news articles, (iv) the consumption of news on social media and finally the case of iv) selective exposure in news consumption. Again, we noticed the lack of studies focused on communities of a place. This led us to conduct our characterization of news consumption in communities of a place (see Chapter 5).

Chapter 3

Identifying Users' Clicks

In this thesis, we explore the possibility to use a novel source of data: a community users' clicks to web pages. We consider the case of users belonging to a community of a place, that are people living or working in the same geographical area. Users of a community of a place are connected to the same network. Thus, we want to obtain the users' clicks just observing this network. This information is not readily available and we need to design heuristics to extract clicks from the network traces. Indeed, when we look into network traces, we realize that the overwhelming majority of data is not useful to determine users' clicks. In this chapter, we use the terminology that we introduce in Sec. 1.3. Note that we distinguish between User-URLs and Browser-URLs. The first ones are the URLs that the users intentionally visit as opposed to Browser-URLs that are automatically requested by the browser. In Sec. 3.1 we describe our heuristics to detect User-URLs. In Sec. 3.2 we describe an algorithm to detect user-URLs online. In Sec 3.3 we show the dataset that we use for the evaluation of the accuracy of the algorithms that we present in Sec. 3.4.

3.1 Detection of user-URLs

Let us take again the example of a webpage access from Fig. 1.1 to illustrate our user-URL detection heuristics. As shown, the input log has five HTTP requests to load 'Web page #1', and four for 'Web page #2'. We need to identify only URL-1 and URL-1c that are actual user clicks, i.e., user-URLs.

To this end, we develop several filters that build a candidate list of user-URLs from a set of HTTP logs:

- **F-Ref.** First, we take advantage of the referer field to pinpoint user-URLs. As we see in Fig. 1.1, when the user visits 'Web page #1', all requests of embedded objects have URL-1 in the referer field. Hence, our first filter only considers URLs that appear in the referer field as candidate user-URLs. In our example, this filter would identify three candidate user-URLs: URL-1, URL-1c, and URL-3c. URL-3c is a CSS file with an embedded object. In general, F-Ref will capture the main object (which we want to label as the user-URL), but also objects whose hierarchical structure triggers the download of other objects (as URL-3c).

- **F-Children.** This filter uses the observation that modern webpages embed many objects,

that we call *children*. We get the number of children of a URL (say URL-1) by counting the HTTP requests with URL-1 as a referer (URL-1 has five children in the example). F-Children removes from the list of candidate user-URLs any URL with less children than a given threshold, min_c. This filter eliminates objects with simple hierarchical structure, e.g., URL-3c in our example. After applying F-Ref and F-Children in our example, we correctly obtain a set with two user-URLs: URL-1 and URL-1c.

- **F-Type.** Previous work proposes to use the type of the object to filter out browser-URLs [86]. We use a simpler variant that looks at the extension of requested objects to discard, e.g., *.js*, *.css*, *.swf* objects from the set of candidate user-URLs. This filter would also eliminate URL-3c.

- **F-Children-Type.** This filter allows requested object to be children on apage only if it's extension is an image or a javascript object. It checks the extension of the objects allowing only *.js, .jpeg, .jpg, .gif, .bmp, .png, and .swf*

- **F-Ad.** Webpages often embed advertisement objects, which could embed several other objects themselves. Their structure will thus fool previous heuristics. To solve this issue, we eliminate URLs pointing to known advertisement platforms using AdBlock's filter [116].

- **F-Time.** It groups together all HTTP requests that happen within a time window T. This heuristic discards, within the time window, all the URLs that come after the first candidate user-URL.

- **F-UA.** It checks the *user-agent* field exposed in HTTP requests to discard those generated by non browser applications (e.g., DropBox, Google Play Store).

- **F-parameter.** It builds on the observation that user-URLs correspond to queries containing few parameters (e.g., `http://www.acme.com?content=YYY`). F-parameter removes from the list of candidate user-URLs any URL that has more than a given threshold max_p parameters after the HTTP GET query string.

3.2 Online Algorithm

We design an algorithm that allows us to combine the heuristics to detect user-URLs to build several filtering configurations and process HTTP requests online.

Alg. 1 describes the algorithm that extracts candidate-URLs out of HTTP logs. We call candidate-URLs the ones worth being promoted. We give a more detailed definition of candidate-URLs in Sec. 4.2.1. It takes as input a stream of HTTP logs, **HS**. It gets four fields for each HTTP request: *<timestamp, URL, referer, user-agent>* and it returns a stream of candidate-URLs, **IS**, in the format *<timestamp, URL, referer>*.

This algorithm employs a hash table – the *Candidate Cache*, **C** – which stores historical information for every observed referer for a limited *Observation Duration*, T_O. As HTTP requests arrive, we keep only those with the user-agent in the browser white-list according to **F-UA** (line 4). Then, we extract the referer, r, and we check its presence in **C**. If r is not in **C**, we check the nature of its content with the **F-Type** filter (line 6). If r passes the filter, we

Algorithm 1 Online candidate-URL detector.

```
Input: HS, T_O, min_c, max_p # HTTP Request Stream, Observation Duration, and parameters for F_chidren and F_param
Output: IS # Candidate-URL Stream
# Init Candidate Cache
 1: C ← ∅
# Read current HTTP request
 2: while h in HS do
 3:     # h contains: timestamp, URL, referer, user-agent, UserID
        # Check user-agent and URL is different from the referer
 4:     if IS_BROWSER(h.user-agent) and h.URL != h.referer then
            # If current referer is not in Candidate Cache
 5:         if h.referer ∉ C then
                # If it passes type-based filter
 6:             if F_TYPE(h.referer) then
                    # Add referer to the Candidate Cache
 7:                 ADD(C, h.referer, timestamp)
 8:             end if
            # If h.URL is a valid child
 9:         else
10:             if F_CHILDREN(h.URL, min_c) then
                    # Increment the number of children and look for social
11:                 UPDATE_INFO(C, h.referer, h.URL)
12:                 GET_CANDIDATE_URL(C)
13:             end if
14:         end if
15:     end if
16: end while

17: function GET_CANDIDATE_URL(C)
        # Iterate all referers in the Candidate Cache
18:     for r in C do
            # Check T_O expiration and if it pass candidate filter
19:         if observation_time(r) > T_O and F_CHILDREN(r, min_c) and F_PARAMS(r, max_p) then
                # Send to the output
20:             write(r, IS)
                # Clean structures
21:             remove(r, C)
22:         end if
23:     end for
24: end function

25: function UPDATE_INFO(C, referer, URL)
        # Increment the number of children
26:     INCREMENT(C.refer.children)
27: end function
```

add it to the candidate cache $\mathbf{C}$ (line 7), which for a period of time equal to T_O will store the timestamp of the first request having r as referer, the number of children of r. Additionally we set a "social flag" to true if we observe a social plugin child for r. We explain extensively this flag in Chapter 4 to chose which URL is to be promoted. If r is in $\mathbf{C}$, we keep updated such information with `update_info` (line 12).

Finally, we call the function `get_candidate_URL` (at lines 8 and 13). For each referer in $\mathbf{C}$, we check if its observation duration T_O has expired. If so, it means that we can run the heuristics to label the URL as user-URL (for example, if the number of children is larger than threshold min_c) and as candidate-URL if the flag signalling the presence of social plugins is true (line 20). If the referer is interesting, we return it to the candidate-URL output (line 21). Note that this algorithm can also output user-URLs.

Table 3.1: Details of the traces considered in this study.

Trace	Network	Period	Users	HTTP requests
ISP-week	ISP	13-20 Jun 2013	65,577	190M
ISP-PoP1-1day	ISP	14 Apr 2015	13,238	16M
ISP-PoP2-1day	ISP	14 Apr 2015	3,019	1M
ISP-PoP3-1day	ISP	14 Apr 2015	18,462	25M
Campus-day-1	Campus	14 Apr 2015	10,787	21M

3.3 Dataset

To design WeBrowse we need to take some decision using network traces from the real world. We make use of these traces to tweak our heuristics. In this section we introduce the traces we use for this purpose. In a second moment (Sec. 4.4) we evaluate WeBrowse's modules using the same traces. We use two types of HTTP logs: ground truth traces and traces collected at real networks.

Ground-truth traces. We generate HTTP logs in a fully-controlled testbed similar to previous work [87, 88]. We manually visit the top-100 most popular websites according to Alexa ranking. When we are in the main page of each of these sites, we randomly visit up to 10 links they reference. We collect all the visited URLs as they appear in the browser bar. In parallel, we capture all the HTTP requests. We call the resulting HTTP log *HTTP-Alexa*. This trace contains a total of 905 user-URLs, corresponding to 39,025 HTTP requests. We also build a similar trace, *HTTP-GNews*, by visiting almost 1000 news sites (user-URLs) from Google News. This trace contains 68,587 HTTP requests.

HTTP logs. For this study we employ several HTTP log traces we collect at the backbone link of the campus network of Politecnico di Torino and at residential networks from a large ISP. We collect the residential traces at three different routers located in three cities in Italy. In all cases, we obtain the traces by running Tstat [117], a passive probe which processes live the stream of packets in the network and tracks all HTTP requests. Table 3.1 summarizes the traces we employ in this study.

3.4 Accuracy

We evaluate the accuracy of our user-URLs detection algorithm, and we compare it with the best performing state of the art. For the evaluation we employ the *HTTP-Alexa* dataset and use two metrics: (i) *Recall*, i.e., the number of true positives (or URLs correctly labeled as user-URLs) divided by the number of positives (which is the total set of user-URLs). (ii) *Precision*, i.e., the number of true positives divided by the number of true positives plus false positives (which is the number of URLs our heuristics say are user-URLs).

In our experiment we compare different combinations of filters, and different parameter tunings. Tab. 3.2 presents the results for the most accurate filters[1]. We present a more extensive performance description in Appendix A. This table also compares with ReSurf [88], the state of the art to detect user-URLs out of HTTP logs. ReSurf also relies on the referer field to separate

[1] We exclude from this table the URLs to webpages delivered with HTTPS protocol.

Table 3.2: Performance and processing time achieved by ReSurf and our best performing combinations of heuristics for the detection of user-URLs on *HTTP-Alexa* (44686 requests).

Method	Recall	Precision	Processing time
ReSurf	82.37%	90.33%	32484ms
F-Ref + F-type + F-Children(2) + F-Param(0)	82.97%	90.52%	1270ms
F-Ref + F-type	**96.51%**	**63.60%**	**1251ms**

HTTP transactions into multiple streams, minimizing the temporal overlap between clicks on different URLs (e.g., when a user keeps several tabs open, for the same time window the technique can identify different streams). Notice that ReSurf relies on the HTTP *content-type*, extracted from the response.

First, we observe that F-Ref + F-type + F-Children (2) + F-Param (0) filters when running online can achieve the same accuracy as ReSurf, if not slightly better (82.97% of recall and 90.52% of precision). More importantly, our algorithm is lightweight, and, when comparing the processing time on the same trace, we find that our algorithm is around 25 times faster than ReSurf. Second, our results show that there is a tradeoff between recall and precision. Indeed, F-Ref + F-type + F-Children (2) + F-Param (0) increases the precision (removing false positives) but comes at the cost of lower recall (it fails to detect some user-URLs).

Fortunately, we find that applying the candidate-URL filter on top of the user-URL filters has a positive side effect: it increases the user-URL detection precision to 100%. For this reason, for our final implementation of WEBROWSE, we choose the filter with the best recall, F-Ref + F-type, and let the candidate-URL filter remove the remaining false positives.

Finally, we run additional tests to see the impact of increasing the observation time T_O, from 5 seconds upto 30 seconds. We find the best trade-off for $T_O = 15$ seconds, even if this choice has very limited impact. We omit results for brevity.

3.5 Conclusion

In this chapter, we introduced our solution to extract users' clicks to webpages from network traces. We focused on the case of communities of a place. First, we designed several heuristics to identify user-URLs. We call user-URLs the URLs that users intentionally request and browser-URLs those that the browser requests automatically. Second, we designed an algorithm to use the heuristics in an online fashion. Third, we evaluated the online algorithm using HTTP traces. In the next chapter, we use the user input obtained from the heuristics described here to design, deploy, and evaluate a novel information filtering system.

Chapter 4

WeBrowse: A Passive Information Filtering System for Communities of a Place

In this chapter, we introduce WeBrowse, a passive information filtering system, that does not need user engagement. It works by processing the users' clicks and outputting promoted content for that group of users. In this work, we use for the design, the deployments, and the evaluation of the system, the users' clicks extraction techniques described in Chapter 3. However, any source of clicks is suitable to be the input of WeBrowse. In the first section of this chapter, we give a general overview of the system. We show the modules composing WeBrowse and briefly describe them. In the second section, we describe WeBrowse's design. First, we introduce the dataset that we use to take decision in the design procedure and after, we describe in details each of WeBrowse's modules. In the third section, we show the deployments of WeBrowse. WeBrowse is currently deployed in two different locations that are (i) the campus of an university in Italy and (ii) a research lab in France. We describe how we realize the deployments and the challenges that we face. In this section, we also evaluate the performance in terms of computational resources needed by WeBrowse. In the last section, we evaluate how good is WeBrowse in promoting interesting content. First, we measure WeBrowse's liveness. Then we conduct a user study and measure some features peculiar to WeBrowse. In particular we verify the existence of the community of a place effect and show how WeBrowse complements other information filtering systems. Finally, we conclude with some considerations about ethical and legal issues.

4.1 Overview

Fig. 1.2 presents an overview of WEBROWSE. WeBrowse takes as input HTTP requests from a raw data extraction module and outputs lists of URLs. These URLs are promoted to user on a presentation module. In outrcurrent deployments the presentations module is a website. We describe in this section the high-level architecture of our information filtering system.

Data extraction is a a traffic monitor running within the network, it observes packets

traversing a link and extracts HTTP requests.[1] In this work, we use the monitoring tool Tstat [117] for extracting HTTP requests, but this module can build on whatever packet capturing infrastructure ISPs already deploy (e.g., [119]). Alternatively, we can extract HTTP requests from web proxies or even from plugins installed on user devices. We are interested in the following information from HTTP requests: *timestamp*, *URL* (obtained by chaining the information in *host* and *resource* fields), *referer*, and *user-agent* fields. We also extract an anonymized user identifier.[2] All fields are extracted from the HTTP requests only, hence we ignore responses.

WEBROWSE consists of four sub-blocks as depicted in Fig. 1.2. The *user-URL filter* identifies the HTTP requests corresponding to actual users' clicks, that we call *user-URLs*. It eliminates the vast majority of HTTP requests that the browser automatically generates and we call *browser-URLs*. In this block we implement the heuristics described in Chapter 3. Filtering out browser-URLs is not enough. Indeed some of the user-URLs are not worth being promoted. For example, websites of search engines and online banks may be popular but this is not the kind of content a user expects to find in his information filtering system. To filter out the not interesting URLs we add the *candidate-URL* and the *content versus portal* modules. The *candidate-URL* module select the URLs that are worth being promoted. In this module we implement heuristics to discover URLs that can be shared with other users. The *content versus portal* discriminate between content URLs and portal URLs. An example of portal URL is the homepage of a news journal, while an example of content URL is an news article from that homepage. WeBrowse aims to promote the single news article rather than the homepage. For this reason we filter out portal URLs. Finally, the *promotion* module takes as input this set of URLs (together with their timestamp) that from the bottom layer passed all the filters and output them to the presentation module. In the promotion module we implement our promotion algorithms. In the evaluation section we study these algorithms asking to the user which one they prefer.

Presentation presents the promoted URLs to the users a user-friendly web portal (similarly to Reddit). It is an interface similar to news aggregation and curation services. We have two different versions for this module. The first one is interactive. It is a website that users can browse. In the website they have the possibility to select different views of the promoted URLs. For example they can choose how long in the past to retrieve the filtered data or they can select particular categories of data they are interested in (e.g blog, video or news). The second version is thought to be shown in fixed monitor, for example in common spaces. As the users can not navigate this interface it updates automatically and periodically shows different portion of data. For this interface we use the tool wonderboard [3].

[1] Although HTTPS is gaining popularity [118], our analysis (not shown for conciseness) shows that only 7% of websites WEBROWSE aims at promoting actually relies on encryption for data delivery. In a possible future when HTTPS will be dominant, we can still envision passive content curation based on alternative solutions, e.g., a browser plugin or corporate proxies that feed the passive promotion system.

[2] In Sec. 4.2.3, we discuss why WEBROWSE needs this information and describe the techniques we adopt to preserve users' privacy.

[3] wonderboard is provided by Nokia Bell Labs and it is available at http://www.getwonderboard.com/

4.2 Design

In Sec 4.1 we introduce WeBrowse and give an overview of its modules. To make the user-URLs extraction we use the heuristics that we extensively describe in Chapter 3. In this section we describe the remaining modules. First, we describe the candidate-URLs detection module. Here, WeBrowse select the URLs that are worth being promoted. Then, we describe the content/portal discriminator module. WeBrowse uses this module to tell whether a URL points to a portal or to a normal content web page. Finally, we describe the promotion module. This module is in charge of selecting the URLs to promote to users.

4.2.1 Detection of Candidate-URLs

We aim to select, among user-URLs, those worth sharing, what we call *candidate-URLs*. Users often visit URLs related to web-mail or e-banking, but these are not the kind of content they would appreciate as a recommendation. Our goal is then to identify the URLs that WEBROWSE should promote. We note that the presence of social buttons in a webpage is an explicit indication that a user may want to share its content. Hence, our approach passively inspects the HTTP requests in the logs to detect children URLs pointing to well-known social network buttons (e.g., Facebook Share button) widely adopted in the Web.[4]

This method labels as candidate 70.72% of the URLs contained in *HTTP-GNews*, the ground-truth of candidate-URLs we built visiting webpages promoted by Google News. We find that the remaining 30% of URLs corresponds to websites embedding custom social sharing buttons, which deceive our heuristic (e.g., YouTube). Although we can reverse-engineer some of these ad-hoc methods to improve the accuracy, we leave it for future work given the complexity of this task.

To understand how WEBROWSE filters candidate-URLs in practice we apply the heuristics to detect user-URLs together with the heuristics to identify candidate-URLs on three days of *ISP-week*. The user-URLs represent a tiny fraction of all the observed URLs as expected. Out of 190 million requests we identify 6.5 million user-URLs. Among these, only 422,500 are candidate-URLs, corresponding to 0.22% of the total number of visits.

To implement this heuristic we expand the algorithm 1 to enable it to count the social buttons in each page. We add a check in the function $GET\_CANDIDATE$. In particular in line 19 we check if the referer r has any social plugin adding the condition $if HAS\_SOCIAL\_PLUGIN(r)$. Moreover we need to modify the $UPDATE\_INFO$ function to count the social buttons as we show in algorithm 2

Algorithm 2 UPDATE_INFO function modified to count social buttons to detect candidate URLs

```
1: function UPDATE_INFO(C, referer, URL)
       # Increment the number of children
2:     INCREMENT(C.refer.children)
       # Check the presence of social plugins
3:     if IS_SOCIAL(URL) then
4:         SET_SOCIAL_PLUGIN(C.refer)
5:     end if
6: end function
```

[4]List available at www.retitlc.polito.it/finamore/plugins.txt

4.2.2 Content-URLs versus Portal-URLs

This section describes how WEBROWSE distinguishes candidate-URLs pointing to web portals from those pointing to specific content. We use the term *web portal* (or *portal-URL*) to refer to the front page of content providers, which generally has links to different pieces of content (e.g., `nytimes.com/` and `wikipedia.org/`); whereas a *content-URL* refers to the webpage of, e.g., a single news or a Wikipedia article. We first design an offline classifier. We then engineer an online version.

Classification Features

Given the heterogeneity of the URL characteristics, we opt for a supervised machine learning approach to build a classifier. We choose the Naive Bayes classifier, since it is simple and fast, thus, suitable for online implementation. As we will see, it achieves good performance, not calling for more advanced classifiers.

We use five features to capture both URL characteristics and the arrival process of visits users generate.

- **URL Length.** It is the number of characters in the URL. Intuitively, portal-URLs tend to be shorter than content-URLs.

- **Hostname.** This is a binary feature. It is set to one if the resource in the URL has no path (i.e., it is equal to "/"); and to zero, otherwise. Usually, requests to portal-URLs have no path in the resource field.

- **Frequency as Hostname.** This feature counts the number of times a URL appears as root of other candidate-URLs. The higher the frequency, the higher the chances that the URL is a portal.

- **Request Arrival Process (RAP) Cross-Correlation.** The URL request arrival process is modeled as a vector in which each element is the number of visits in five-minute bins. We notice that users often visit portal-URLs in a diurnal periodic pattern. Intuitively, the more the request arrival process signal of a given URL is "similar" to that of a well-known portal, the higher the chances that the URL corresponds to a portal. To capture such a similarity we employ the cross-correlation, a well-known operation in signal processing that measures how similar two signals are, as a function of a sliding time lag applied to one of them. We compute the maximum cross-correlation between (1) the request arrival process of a tested URL and that of (2) well-known portals (e.g., `www.google.com` or `www.facebook.com`). The higher the value of the maximum of the cross-correlation, the larger the chance of a URL being a portal.

- **Periodicity.** This is a binary feature that builds on the observation that users visit portals with some periodicity. We use the Fast Fourier Transform (FFT) on the discretized aggregate visit arrival process for a given URL. If the visit arrival process shows one-day periodicity (principal frequency of 1/24h), then we set periodicity to one; zero, otherwise.

Table 4.1: Accuracy of the web portal/content classifiers.

Features	Portal		Content		Accuracy
	Precision	Recall	Precision	Recall	
Hostname	100%	23%	55%	100%	60%
Hostname + RAP Cross-correlation	100%	75%	80%	100%	87%
Hostname + Periodicity	100%	82%	84%	100%	91%
Hostname + Periodicity + RAP Cross-correlation	100%	87%	89%	100%	93%
URL length + RAP Cross-correlation	100%	76%	80%	100%	87%
URL length + Hostname	100%	88%	88%	100%	93%
URL length + Periodicity	**100%**	**93%**	**94%**	**100%**	**96%**
All features	100%	93%	94%	100%	96%

Some of these features require accumulating information in the system and pose practical constraints for an online implementation, which we address in Sec. 4.2.2.

At last, note that some features work only for portals that are popular enough to observe enough clicks to detect the periodic diurnal cycle. Given that our promotion mechanisms (described in Sec. 4.2.3) only promote URLs that are somewhat popular, it is unlikely that WEBROWSE will present to users URLs of unpopular portals.

Feature selection

We build a dataset to train and test the accuracy of classifiers based on different features from *ISP-week*. We manually visit each candidate-URL extracted from *ISP-week* in order of appearance to label each as content-URL or portal-URL. We perform this task until we get 100 URLs of each type. Note that to overcome class imbalance, we follow the approach of oversampling the minority class, i.e., portal-URLs [120]. Then, we randomly divide the resulting 200 URLs into two sets: two thirds of the URLs for training and one third for testing. We use a ten-fold cross-validation, averaging results from 10 independent runs.

Tab. 4.1 presents the accuracy of classifiers based on different combinations of features. The table shows the recall and the precision in identifying portal-URLs and content-URLs, as well as the overall accuracy of the classifier. Identifying portal-URLs is an easy task: all combinations achieve a 100% precision. However, lots of content-URLs are misclassified as portal-URLs. Interestingly, the hostname feature alone miserably fails to discriminate between portal-URLs and content-URLs. This is due to the fact that a lot of portal-URLs do not match their hostname (e.g. `www.nytimes.com/politics/`). Adding RAP and/or periodicity we enhance the performance, achieving 100% precision and 93% recall for identifying web portals, and more

important, in our case: 94% precision and 100% recall for identifying content-URLs. This means that when using this combination we correctly label all the portals as portal-URLs, and we have a small probability (6%) that the classifier wrongly labels portal-URLs as content-URLs. At last, we obtain the same performance when considering all five features, suggesting that frequency, hostname and RAP do not bring more information.

When engineering the online system we consider both the combinations URL length and hostname, and URL length and periodicity. The first combination allows an immediate decision, while the latter requires the system to collect information for some days.

Algorithm for online classification

The online algorithm we use to distinguish content-URLs from portal-URLs in real time takes as input a stream of tuples <candidate-URLs, timestamp> and labels them as content-URLs/portal-URLs. The algorithm sends URLs labeled as content-URLs to the promotion module, which will decide which ones to present to users.

The best performing feature combination, URL length and periodicity, requires us to collect in a database the timestamps of requests to candidate-URLs for some days. Therefore, we engineer the workflow of the algorithm to rely on this combination as soon as enough observations are available, and to use an on-the-fly classifier as a temporary solution. For the latter, we choose the best performing among the combinations that can execute on-the-fly, i.e., URL length and hostname.

Fig. 4.1 depicts the workflow of the algorithm. We employ the *Knowledge Database*, $\mathbf{K}$, which the algorithm populates with the portal-URLs obtained by running the more precise classifier based on (URL-length, periodicity). As soon as the algorithm receives a new candidate-URL i, it checks its presence in $\mathbf{K}$. If present, it immediately returns the classification result tagging the URL as a portal-URL. Otherwise, the algorithm counts the number of observations it has collected for i. If the number of observations is large enough ($\geq W$), the algorithm classifies i using the classifier based on (URL-length, periodicity), and stores the outcome c_i in $\mathbf{K}$ for future decisions. Otherwise, the algorithm performs the classification on-the-fly using the classifier based on (URL-length,hostname).

4.2.3 Privacy-Preserving Promotion

Once WEBROWSE identifies content-URLs, it must decide which ones to promote. Inspired by Reddit, we design content promotion modules based on both popularity and freshness. These two metrics are easily measurable from a network perspective. In addition, we separate content-URLs according to their type: news, blog, video, and other content. We detect news using a predefined list of traditional news websites. In our deployments, we construct this list by crawling, every 20 minutes for a period of one week the hostnames associated to the news appearing on the frontpage of the local Google News edition (i.e. Italy and France). After one week of crawling we obtain more than 500 distinct news websites in each country that are indexed by Google News. For videos, we create a list with the most locally popular video sites (e.g., YouTube, Dailymotion, and video sections of popular newspapers). For blogs, we crawl the local blog indexes to get the list of the top-30k most popular ones. Finally, we label the content-URLs which do not fall in these categories as other content.

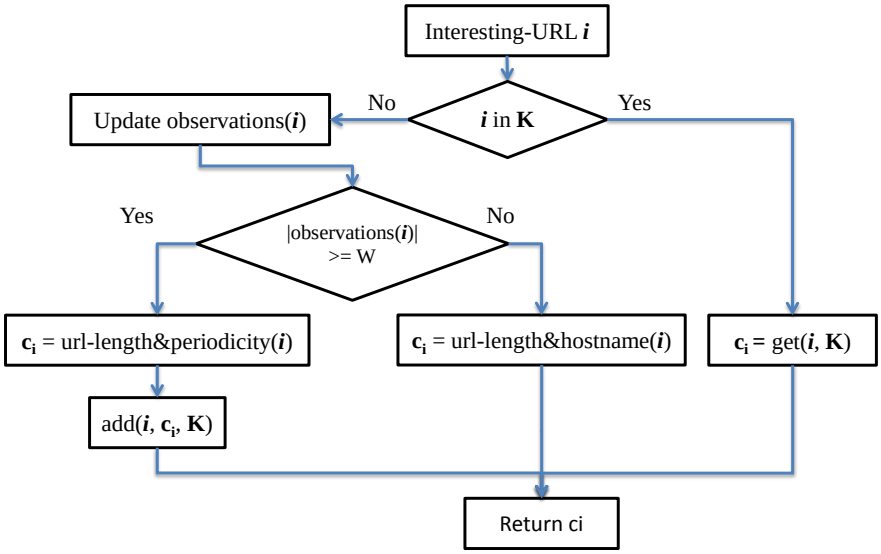


Figure 4.1: Workflow of the algorithm for online content vs portal URL classification.

Privacy threats

One of the most important requirements for passive crowsdourcing is to protect users' privacy. For WEBROWSE, we identify, in collaboration with the privacy and security boards of our institutions, three main privacy threats that are related to content promotion (we conduct a similar analysis for the data collection part):

- Personal information can leak into promoted URLs. For example a URL of a website showing users' login credentials. We design promotion mechanisms that prevent this threat.
- Private pages with open access could be promoted. For example, WEBROWSE could promote the URL of a hosted private image. Luckily, such URLs do not fall under our definitions of candidate-URL.
- Inferring personal information by combining WEBROWSE's data with information from other sources (e.g. list of persons present on site, or particular preferences of certain users). The risk analysis conducted by the boards labeled this latter threat as very improbable.

Promotion

Having in mind above issues, we build three promotion modules in the WEBROWSE's website.

- *Live Stream.* This module promotes web pages that are freshly clicked and belonging to news, videos, and blogs categories.
- *Top.* This module promotes web pages by building a rank based on their number of clicks. In the current version, we show the ranking for different time spans: one day, one week, and one month.

- *Hot.* This module builds on Reddit's Hot ranking algorithm [121], which promotes URLs that are both popular and recent. This algorithm assigns each content a score based on users' up and down votes. The links are then presented by decreasing score. For our case, we replace votes with the number of clicks, and modify the ranking score to obtain the following:

$$Score(u) = log(N_{views}(u)) + \frac{T_{first}(u) - T_0}{T_P}$$

  where, for each content-URL u, $N_{views}(u)$ reports the number of clicks, $T_{first}(u)$ is the timestamp corresponding to the first time u has been observed, and T_0 is the timestamp corresponding to an absolute reference (i.e., the start of WEBROWSE deployment). Finally, T_P is a normalization factor defining the "freshness period". We set it to 12 hours. Intuitively, this score finds a balance between the content popularity and its recentness (i.e. its age with respect to the absolute reference). URLs new to the system get larger T_{first}, and consequently, a larger second term of the formula, increasing their probability to reach the top rank positions because of their freshness, even if yet relatively popular. However, when a URL stops attracting attention (i.e., clicks), the first term of the formula stops increasing, and the URL is likely to get surpassed by fresher URLs. Finally, the hot module keeps and permanently updates the scores and rankings of content-URLs. Content URLs are then presented to the users by decreasing score.

To protect users' privacy from the threats described above, we adopt the following techniques. For the case of Live Stream module, we protect users' privacy by *(i)* promoting only content coming from public, trusted and known hostnames, and, as an additional security measure, *(ii)* stripping all the parameters contained in URLs. For the Top and Hot modules, we opt for k-anonymity, i.e., a content-URL is allowed to be promoted iff WEBROWSE has observed at least k different userIDs (built by combining the IP address of the clients and the User-Agent field in the header of HTTP requests) accessing it. We choose k=5 and do not show URLs older than 24 hours. With this configuration, the same user has to visit the same URL, assuming it contains sensitive information, from five different devices in one day for her personal information to be exposed. Finally, we allow the users to opt-out from the system by enabling the DoNotTrack flag in their browsers, and ignoring all HTTP requests containing the flag. These policies have been discussed and agreed with the Security and Privacy boards for our deployments.

Removing WEBROWSE bias.

Finally, since our modules query promoted URLs to create previews, we have to remove the effect of WEBROWSE on itself. Similarly, content promoted on the portal has a higher probability of being visited by the users. Both of these artifacts inflate promoted URLs' popularity. To counter this effect, we instrument the system to ignore requests having WeBrowse's website as a referer.

4.3 Deployments

This section describe our deployments in the campus network of Politecnico di Torino and in the Inria Paris network. Before, we evaluate the amount of resources needed for the system to

Table 4.2: Sizes of the deplyoments in Politecnico di Torino and Inria Paris in terms of number of daily users, daily requests and daily URLs

Network	Daily Requests	Daily User-URLs	Daily Candidate-URLs	Users
Politecnico di Torino	7M	55,000	25,000	15,000
Inria Paris	1.3M	27,000	5,000	1,200

work. To this end, we rely on *ISP-week*, which aggregates the HTTP traffic from about 20,000 households.

We measure the time that WEBROWSE spends to process the trace using a server with off-the-shelf hardware configuration (2.5GHz CPU and a 32GB RAM). Our results, show that WEBROWSE's implementation is lightweight enough to process up to 4M HTTP requests in less than one hour using a marginal memory footprint, and demonstrate that WEBROWSE can sustain the processing of such large rates of HTTP requests on a rather standard hardware configuration.

We install a Tstat probe at the egress vantage point of the monitored network, where we can observe HTTP requests to/from the Internet. The probe then streams HTTP logs to the WEBROWSE server, equipped with a quad-core 2.66GHz CPU and a 4GB RAM, which is enough to sustain the HTTP request in this scenario. Even at peak times, the system performance is stable. The CPU load rarely overcomes 20%, and the memory footprint is stable at around 1.68GB.

Table 4.2 summarizes the sizes of the two monitored networks. The campus network aggregates approximately 15,000 users – students, professors, and staff – regularly accessing the Internet. The WEBROWSE server processes in real-time HTTP logs containing on average 7M requests per day. It extracts on average 55,000 user-URLs, corresponding to 25,000 candidate-URLs each day. The Inria Paris Network aggregates around 1,200 users. In this deployment we process 1.3 M requests corresponding to 1,200 daily user-URLs and 5,000 daily candidate-URLs. The server extracts also content-URLs and then promote a subset of them in the WEBROWSE website,[5] which implements the three promotion tabs discussed in Sec. 4.2.3. The deployment at the Politecnico di Torino includes also one additional tab that displays the live stream and top webpages having the university's domain (i.e., the university webpages attracting clicks from the Internet). In the deployment at the Inria Paris, we exchange this last tab with one displaying research papers. Each tab contains a content feed inspired by the "wall" implemented in popular social networks such as Facebook and Twitter. URLs in the feed have a preview image, a title, and a short description when available.

In addition to data being promoted, we also store results so that we can compute offline statistics and analyze WEBROWSE's behavior and performance. We must also ensure that our data collection effort respects users' privacy. We only collect clicks to content-URLs (i.e., content-URL, timestamp). We neither store the IP address the visit came from nor the anonymous user ID. The anonymized user-ID is only kept in RAM for 15 hours and then discarded. All data is stored in a secured server with restricted access. This deployment got approval from the Security Office of Politecnico di Torino (equivalent of an IRB approval in the United States).

[5]The website for the campus deployment is public and can be accessed at: http://webrowse.polito.it/.

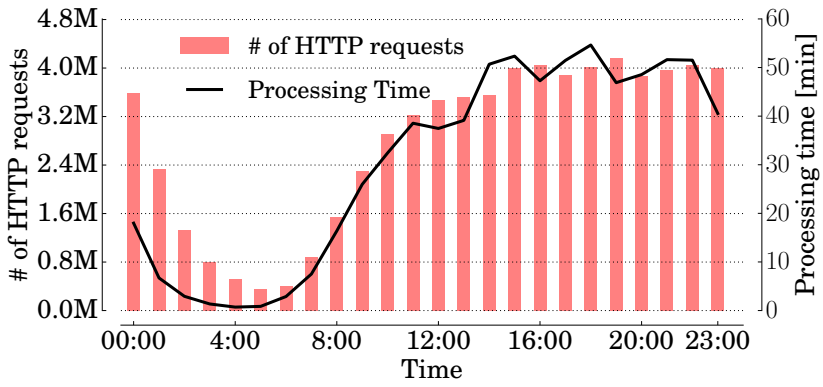


Figure 4.2: HTTP requests rate (left y axis) and processing time (right y axis) over time for one day extracted from *ISP-week*.

We advertise WEBROWSE in the campus network broadly starting on the beginning of March 2015 in two rounds (to different mailing lists of the university) reaching around 6000 users. Such advertisement campaigns generated 1636 visits between March 27th and May 1st 2015.

As one important goal of the deployments is to get feedback from users and to study their interaction with WEBROWSE, we enclose a link to an evaluation form on WEBROWSE's front page that users can use to rate the service.

4.3.1 Performance Evaluation

We evaluate the performance of our WEBROWSE implementation to process requests in *ISP-week*, which aggregates the HTTP traffic from about 20,000 households. We select the day in *ISP-week* with the largest peak hourly rate of HTTP requests. We split the one-day trace in 1-hour long sub-traces, and use them to feed WEBROWSE. For each hour, we measure the time that WEBROWSE spends to end the processing. For this experiment, we run WEBROWSE on a a server equipped with a 2.5GHz CPU and a 32GB RAM. Fig. 4.2 reports the amount of HTTP requests (left-hand y axis) and the corresponding processing time (right-hand y axis), for all the 1-hour long bins in the day of *ISP-week*. The figure shows that WEBROWSE's implementation is able to complete up to 4M HTTP requests in less than one hour. This demonstrates that WEBROWSE can sustain the processing of such large rates of HTTP requests on a rather standard server like the one we pick. Finally, we note that WEBROWSE's memory footprint is minimal, as less than 960MB of memory was used throughout the experiment.

We are convinced that the content promoted by WEBROWSE can only become more interesting with a larger population of users. When scaling to, e.g., a country-wide ISP network scenario, WEBROWSE will be asked to analyze the HTTP traffic of millions of users, which is challenging. Hence, to let the system scale, we could spatially distribute WEBROWSE deployments, or adapt WEBROWSE to work based on BigData paradigms. We leave the exploration of this aspects for future work.

4.4 Evaluation

Evaluation challenges. Evaluating a system like WEBROWSE is very challenging: there is no "perfect-match" competitor to WEBROWSE in the campus community, and existing platforms are not really comparable. Reddit is not popular in Italy and France because only few users contribute to it. Our user poll (see Sec. 4.4.2) shows that users in the campus rely mainly on news media and Facebook to get informed. Comparing WEBROWSE to news media is not fair because news is only a small part of what WEBROWSE promotes (see Sec. 4.4.4). What we do instead is to check whether the topics of the news that users consume in the campus match what the Italian version of Google News promotes (see Sec. 4.4.3). Analyzing Facebook feeds of users in the campus is not possible, because it is hard to get a representative number of volunteers. Finally, Twitter has no trending data based on the campus location.

Instead, we focus our evaluation on a number of objective and subjective metrics. The objective metrics will help us answering the challenges that come with WEBROWSE: What is the critical mass of users for WEBROWSE to work? (Sec 4.4.1) And is there really a community of a place effect? (Sec 4.4.3). The subjective metrics consist in collecting users' feedback on WEBROWSE's content (Sec 4.4.2). We evaluate WeBrowse using the campus network deployment.

4.4.1 Effect of the crowd size

WEBROWSE depends on users accessing the web from the network to identify the content to promote. The more users on the network, the more dynamic WEBROWSE will be and the more clicks the promoted content will have. We measure, for our campus deployment, the *liveness* of WEBROWSE with the number of new added links in the Hot tab, and study how many users are needed for WEBROWSE to be "alive". Fig. 4.3 presents three sub-plots reporting statistics collected every 30mins. The bottom plot presents the number of active users in the network, which we approximate by the number of active userIDs; the middle plot presents the total number of user-URLs; and the top plot reports the number of URLs which appear for the first time in the Hot tab (new Hot URLs). We observe a typical day/night pattern with the number of active users which grows during working hours and decreases during the night and weekends. The number of new Hot URLs presents a positive correlation with the number of active users. In the plots we compare the data from the campus trace with data from ISP trace. We notice that in the campus trace there is not new Hot URLs during night and weekends. This is expected, as many users leave the area during these days. In the Campus, we rarely observe new Hot URLs when the number of active users is below 200. The Hot tab is clearly more dynamic when there are at least 800 active users in the network. As the users of the ISP trace are more active and produce more User-URLs, the Hot Tab requires less users to be dynamic. In the ISP trace we observe new Hot URLs starting from 100 users. The higher activity in the ISP trace is partly due to the large amount of internal-network URLs that people in the Campus consume and we do not take into account in this plot.In particular, there are no new Hot URLs during nights and week-ends. In this deployment, we rarely observe new Hot URLs when the number of active users is below 200. The Hot tab is clearly more dynamic when there are at least 800 active users in the network. However, this scenario is challenging for WEBROWSE because of the relatively low browsing activity of users in the campus. In fact, results for the ISP deployment show that the number of active users needed to the Hot tab to be dynamic is even lower.

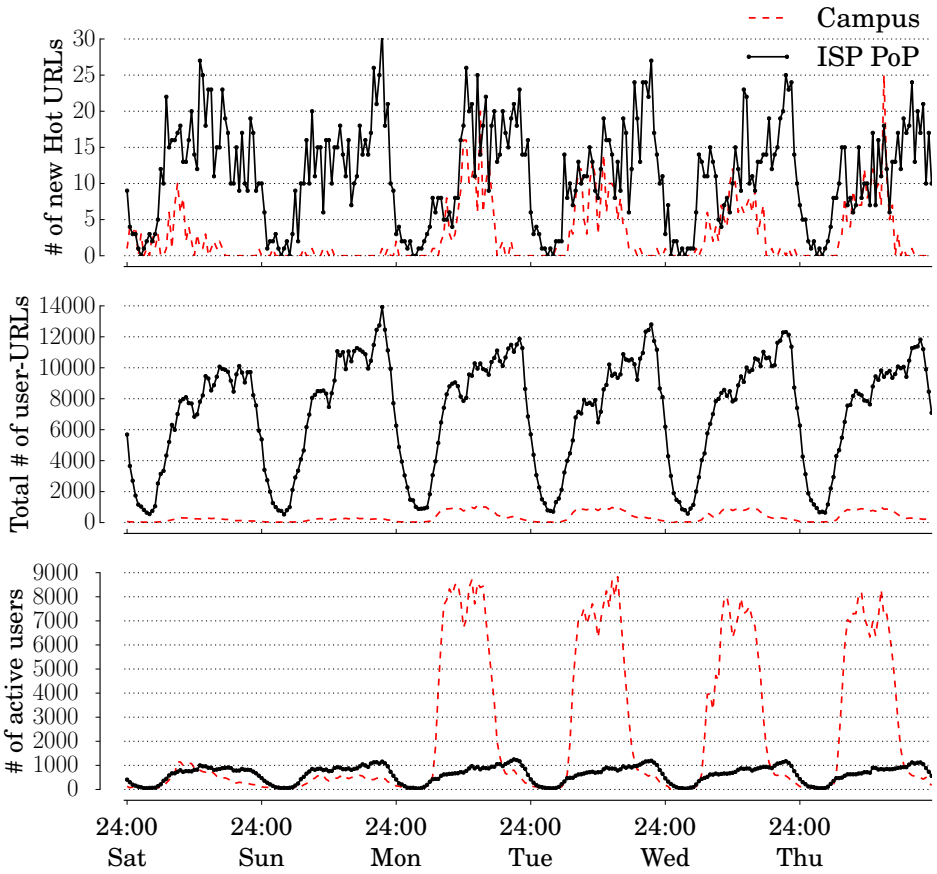


Figure 4.3: The number of active users observed by WEBROWSE (bottom), and number of never seen before URLs in the Hot category (top). Statistics collected from WEBROWSE's campus deployment every 30min.

4.4.2 User study

We study users' interaction with the WEBROWSE site using the results of Google Analytics and the feedback they leave in the evaluation form. One week after the email announcements, we observe visits coming from 1506 distinct users.[6] Interestingly, we observe that 20% of users keep visiting the website after their first visit. The average visit is 2.5min long. Out of the users who visited the website, 115 (8%) filled our evaluation form. 93% of those who were contacted during the first announcement (R1 in short) are mostly students and professors from the Computer Science and Electronic departments of Politecnico di Torino. We specifically targeted these users to collect feedback with a more technical nature. The second round of advertising (R2) reached a wider population, i.e., professors, researchers, and students from different areas (engineering and architecture) and administrative employees. Although the number of respondents is small, their feedback is extremely valuable for our future work on WEBROWSE.[7]

[6]Google Analytics can count these numbers very precisely thanks to the cookies it installs in the client.
[7]The response rate we obtain is in line with that of surveys with no invite incentive [122].

Table 4.3: User feedback from the evaluation form.

How interesting is WEBROWSE's content?	R1	R2
extremely interesting	8	24
very interesting	25	19
interesting	17	10
poorly interesting	5	1
not relevant	4	12

How useful is WEBROWSE?	R1	R2
extremely useful	4	14
very useful	20	20
useful	18	16
poorly useful	10	1
not relevant	7	4

Which service do you use to keep informed?	R2
Web Newspapers	25
Facebook	12
Google News	4
Twitter	3
Newsletters	2
Other Media	10

How do you compare this service to WEBROWSE?	R2
More interesting	5
Less interesting	7
Simply different	41

We summarize the feedback of the 115 respondents in the following. We split the questions in our evaluation form into two main groups. The first group helps us evaluate whether users like WEBROWSE and the second focuses on our promotion methods. Not all respondents answered all questions.

Do users like WEBROWSE? We ask questions about their experience with WEBROWSE and to compare WEBROWSE with the service they use to discover content on the Web. Tab. 4.3 summarizes the responses. Overall, respondents were positive: the wide majority of respondents find WEBROWSE at least interesting or extremely interesting. Similarly, 71% in R1 and 91% in R1 of respondents find it useful or extremely useful. Interestingly, responses during working hours (9am to 6pm) found WEBROWSE more interesting than answers in other hours. This positively correlates with the dynamic behaviour of WEBROWSE.

We also ask users in R2 to list the services they usually use to stay informed, and compare them to WEBROWSE. As shown in Tab. 4.3, 25 of respondents rely on news portals to keep informed; Facebook comes second (12 respondents). Interestingly, 41 respondents say that WEBROWSE is simply different from these services. These answers are encouraging as we see WEBROWSE as a complement, and not a replacement, to existing curation systems.

Finally, the 62% (70%) in R1 (R2) say they would like to have WEBROWSE as a service offered by their network. 44% (30%) in R1 (R2) would use WEBROWSE at least once a day.

How good are WEBROWSE's promotion algorithms? We ask users to rank the three different tabs in WEBROWSE (Top, Hot and Live Stream) using a Likert Scale from the most interesting to the least interesting. We calculate the average ranking score for each tab. The scores are fairly close, with the Top tab coming first, then Hot, and Live Stream as last. This result indicates that users have different tastes, and having different tabs with different promotion methods is important to please a large user population.

4.4.3 Community of a Place Effect

One of the main arguments for passive crowdsourced content promotion is its usefulness in the case of a community of a place. In this section, we characterize the community-of-a-place effect by comparing the content promoted by WEBROWSE in different communities. We aim to understand to what extent hot content varies across different communities of a place, and whether there is an actual community effect beyond the simple influence of the country and geographical location. To achieve this goal, we study popular topics in different communities of a place and compare them with each other. We also compare topics consumed by distinct randomly chosen populations of the same community of a place.

We use the 1-day log traces (same day) described in Tab. 3.1. As shown, each trace corresponds to a different network scenario and thus community: the campus and three different ISP PoPs in the same country. For each trace, we extract the popular topics. First, we run WEBROWSE on the set of HTTP logs in the trace to pinpoint the corresponding content-URLs. Then, we access each content-URL, scrape the webpage and extract title and main text. For each webpage, we tokenize the text, remove stop words, and use term frequency to identify the top-10 most relevant terms, which we call *keywords*. We then weight each keyword by its overall *popularity*, i.e., the sum of visits on all articles in which it appears: if a keyword appears in two pages that were viewed 3 and 5 times, the keyword's popularity is 8. We then consider the interests of users in the trace as the set of weighted keywords. For a fair statistical analysis, we compare keyword sets extracted from traces whose user populations have similar size. Thus, from each trace we randomly extract subsets of users so that all the populations we compare have equal size. We set the default population subset size to 3,000 users, which corresponds to the number of users we observe in our smallest vantage point. We perform different runs using different populations and average the results.

Next, we compare the keyword sets extracted from different population pairs. For a qualitative comparison, we draw a scatter plot for each pair. Dots in plots correspond to keywords, and coordinates match their popularities measured in the two populations. To quantify the correlation between keyword popularities, we measure the Pearson correlation coefficient. Note that a perfect linear relation would imply the keywords are equally popular in the two user sets, corresponding to a Pearson correlation coefficient equal to 1. Finally, as we run each comparison five times, we show one of the five scatter plots we obtain, and report the average Pearson coefficient.

All in all, we test four scenarios. In the first case, we compare the keywords we extract from distinct populations obtained by randomly picking users in *Campus-day-1*. Fig. 4.4(a) shows the results for this case. Notice how the dots (i.e., keywords) are distributed on the main diagonal of the graph and the Pearson coefficient is close to 1 (0.89). This result suggests the presence of a community effect within the campus network. Second, we consider a residential network scenario and we compare in Fig. 4.4(b) distinct populations we obtain from *ISP-PoP1-1day*. Even if smaller than in the campus case, the correlation of keywords' popularity is large, testifying the presence of a community of a place effect also in this case. This is not surprising as people in the same city or neighbourhood typically share common interests (e.g., local news). We consider two other scenarios. First, we compare keywords from the campus trace *Campus-day-1* with those extracted from the ISP trace *ISP-PoP3-1day*, collected at a PoP in the same city of the campus. The results, reported in Fig. 4.4(c), show a low correlation. This demonstrates that despite accessing the web from the same city, the interests observed

in the two populations are different. Fig. 4.5 and Fig. 4.6 show the word clouds of popular keywords from the campus trace and from the ISP trace. We observe that those of campus trace reflect the typical activity of a university (e.g. “study”,“scholarship”,“project”). The most popular keywords from the residential trace relate more to local services (e.g. “court”, “hotel”,“reviews”)

Finally, we compare keywords of populations belonging to two different cities. Fig. 4.4(d) shows the scatter plot reporting keywords from *ISP-PoP1-1day* and *ISP-PoP2-1day*. As expected, the correlation of keyword popularity is lower compared to that of two populations belonging to the same city or the same campus.

We conclude that different populations in the community of a place share common interests, confirming our intuition about the usefulness of passive content promotion in this scenario. Interestingly, our results underscore a community effect that is stronger in the campus community compared to the geographical-based one. Indeed, although people in the same neighbourhood have more common interests than people in different cities, shared interests are higher for people in the same campus.

4.4.4 Complementing Existing Systems

Our user study shows that users think that WEBROWSE is simply different from the services they use to get informed. In this section, we study the content WEBROWSE promotes to shed light into this reasoning. We compare the frequency of keywords promoted by WEBROWSE across webpages with the frequency of those promoted by (1) a news curator (Google News), (2) an active crowdsourced system (Reddit), and, finally, (3) traditional news media.

WEBROWSE and Google News Similarly to what we did in Sec. 4.4.3, we use the keywords analysis to compare interesting topics in WEBROWSE and Google News. In particular, we focus on news media and analyze the difference between the news that Google News promotes on its homepage, and those consumed by the community of the campus and captured by WEBROWSE. On one side, we obtain a set of URLs corresponding to the news promoted by the main page of Google News for the same day when *Campus-day-1* was collected. On the other side, we extract from *Campus-day-1* a similar number of popular *news-URLs*, i.e., content-URLs whose hostname appears in the Google News list of around 500 publishers in Italy. We remind that these news-URLs appear in both the live new stream and the Hot News sections of WEBROWSE. Then, for each of the two URL sets, we extract a set of keywords, and we weight each keyword with its frequency, i.e. the number of times the keyword appears across webpages (URLs). In total, we count around 2,819 distinct keywords for *Campus-day-1* and 2,232 for Google News. We depict in Fig. 4.4(e) the keywords with their frequencies. Interestingly, the correlation is extremely weak, suggesting that the users in the campus are interested in different news than those promoted by Google News. This observation can be explained by two facts: first, we observe that the campus users often consume news-URLs that are one to few days old while Google News mostly targets fresh news. Second, the news promoted by Google News have country-wide interest and do not reflect the tastes of the campus users, which are however influenced by the location.

Interestingly, although Google News tends to promote the most recent news (from few minutes to 2 hours) on its frontpage, what the users really view is not that fresh. Indeed, more than 44% of news articles keep getting views from 1 to several days after their publication. A system like WEBROWSE would offer here a different perspective that is not linked to the

freshness of the news, but rather to what the crowd is viewing, which is not always fresh news.
WEBROWSE and Reddit Similarly, we compare keywords extracted from the campus network (*Campus-day-1*) with those we extract by web scraping the Italian version of Reddit's portal on the same day of the trace, considering a similar number of URLs. In this case, we observe that the correlation between the keyword sets is very weak (Pearson coefficient equal to 0.012). By manually inspecting the Reddit keywords, we observe that the topics differ from those in the campus, a large fraction of them refers to leisure activities or to Reddit itself. Moreover, we observe that the number of keywords in the campus set is much higher, which is due to a larger diversity of content.
WEBROWSE and traditional news media We perform further analysis to understand how much users rely on traditional news media for content curation. To this end, we quantify the fraction of news-URLs we observed since the beginning of our deployment (in March 2015) and the remaining portion of content-URLs, i.e., not published by a news website in the Google News list and that we name *not-news-URLs*. We find that news-URLs represent 14% of overall consumed content-URLs. In addition, not-news-URLs are published by a much larger set of domains (nine times larger then the list of news domains). Interestingly, in our deployment not-news-URLs tend to be often more popular than news-URLs.
Content locality. We apply a language detection software, Google Language Detection [123] , on the titles and text previews of the promoted content-URLs, and we find that around 94% of them are in Italian, around 2% are in English, and the rest covers 33 distinct languages, reflecting the diversity of the students in the campus. More interestingly, around 5% of content-URLs contain in the title or text description the name of the city or the region of the campus network. Few articles even relate to the neighborhood of the university and some to the university itself. This result highlights that WEBROWSE natively offers a regional service.
Content diversity. We quantify the fraction of content-URLs we observed since the beginning of our deployment (in March 2015) that were published by online newspapers. We use the Google News list of around 500 publishers in Italy (see Sec.4.2.3) to split content-URLs in two parts: the first contains *news-URLs*, content-URLs originated by one of these publishers; the second set contains the rest of content-URLs, that we name *not-news-URLs*. We find that news-URLs represent only 14% of overall content-URLs. In addition, not-news-URLs are published by a much larger set of domains (nine times larger then the list of news domains). This result confirms our intuition about the fact that WEBROWSE captures a large diversity of content. Finally, in our deployment not-news-URLs are always more popular than news-URLs.

4.4.5 Speed of Content Discovery

This section measures the speed at which WEBROWSE discovers new Internet content, and in particular news. A service like Google News has robots that actively look for new content to index using a predefined list of news publishers [124]. WEBROWSE, instead, relies on users exploring the web looking for fresh news. We set the bar high and leverage our campus deployment to compare the two approaches to discover news.

To compare WEBROWSE to the Google News approach, each time WEBROWSE promotes a content-URL to "Fresh news", we check if Google has already indexed it.[8] If so, we measure since when (this "age" is an information available below each link returned by Google Search).

[8]We use several instances of a headless browser to do a "site:" search on Google Search for each news-URL WEBROWSE detects.

We ran this experiment for a period of one day (after that, Google has banned our IP network because of the extensive probing).

Fig. 4.7 shows the number of news not indexed by Google News in each hour of the day (left-hand y axis). We put this number in perspective with the total number of "Fresh news" consumed per hour (right-hand y axis). The figure shows that even though we deploy WEBROWSE in a network with only a few thousand active users, it was able to find few not-yet-indexed news-URLs. Not surprisingly, there is a positive correlation between the number of viewed news-URLs per hour and the number of news-URLs in WEBROWSE that have not yet been indexed by Google News.

We do not expect WEBROWSE to compete with a large-scale active crawling system like Google News on the speed of news discovery. But clearly, increasing the number of users would increase the speed of content discovery. This would mean deploying passive crowdsourced content curation in a setting that goes beyond the community of a place scenario we explore in this work. However, popular toolbars, a search engine like Google or a social network like Facebook have however access to a much larger number of clicks and could try implementing this.

4.4.6 Ethical and Legal Issues

Deploying a system like WEBROWSE comes with a series of ethical and legal issues.

First of all, the users accessing the Web from the monitored network might suffer from the "Big Brother effect", i.e., experiencing the fear of having someone spying on their browsing activity. Surprisingly, however, after the first announcement of WEBROWSE, only two people expressed some privacy worries, an union official and one respondent to the evaluation form. In the subsequent announcements and talks presenting WEBROWSE, we took care of describing the privacy-preserving mechanisms WEBROWSE adopts, and highlighted that these were approved by the Security and Privacy boards of Politecnico di Torino, Inria, Nokia and the ISP collaborating with us. Moreover, we enriched the website with a section describing these mechanisms.

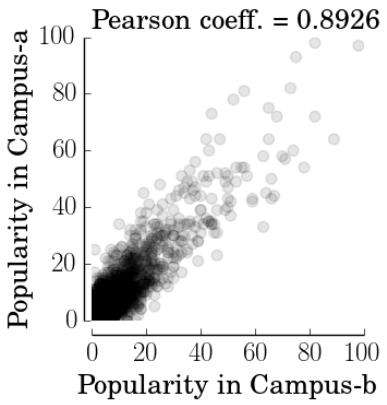
Finally, WEBROWSE might promote URLs pointing to inappropriate or potentially damaging content (e.g., porn or malwares). Also, the promoted content may expose internal information that the deploying institution is not willing to share. Although, we have not faced any of these issues yet, we plan to allow the deploying institution to create blacklists to filter out unwelcome hostnames and users to notify contents they consider inappropriate or offending.

4.5 Conclusion

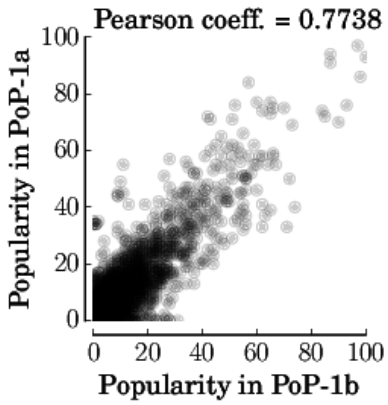
In this chapter, we introduced WeBrowse, our passive information filtering system based on web-clicks to boost content discovery in communities of a place. WEBROWSE takes as input a stream of HTTP requests observed in a community and processes it online to infer a set of URLs to suggest to its users. We have deployed WEBROWSE in a large campus network, and used this deployment for evaluation. We also deployed it in a research center of around 1,200 researchers to test it on smaller crowd.

Of the campus users who evaluated WEBROWSE, more than 90% welcome the quality of WEBROWSE content, and judged it as different from what they usually use to get informed. Our analysis of WEBROWSE's liveness shows that passive crowdsourcing becomes more interesting

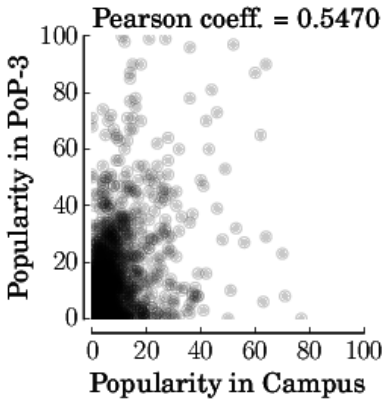
starting from around 1000 active users browsing the web. Finally, our analysis of the topics observed in different communities of a place confirm the promise of passive crowdsourcing to foster content discovery in such under-engaged communities.



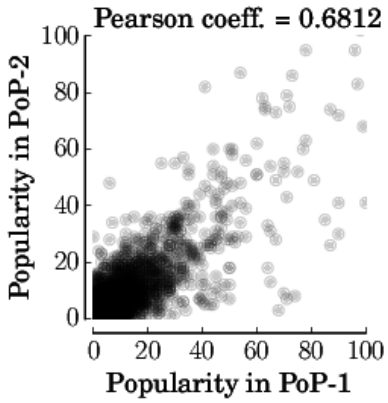
(a) Same Campus



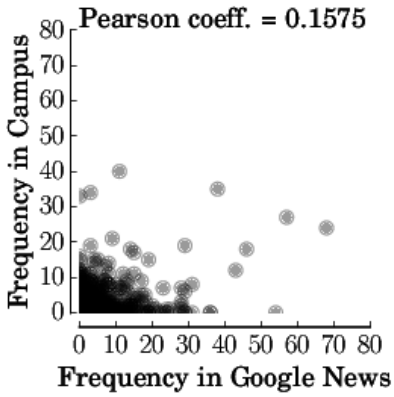
(b) Same city



(c) Campus vs. Residential



(d) Different cities



(e) News

Figure 4.4: Scatter plots where dots represent keywords extracted from different trace pairs and coordinates correspond to their popularities (Figs. 4.4(a)-4.4(d)) and their frequency in news webpages (Fig. 4.4(e)).

Figure 4.5: Wordcloud of popular keywords in the campus trace

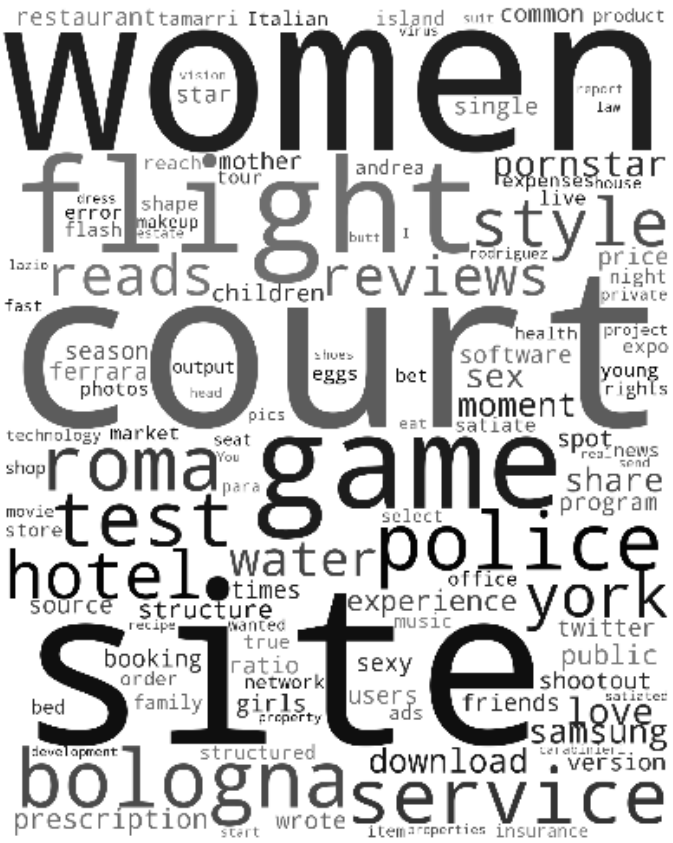


Figure 4.6: Wordcloud of popular keywords in the ISP trace

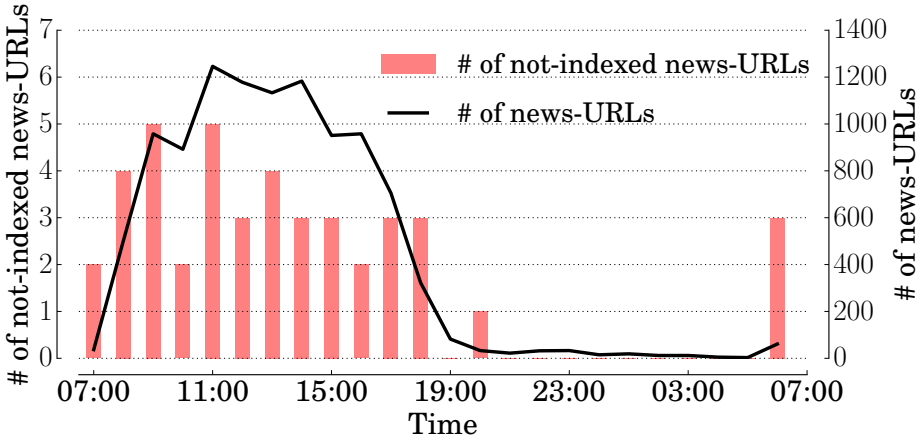


Figure 4.7: Number of not-indexed news-URLs (left y axis), and number of news-URLs (right y axis) during one day (Online deployment).

Chapter 5

News Consumption Characterization

Humans have always given a relevant importance to the exchange of news. With the advent of the web, this activity has been slowly moving to the Internet. The usage of the web, rather than the traditional paper medium, gives the possibility to conduct research about news consumption. Plenty of studies about users' news consumption habits have thus recently emerged. These studies are useful for socials scientists, but also for news producers who use these insights to improve their offers.

The literature has focused on different fields, such as the propagation of news, their coverage in social networks and traditional media, and users preferences with respect to editors choices. However, as every study considers a single journal or a single social network, we lack studies that collect data from multiple sources. In this chapter, we characterize news consumption using entire browsing histories of different communities of a place.

5.1 Datasets

Our analysis relies on news visits we extract from network traffic traces. We collect data from four different network links that aggregate traffic of tens of thousands of users. In this section, we describe how we collect network traffic traces and our methods to extract news visits and to identify users from these datasets.

5.1.1 Data collection: Method and Vantage Points

We analyse data collected from four network links: one at one university campus in Italy (*Campus*) and three in the network of a large residential ISP in two different Italian cities (*ISP-Bologna*, *ISP-Turin1*, and *ISP-Turin2*). In the campus, we monitor the link that connects the campus to the Internet, so we can observe when people visit online news sites from the university. Note that there is no student dorms in the university campus, so most of the online activity we observe in *Campus* is during working days and hours. In the ISP, each link we monitor aggregates traffic from one district with thousands of households. We can then observe all news visits of members of the monitored households when at home. *ISP-Bologna* was collected in one district in Bologna, *ISP-Turin1* and *ISP-Turin2* in two different districts in Turin.

The traces were obtained by running Tstat [117], a passive network monitoring tool. Tstat captures all network packets traversing the monitored links and extracts HTTP requests (i.e.,

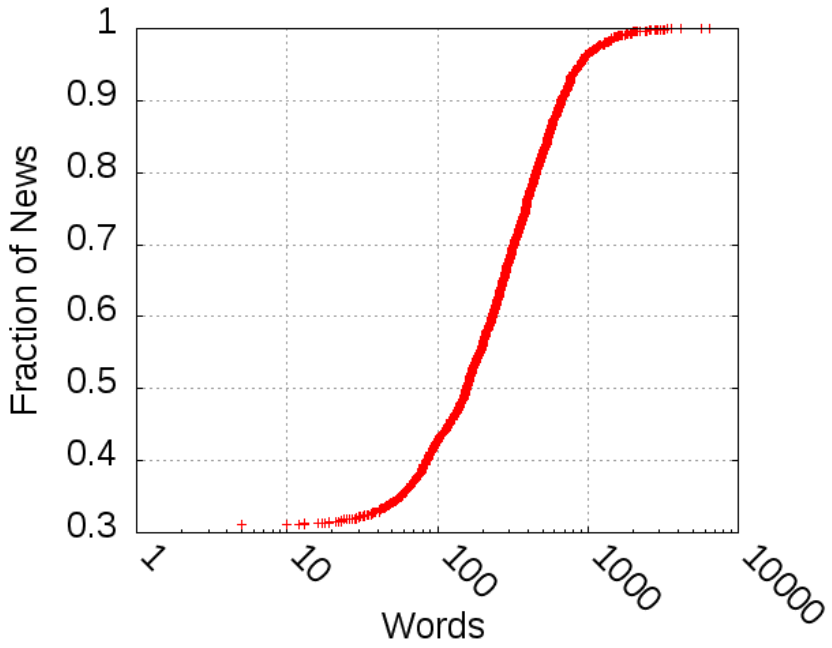


Figure 5.1: Lengths in words of news articles in one day.

a web visit). More precisely, for each webpage visit, we extract the visited URL, the referral (which captures the URL of the previous page visited by the user, if any), an anonymized IP address, the timestamp of the visit, and the user agent (which captures the browser, operating system of the client originating the request). Our data collection got approval from the Security Offices of the campus and the ISP (equivalent of an IRB approval in the United States).

5.1.2 From HTTP Traces to News Visits and Articles

HTTP traces are noisy—they contain a vast majority of requests to objects, scripts, or ads that compose webpages, but that are not the page users visited. We first must extract web visits from the traces. To do so, we apply our heuristics described in Chapter 3.

We detect which of these visits corresponds to news by selecting only the hostnames of well-known online journal sites. We build the list of hostnames of online journals in Italy by crawling Google News Italy once every 20 minutes during one month in 2015. The resulting list has 667 online journals. Google News does a thorough job in indexing news and our manual verification shows that the list covers all the popular online journals in Italy. Our analysis, however, may miss some foreign online journals or niche journals.

We also analyse the content of the news articles users visit. For each news URL in our traces, we crawl the URL to extract title and text. In this process, we identified that 37% of news URLs were either broken or the article was unavailable by the time of our crawling. We also identified a number of news URLs that did not point to proper news articles. In some cases the URL was pointing to one item in a slide show, for instance. To filter out these corner cases, we analyse the distribution of text length of all crawled news articles. Fig. 5.1 shows the cumulative distribution function of the lengths in words of news articles in one day. We see, that 32% news articles have less than 20 words. For the remainder of this work, we only

Table 5.1: Description of datasets (collection from January 2015 to May 2016).

Trace	Network	City	Households	Surfers	Distinct news
ISP-Bologna	ISP	Bologna	12,193	285,038	1M
ISP-Turin1	ISP	Turin	1,760	116,730	0.8M
ISP-Turin2	ISP	Turin	11,520	493,184	1.8M
Campus	Campus	Turin	6,635	1,102,348	1.8M

consider news visits if the visited article had at least 20 words.

5.1.3 Identifying Users: Households and Surfers

One challenge for our analysis is that our traces have no per-user identification. A single user may appear multiple times in our traces, because she may connect from multiple devices or because her IP address may change over time. Reversely, multiple users may share a single IP address, for example, all members of a household. To overcome this challenge, we study "users" at two granularities: surfers and households. We define a *surfer* as the concatenation of an anonymized IP address and a user agent. A surfer captures a particular user surfing the web from a mobile device or a particular browser. Note that with this definition, the same person might appear as multiple surfers. This is particularly likely if we take into account the entire duration of our traces. During our long observation period, user agents might change due to software upgrades. We define a *household* as an anonymized IP address. This corresponds to a household in our ISP traces. However, in the case of the campus traces, it simply corresponds to an allocated IP address.

5.1.4 Dataset Description

In this work, we analyse data collected in the four links from January $1st$ 2015 to May $15th$ 2016. Table 5.1 summarizes all our traces. The large number of surfers compared to households is due in part to the presence of multiple devices behind the same IP, but mostly because of software upgrades that change the user agent. Indeed, over a period of only one week, the average number of observed distinct user agents per IP is around three.

Fig. 5.2 presents the number of news visits per hour of the day in our datasets. We see the typical daily pattern with almost no news visits during the night and a significant increase during day time. In *Campus*, as expected there are practically no news visits during the weekends and outside of working hours. *ISP-Bologna* and *ISP-Turin2* news visits pattern is typical of a residential neighbourhood with an increase in news visits after 8pm when people are back from work. *ISP-Turin1*, however, is in between a typical work access pattern we see for *Campus* and the residential patterns. The neighbourhood where we collect *ISP-Turin1* also has a number of offices and we suspect that some smaller companies subscribe to residential Internet access plans.

5.2 News Referrals

We study the most popular news referrals in our datasets. Table 5.2 presents the results for *ISP-Bologna* (the conclusions remain across datasets). "Self referral" means that the visit

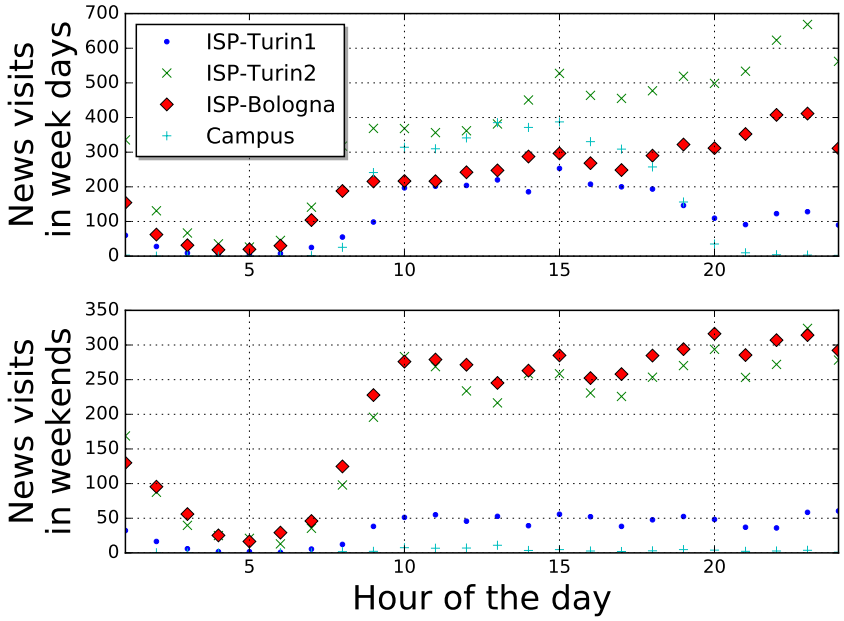


Figure 5.2: News visits per time of day in the four datasets (considering one hour time bins). The top plot is for week days and the bottom plot for weekends.

comes from a webpage with the same hostname as the visited news article. Most of these visits come from the home page of the online journal itself. "Direct browsing" corresponds to HTTP requests with an empty referrer. These visits can be cases when users click on a news article link in an email, for instance, or when the previous webpage was in an HTTPS website that does not pass the referer field (unlike Facebook and Twitter).

We find that more than 55% of all news visits are self referrals. This result implies that users still rely mostly on visiting the website of their favorite online journal to discover news. Facebook is the second most frequent news visits referral. Still it accounts for a relatively small percentage of all news visits in our dataset; and Twitter's share is much smaller. News visits with Google search referral account for 13%, which suggests that users were intentionally looking for news about a particular event or story.

This relatively small fraction of news visits from social networks is interesting as many

Table 5.2: News referral share in *ISP-Bologna*

Source	**Share**
Self referral	55.67%
Facebook	15.33%
Google	13.04%
Direct browsing	10.6%
Others	3.87%
Google News	1.05%
Twitter	0.43%

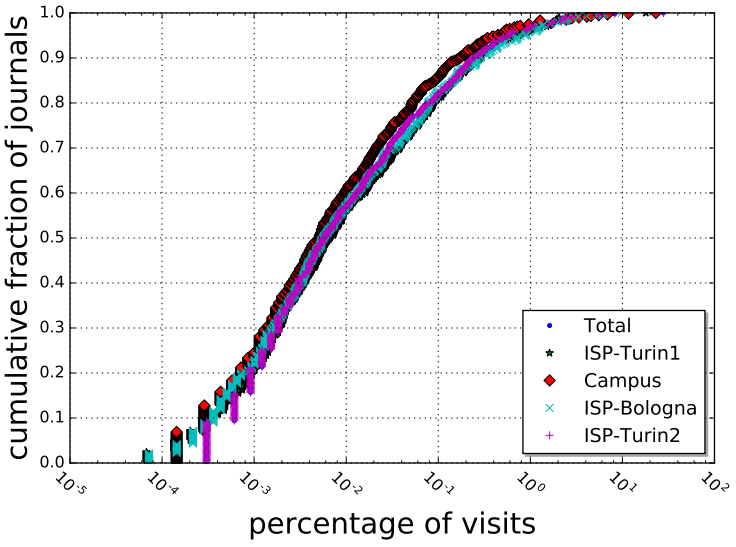


Figure 5.3: Cumulative distributions of popularities of journals in 16 months

studies of news consumption rely on news shared in social networks. Clearly understanding what people actively share is still relevant, but it is important to understand that these only represent a small fraction of visited news articles. To verify that our datasets do not introduce a bias concerning social networks, we study the percentage of active Facebook and Twitter users in our dataset.[1] We find that 65% of surfers in our datasets visited at least once Facebook in 16 months, whereas only 7% visited Twitter. These percentages are similar to the available statistics of Facebook and Twitter usage in Italy [125, 126]. Note that the fraction of Facebook users in the US is also similar, but that Twitter is less deployed in Italy than in the US (with three times less Twitter users). We also note that our result is consistent with a previous study of visits to the Al Jazeera English site [101].

Since online journals are the most influential news referral method, we study how often individual online journal sites appear as self referral. Fig. 5.3 shows the cumulative distribution of the percentage of self-referral news visits across online journal sites. The figure shows a large disparity between online journals: 80% of online journals are referral in less than 0.1% of news visits, while each of the top-5 journals are referral 31% of all news visits. Fig. 5.4 shows as an example the top 20 journals in terms of referral share in *ISP-Bologna*. We also compute the top 20 journals using the other traces. The top journals were similar in other traces with some differences in their ranking. This rank of top journals has all the large Italian newspapers, but also some topical journals: financial, sports, and local news sites. We next explore which categories of news are popular with users in our datasets.

5.3 News Categories

We study the most popular categories of news in our datasets. We label news articles with categories as follows. Whenever available, we use the category of news assigned by the news

[1] For this study we use the full TCP logs available with our datasets.

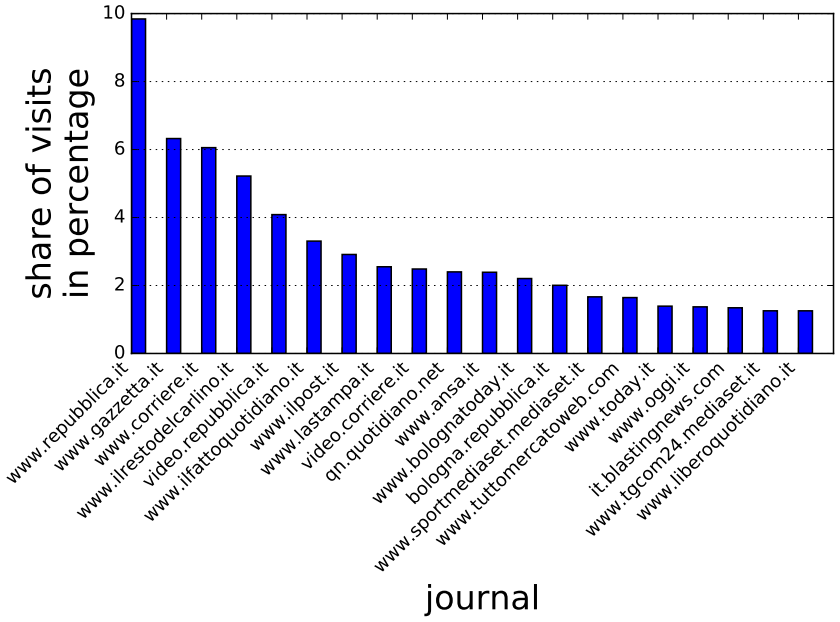


Figure 5.4: Share of visits of the top 20 journals in *ISP-Bologna*

editor. Using this method, we identify 167 category names in the four datasets. Since different editors may give different names to the same categories, we manually merge similar categories. We similarly manually group sub-categories into larger ones. After applying this procedure, we obtained 40 categories, which we use in our analysis, and we were a able to label 70% of news articles with a category.

Table 5.3 presents the top-10 categories for each dataset. We see that irrespective of the dataset, the most popular categories are Crime, Sport, and Editor's columns. Politics is surprisingly less popular with less than 4% of visited articles per dataset, except in *Campus*, where 11% of visited articles are about politics. In fact, we see interesting differences between campus and ISP users. Campus users are also more interested in Economy and less in the

ISP-Bologna	*ISP-Turin1*	*ISP-Turin2*	*Campus*
Crime 15.01%	Crime 18.03%	Sport 17.90%	Editor's column 14.72%
Sport 13.98%	Sport 17.29%	Crime 14.64%	Sport 14.10%
Editor's column 10.43%	Editor's column 11.43%	Editor's column 12.40%	Economy 13.45%
Entertainment 8.47%	International 8.74%	Entertainment 7.82%	Crime 12.36%
Regional 7.33%	Entertainment 7.87%	Economy 7.11%	Politics 11.00%
Economy 6.42%	Economy 7.3%	People 6.13%	International 6.16%
International 6.41%	Science 4.12%	International 6.03%	Entertainment 4.72%
People 5.33%	Politics 3.81%	Photo Gallery 4.27%	Science 3.68%
Photo Gallery 4.28%	People 3.74%	Politics 3.86%	Photo Gallery 2.40%
Politics 3.38%	Photo Gallery 2.95%	Science 3.62%	Announcements 2.07%

Table 5.3: Top 10 most popular categories per dataset.

category “People”, which appears in the top-10 in the other datasets. The university of the campus is an engineering school and a large fraction of users are students, whereas the ISP traces capture more diverse demographics. In *ISP-Bologna*, we see that approximately 7% of news articles are about Regional news, but Regional news are not present in the top-10 categories in the other datasets.

In the next section, we explore the content of news articles to study whether different datasets (or population of users) share common interests, in other words, whether we see some “locality” in our datasets beyond visiting regional news.

5.4 News Locality

In this section, we characterize the “locality” of visited news articles, or whether people in the same community (in our case, the three different neighbourhoods and the campus) visit news about similar topics.

5.4.1 Method: Text and Keywords Extraction

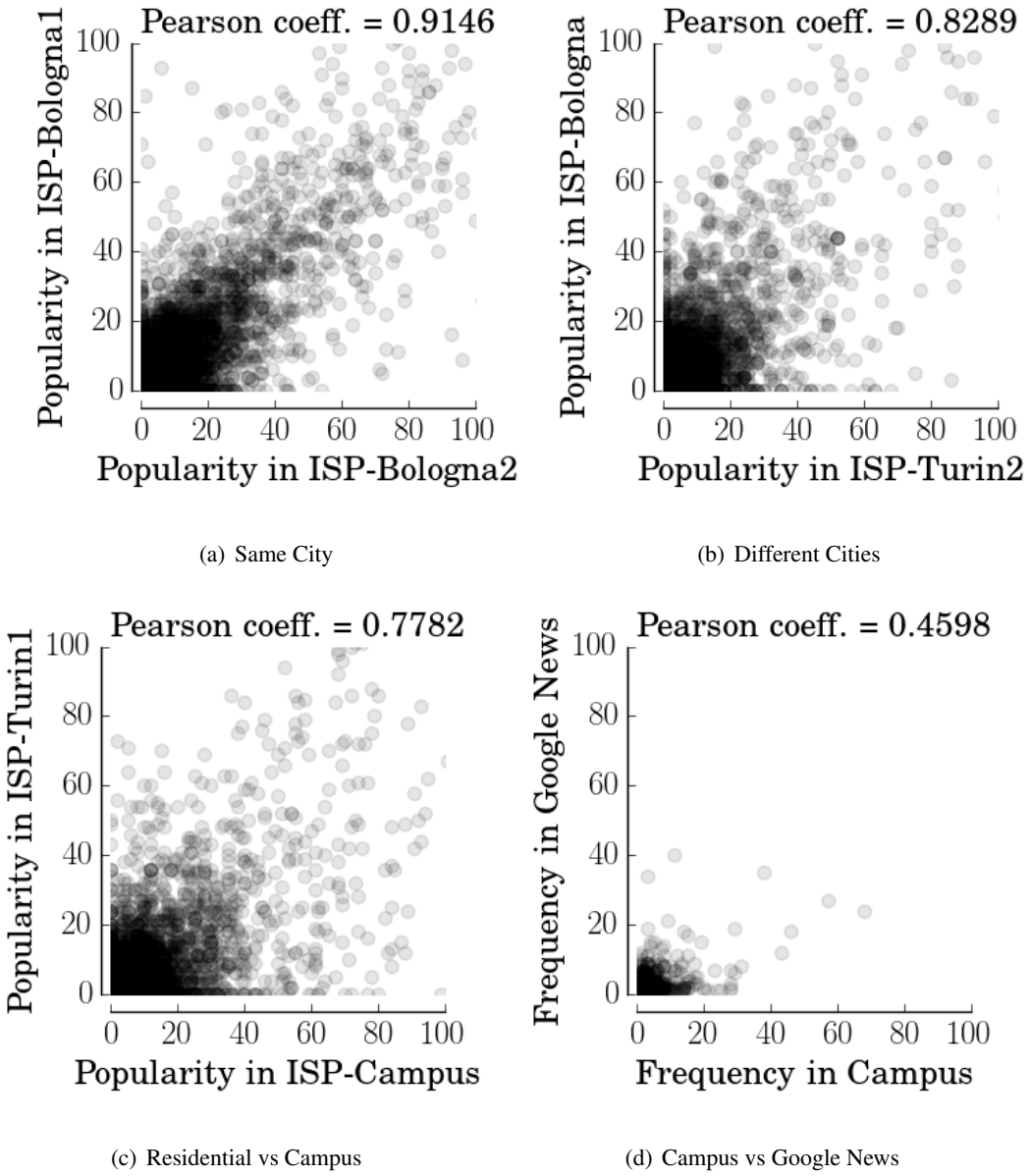
We study the topics of news articles based on a list of keywords we extract for each article. First, we filter out stopwords based on a list of Italian stopwords. Then, we select ten keywords per article using term frequency. To construct the list of Italian stopwords, we rely on existing Italian natural language processing resources. Our analysis of the resulting keywords show that a few keywords appear often and across all four datasets. We inspected these keywords. In many cases, keywords appeared in a really large fraction of articles, for example, Italy or Euro. These words are too generic for our dataset to capture specific topics, so we added most of them to the list of stopwords.

5.4.2 Method: Measuring interest

We infer the interests of the population of users monitored in each community based on the popularity of keywords across all visited articles. Instead of computing popularity over the entire dataset, which is almost 18 months long and may mix periods with different interests, we conduct our analysis in one day of data. We select the same day of data for all communities to compare interests across communities in the same time period. To have a fair comparison across communities, we sample the population and pick each time a subset of 3,000 surfers, which corresponds to the number of surfers we observed during the day of study in our smallest community. We chose the samples randomly and we perform different runs of each comparison and verify that the results are consistent across different runs.

5.4.3 Results

First, we compare the keywords extracted from couples of sample populations of 3000 users each and study the correlations of their popularities. For a qualitative analysis we draw scatter-plots reporting keyword as dots. Each axis corresponds to the popularities of each keyword within the two populations. We use the Pearson coefficient as measure of the correlation. Note that a perfect linear relation implies that the keywords are equally popular in the two analysed populations.



(a) Same City

(b) Different Cities

(c) Residential vs Campus

(d) Campus vs Google News

Figure 5.5: Scatter plots where dots represent keywords extracted from different trace pairs and coordinates correspond to their popularities (Figs. 5.5(a)-5.5(b)-5.5(c)) and their frequency in news webpages (Fig. 5.5(d)).

As a first step we compare the keywords extracted from two random populations of 3000 users from the same location. Fig. 5.5(a) shows the scatterplot for this scenario. We observe that the dots distribute on the main diagonal. The Pearson coefficient is high, 0.91. This result suggests that people within the same city share the same interests in terms of news.

We next compare keywords from two different locations. Fig. 5.5(b) shows the popularities of keywords in two different cities, namely *ISP-Bologna* and *ISP-Turin2*. As expected the dots are more sparse and the Pearson coefficient is lower (0.82) showing high but less shared

interests compared to people in the same city.

To better understand the reason of the high correlation within the same city, we compare the keywords from populations in *ISP-Turin1* and *Campus*. Note that the two traces come from the same city (i.e. same geographical area). We show the result of this comparison in fig. 5.5(c). The Pearson coefficient is lower than the case of two populations in the same city. This means that despite accessing Internet from the same location, the interests of the two communities are different.

These results show the presence of different communities with different interests. To quantify more this phenomenon, we approximate the amount of articles that are proper to each location. To this end, we compare two traces from different cities (namely *ISP-Bologna* and *ISP-Turin2*). We first select only the keywords that are popular in each location (closer to the axis in the scatterplots). In particular, we make sure that their popularity is at least four times higher than that of the opposite location. We then match these keywords to their corresponding news articles. Using this approach, we find that on average, 9% of the articles are of local interest.

5.5 Visited News versus Headlines

We now use the same method as in the previous section but to compare the news articles visited by users with the articles that editors select as headlines. To do this analysis, we compare the top news from one day in *Campus* with the articles promoted by Google News Italy during the same day. Google News selects the most important articles from web journals and promotes them in its webpage. Editors of web journals have means to influence their presence on Google News following its official guidelines[2]. Fig. 5.5(d) shows the result of the analysis.

We observe a weak correlation, suggesting a high difference between the users' interests and the Google News promotions. This can be explained by several facts. First Google News targets the interests of the entire country and those of the users in campus may not reflect them. Second, as the authors of [49] show, there is a divergence between what editors think as headlines and what users consume.

5.6 Visited News Articles per Day

We study the number of visited news articles per day. Fig. 5.6 plots the cumulative distribution of the visited news articles per day for a period of one month in *ISP-Bologna*. It is interesting to see the number of daily visited articles is so steady. The figure shows all visited articles as well as the subset that with Google search and social networks as referral. Other communities and time periods exhibit a similar trend. The number of visited news articles per day by this population of $8,372$ surfers varies from 14,000 to 18,000. The number of news articles with Google or social networks as referral also varies modestly. Intuitively, news articles with Google as referral correspond to events that users heard about and intentionally looked for, so we expected a larger variation depending on big events happening in the world. Our data, however, does not confirm our initial intuition.

This steadiness in the number of visited news articles suggests that either news consumption is limited by the offer of online editors and the events that happen; or that users have a fixed

[2]https://goo.gl/ZWovU9

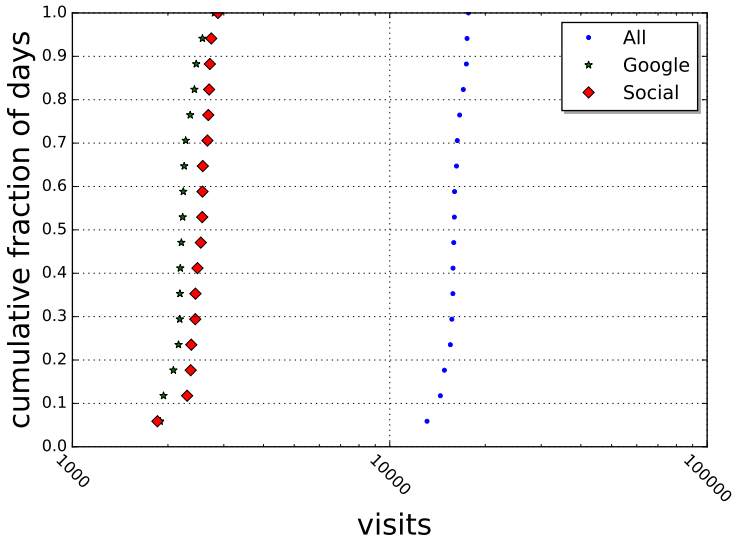


Figure 5.6: Number of visited news articles per day in *ISP-Bologna* (from different referrals)

"daily dose" of news. To shed more light on this observation, we first study the numbers and sizes of stories per day, and then variations of visited news articles among users.

5.7 Number and Sizes of News Stories

We now study news visits across users. Different news articles often cover the same event or *story*. We would like to understand the sizes and numbers of such stories that users visit.

5.7.1 Method: Story Identification

We identify stories on a daily basis by clustering articles. A number of methods exist for this purpose. We experiment with two well-known methods: DBSCAN [127] and identifying connected components on the graph of news articles. The results of the connected components method is more deterministic in our datasets, so we select stories based on connected components and we tune for our datasets as follows.

We first build for each day a graph of the visited news articles, where vertices are visited news articles and edges between articles are weighted with a measure of their similarity. We use the Jaccard similarity coefficient between the top-10-keywords vectors of two articles to measure their similarity. Intuitively, the more keywords in common, the closer the two articles. We must pick a threshold on the similarity to decide that two articles belong to the same story. Then, we remove from the graph any edge with similarity smaller than the threshold. Each connected component of the resulting graph corresponds to a story.

Intuitively, the larger the similarity threshold, the finer the granularity of the story, i.e., the topic of the story is more precise. And the smaller the threshold, the coarser the story. There is no perfect threshold as the definition of a story is subjective. For instance, the ceremony of the 2016 Oscars could be considered as a story, but this event can be a also a source of multiple smaller stories that happen within the ceremony.

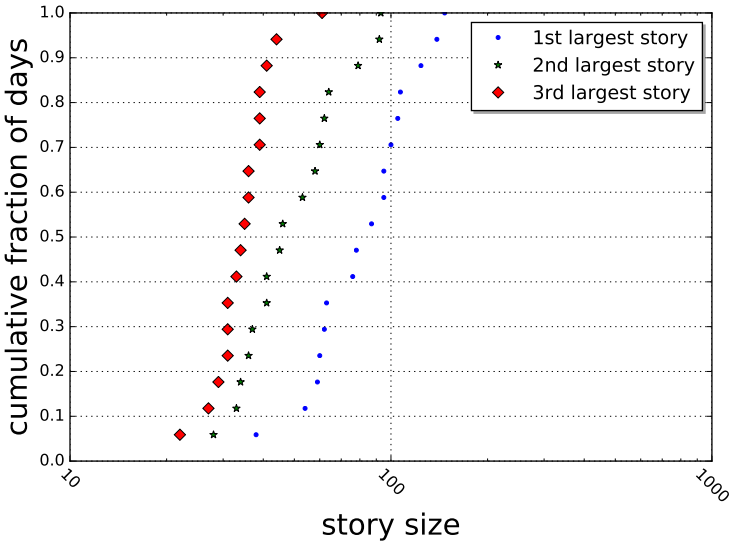


Figure 5.7: Distributions of biggest stories per day

We pick the threshold that allows us to obtain a similar granularity to stories in the Google News site, as this is a popular service with users. To this end, we collect a "ground truth" set of stories from Google News where each news article is assigned by Google to a cluster. We apply, on this trace, the connected components clustering method multiple times varying each time the threshold. We finally pick the threshold that gives us similar granularities of stories compared to Google news. We obtain this result with a threshold of 0.25.

5.7.2 Result

Fig. 5.7 shows the distributions of the size of top three biggest stories per day (in number of visited articles) for the month of April 2015 in *ISP-Bologna*. We see that the biggest story per day varies from 30 to 200 articles, suggesting a variety of events underlying a story. For example, the biggest story in this plot corresponds to an attack in the Milan tribunal where an armed men attacked a judge. This story was all over Italian news with coverage in most online journals leading to 200 articles. The biggest stories of the day, however, are an exception. The third biggest story per day is often two to three times smaller than the biggest. Fig. 5.8 confirms this analysis. It plots the distribution of the number of daily stories, stories with at least two articles, and stories with at least five articles. The figure shows that the number of stories with at least five articles is very small, and that most articles do not fall under any story using our clustering method (i.e., there are many clusters with a single article). We conclude that news are of two types: popular news that will have a varying number of articles depending on the events of the day; and less popular news, which account for the majority of news articles.

5.8 Visited News per User and News Addiction

We check how news consumption varies across users, and how it fluctuates over time. Remember that our dataset can only give us an upper and lower bound on the number of users. More

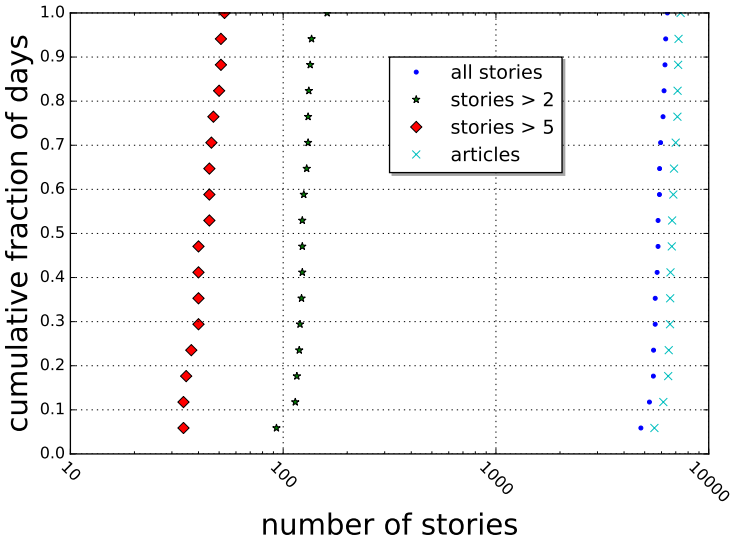


Figure 5.8: Number of visited news articles and stories per day

precisely, the number of households is a lower bound on the number of users, and the number of surfers is an upper bound. Please refer to the definitions in Section "5.1.3".

Fig. 5.9 shows the cumulative distribution of the average number of visited news articles per household and surfer. Both curves present a large variation, which suggest that users have different news consumption patterns. Some users seem to be clearly more addicted to news than others. Indeed, while few households visited more than 400 articles per month, 20% of households visited less than 30 articles on average.

Next, we study if news addiction is a steady habit across time, or more temporary. To this end, we measure the variance of the monthly amount of consumed news by households and surfers. Unfortunately, since user agents can change due to software upgrades, our definition of a surfer is unsuitable for long term analysis. The same device can have multiple user agents during the period of our traces. Additionally, new devices can be added and new IP addresses reallocated. For this reason, we focus only on surfers, and households, which have been present for at least seven consecutive months in our datasets, and study how their news consumption varies over this time period. We show the results for households. The results for surfers that lasted at least seven months exhibit a similar behavior.

Fig. 5.10 shows the average number of visited news articles per household per month together with error bars representing the standard error over the seven months of observation. Households are sorted by decreasing number of average visited news articles per month. The large error bars, in particular for households with higher average, indicate that the average number of visited news articles per month for a given household will vary over the months. This variation reflects vacation periods for instance. Despite this variation, there is a clear trend with some households that often visit news articles and others with orders of magnitude less visited news articles.

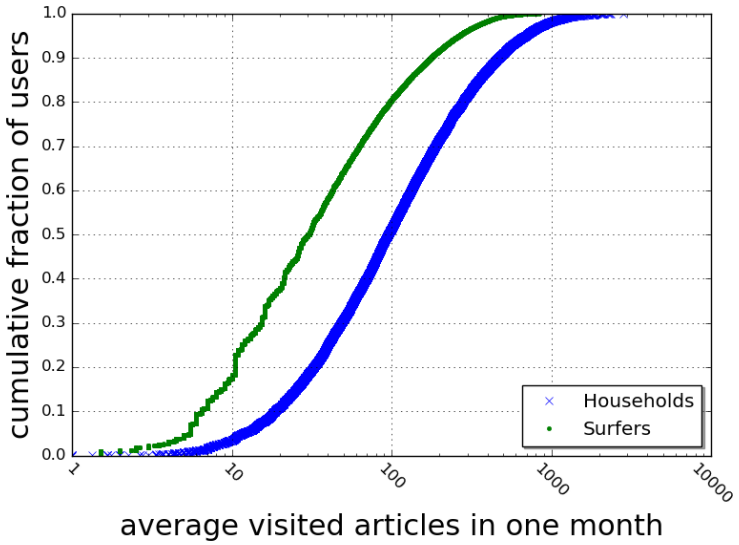


Figure 5.9: **A**verage number of visited articles per month per user

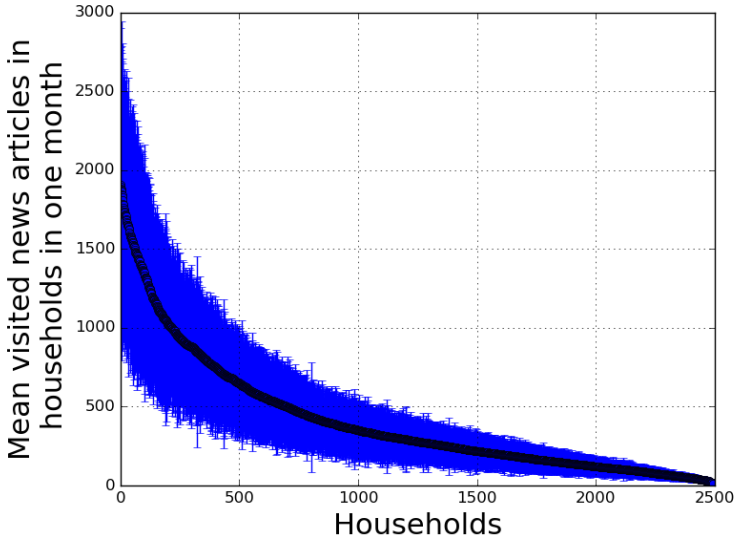


Figure 5.10: Mean consumed news per month per household observed in seven consecutive months, with Standard Errors across the seven months.

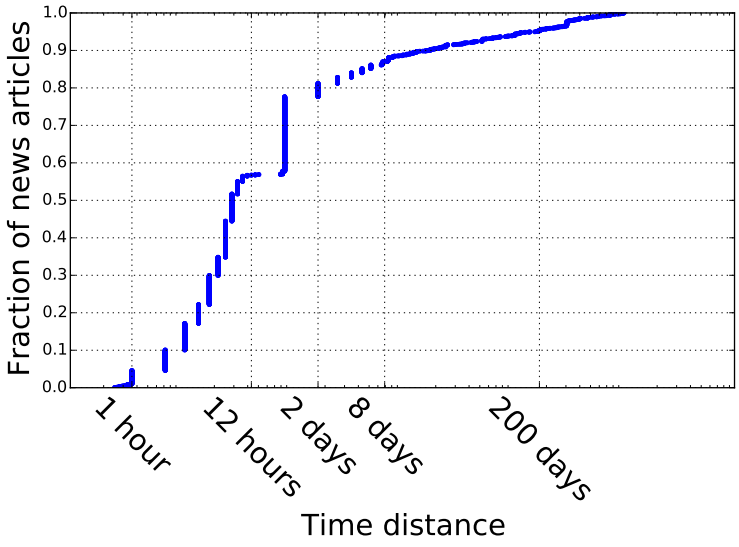


Figure 5.11: Age of visited news articles.

5.9 News Freshness

We study the age of news users in our datasets visit. For this analysis, we collect extra datasets. We continuously run Tstat and extract news visits as soon as they occur. For every news visit, we check if Google News has already indexed it.[3] We use several instances of a headless browser to do a "site:" search on Google Search for each visited news article. If the news article is indexed, we measure since when. This "age" is an information available below each link returned by Google Search. We could run this experiment only for only one day. After that, Google has banned our IP network because of the extensive probing.

Fig. 5.11 shows our results. The figure shows the cumulative distribution function of how long, after the Google indexing, the news visit happened. Note that the steps in the distribution are due to the granularity of the time information provided by Google. The figure shows that for 96% of the time, the visit happens after one hour of publication time. Interestingly, although news aggregators such as Google News tend to promote the most recent news (from few minutes to two hours) on their front pages, what the users really view is not that fresh. More than 44% of news articles keep getting views from one to several days after their publication, and 10% of visited news articles are more than 20 days old.

5.10 Online Journals Clustering

Finally, we would like to understand the different tastes of users in terms of editorial choices. Intuitively, the more two online newspapers are visited by the same users, the closer are these newspapers. As such, the number of co-visits to two newspapers is a good measure of similarity between these newspapers.

We use this similarity metric to cluster online newspapers. We test different clustering

[3] The time at which Google News indexes the news article is a good approximation of its publication time.

Repubblica.it	Corriere.it	Sport	Bologna
repubblica.it	corriere.it	gazzetta.it	ilrestodelcarlino.it
www.huffingtongpost.it	video.corriere.it	corrieredellosposrt.it	lanuovaferrara.geolocal.it
espresso.repubblica.it	milano.corriere.it	tuttosport.com	telestense.it
bologna.repubblica.it	roma.corriere.it	juvenews.com	bolognatoday.it
Finance	**Blogs**	**Strong Opinions**	**General**
ilsole24ore.com	greenme.it	messaggero.it	fanpage.it
borsaitaliana.it	davidemaggio.it	ilsecoloxix.it	ilpost.it
forexinfo.it	ilsussidiario.net	ilgiornale.it	vanityfair.it
finanza-mercati.it	it.blastingnews.com	ilmattino.it	leggo.it

Table 5.4: Biggest nodes in each community

algorithms and realize that the best one for this problem is community detection [128] in the graph of online newspapers. Indeed, in our setting, Newspaper A (e.g. Politics) can be close from Newspaper B (e.g. Sport), Newspaper A can be close from Newspaper C (e.g. People or Fashion), without this implying necessarily that Newspaper C and Newspaper B are close or similar. In the graph of newspapers, each vertex is an online newspaper, and there is a link between two newspapers if at least one user visited both newspapers. We weight links with the number of users who visited both newspapers. The community detection algorithm will intuitively detect Online Newspapers that are more "fully meshed", grouping thus better the interests of users.

We apply the community detection algorithm proposed by [129] to *ISP-Bologna*. This algorithm takes into account the weights of the links when detecting communities. It executes iteratively two steps until its convergence. In the first step, it optimizes local modularities. In the second step, it aggregates the communities found in the previous step. We try different resolutions allowing us to obtain more or less detailed communities. With a resolution of 1, we obtain 4 big communities that we could easily tag as, (1) Repubblica.it, the biggest online editor in Italy, and its derivative newspapers, (2) Corriere, the second biggest, and its derivative newspapers, (3) All Sport magazines, and finally (4) the rest of newspapers.

With a resolution of 0.7, we obtain the 8 communities. We show the nodes that we use to label the clusters in in Tab. 5.4 and the complete clusters in Fig.5.12 to Fig.5.19. We find again the first three clusters of the previous setting. The last cluster is split into what we tagged as "Bologna", "finance", "blogs" and "general".

Interestingly, a remaining cluster contained all newspapers with strong opinions, regardless of their directions. This suggest that people who read newspapers with strong opinions (e.g. overtly supporting one political side) tend also to read newspapers from the opposite opinion, suggesting the absence of selective exposure for such a category of users. Selective exposure is the tendency of people to select information (in this case news sources) that corroborate their previous knowledge or opinions.

5.10.1 Limits

Manually most of the clusters are easy to tag. For example, the *Repubblica.it* cluster contains all the sub-domains of the website www.repubblica.it (e.g. video.repubblica.it) and journals

associated to the same news agency (e.g. www.huffingtonpost.it and www.wired.it). However, with both resolutions (1 and 0.7) we obtain modularity values around than 0.1. Modularity measures the quality of the clusters. It compares the number of edges attached to the nodes in a cluster with the number of edges of the same nodes if the network was created randomly. The value of modularity goes from zero to one and higher values indicate better clusters. Empirically, starting from 0.3 is considered a good value. One possible explanation is that, the modularity measure assumes that each node of the network can connect to any other node. This assumption could not hold in our dataset. For example, reader of the local journal of a small city wouldn't read another journal of another small city. This problem of the modularity metric is known as resolution limit [130]. As a future work, we think to investigate more the validity of this metric for our case.

5.11 Conclusion

In this chapter we used the data that we obtained by means of the techniques described in Chapter 3 to characterize the online news consumption. Understanding news consumption habits is important for social scientists but also for news editors who can use this knowledge to enhance their offer and adapt to user needs. We described our dataset including 80 million news visits to 5.4 million news articles, extracted from a 16 months. We studied the sources of news articles. We showed that the users of the studied communities mainly consume articles directly from web journals. We studied the locality of news. We quantified that 9% of articles talk about local news. We showed that user preferences and editor choices of headlines are not aligned. We showed that users consume old articles. This is counter intuitive as most of news aggregators focus of fresh articles. Finally, we clustered journals using co-visits. Although the modularity values were weak, most of the clusters made sense to a human (e.g. all the sport magazine were grouped together). Interestingly, one of the clusters grouped journals with strong opinions (both left and right political sides) suggesting that people in such categories do not suffer of selective exposure. More work is needed to validate this intuition.

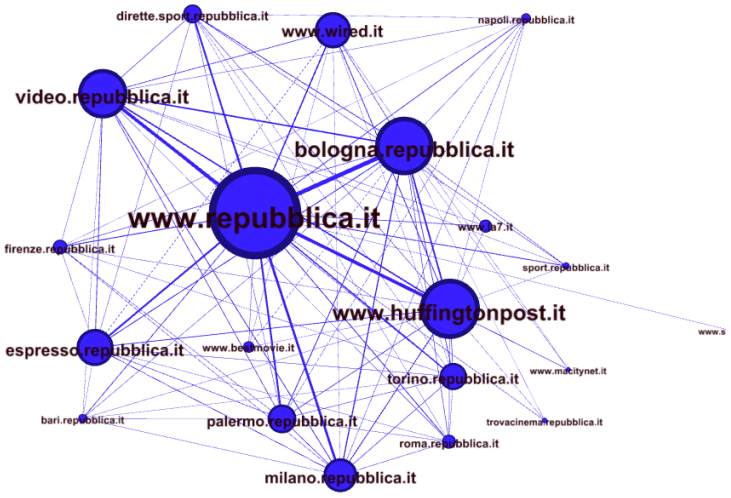


Figure 5.12: Cluster: Repubblica.it

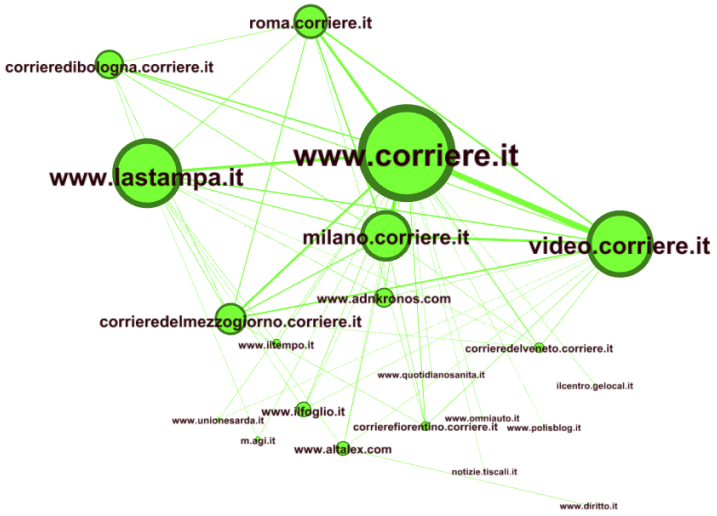


Figure 5.13: Cluster: Corriere.it

Figure 5.14: Cluster: Sport

Figure 5.15: Cluster : Bologna

Figure 5.16: Cluster: Finance

Figure 5.17: Cluster: Blogs

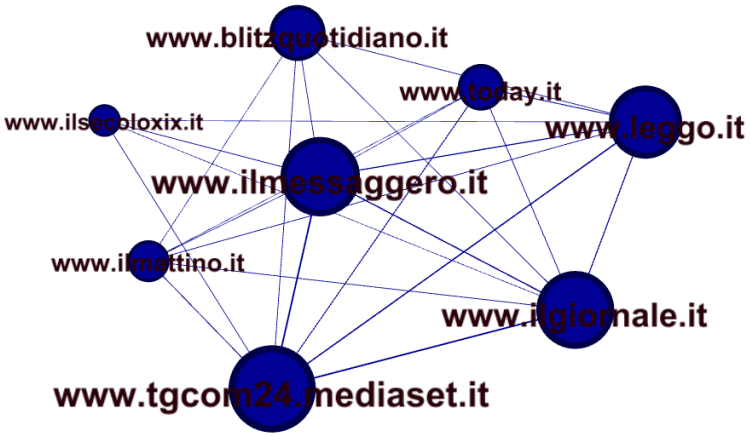


Figure 5.18: Cluster: Strong Opinion

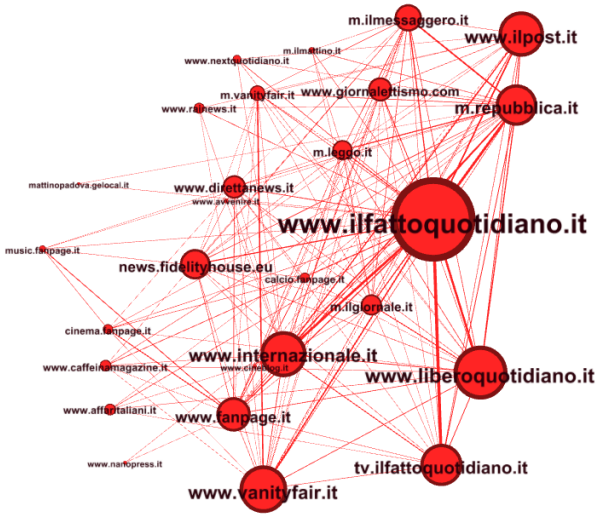


Figure 5.19: Cluster: General

Chapter 6

Conclusion

Several information filtering systems exist to help users finding relevant web content. Examples include content curation tools and recommender systems. One of the biggest challenge of such systems is to obtain a critical mass of users, such that the system has enough input to work. In this thesis, we proposed and evaluated the use of users clicks to augment users' input. In this chapter, we summarize our work and present the open issues and challenges.

6.1 Lessons from Users' Clicks Extraction

In this thesis, our first challenge was to design heuristics to extract users' interests in web content in communities of a place. Our heuristics take as input the log traces of the network. Since log traces contain much more information than users' clicks to web pages, the biggest challenge is to distinguish between what we call user-URLs and browser-URLs. The first are URLs requested by the users. The second are those that the browser requests automatically, with users not being aware of them. To use these heuristics in our system, our following step was to design a scalable algorithm that implements them in an online fashion.

However, this approach faces a serious threat: encryption. Fortunately, there are alternative solutions to explore. For example, one can install a plug-in on the users' browsers. In this case, we don't need heuristics to distinguish between user-URLs and browser-URLs. This solution however, needs more user engagement (e.g. to install the plug-in). Incentives have to be found to get the users install it. Such a setting would however open the way to more innovative uses of our concept. For instance, users can tag themselves as members of communities of Internet (e.g. technology, politics) which would open the way for more than only communities of a place. Another alternative is to use proxies. In the case we want to monitor the network of a campus or an enterprise network, this is an easy solution. Proxies process all the traffic from and directed to the target network. In some cases, proxies can also read encrypted traffic (i.e. private information of the users such as passwords). Thus, the main open issue with them is the privacy. Using proxies, augments the amount of data we can work with. In this case, we might need to design new policies to exclude some traffic from our framework (e.g. HTTPS traffic).

6.2 Lessons from WeBrowse

Once we were able to extract users' clicks online, we used them as input for WeBrowse. WeBrowse is an information filtering system for communities of a place that takes as input user clicks.

To select the URLs that are worth promoting, we designed different blocks that work as filters one on top of the other. Indeed, not all user clicks are worth being promoted. For example, WeBrowse discards websites of banks or search engines. The URLs selected for the promotion appear on a website available to the members of the community. We designed WeBrowse to respect users' privacy and avoid that personal or sensible information is exposed. The security boards of the institutions hosting WeBrowse approved these techniques. Finally, we evaluated WeBrowse by means of surveys directed to its users and passive observation of their behaviour while using it. We obtained encouraging feedback for the future of WeBrowse.

We showed how people in the same community share common interests, what we called a "community of a place effect". There are still some open problems. First, it remains to investigate how to detect more specific community-oriented content compared to content of general interest (e.g. national and international news), and understand whether users would like to see such a split in WeBrowse. To do so, we need techniques that select the content related to a community. Second, other forms of feedback can be used. For example, one can imagine a hybrid (i.e. passive and active) scenario where users can vote up and down WeBrowse's content or submit new links. Third, WeBrowse is mainly a system that deals with users' preferences. Analysing the content that people consume (e.g. the text of the webpages), one can infer a lot about users. We imagine, as a future upgrade of WeBrowse, grouping the promoted items by theme to ease their consumption.

6.3 Lessons from News Characterization

We used our heuristics for clicks extraction also to build a dataset of online news browsing activity. We monitored four different networks in Italy, three residential and one campus for 16 months. We obtained 80 million clicks to online news coming from around 24,000 households. Using this dataset, we made a characterization of online news consumption. We had a privileged point of view. Indeed, all the existing studies focused on only one journal or one platform (e.g. Twitter or Facebook). We presented a number of findings, some of them confirm smaller scale previous work. One of the most promising directions of this characterization was the clustering of journals by groups of readers. We grouped the journals by communities of readers. We investigated the presence of the selective exposure problem, but more can be done in this direction. Especially augmenting the amount of data used for the experiments and testing other clustering techniques. We already tested some other techniques which did not work. First, we considered also the graph of journals where edges are weighted by the number of users who co-visited two journals. We cut the edges having a weight lower than a given threshold and look for the different connected components. This method worked well for the clustering of articles into stories. However, when the nodes of the graph were journals, we always found one single connected component independently from the threshold. This is due to the fact that journals that belong to very different clusters (say finance and people) will keep having a link in common with very popular journals clusters (i.e. sport). Second, we also tested DBSCAN with no success. We believe that more work is needed in this direction to understand the selective

exposure problem.

Indeed, we worked testing other theories to categorize web content according to their topic. We used Latent Dirichlet Allocation, a well know technique for topics discovery in text collections. LDA is a supervised algorithm, meaning that it needs in advance the number of topics to look for in the set of texts. We tried tuning with different values but we didn't obtain any valuable result. More work is required in this direction, trying different techniques maybe more appropriate. We tried also LDA to categorize news articles. We expected this work to be easier because they come in a neat format that allows to extract the real category of the news (as chosen by the editors). However, also this experiment, despite trying different LDA settings, did not give valuable results matching the human characterization.

To summarize, we approached the problem of lack of users' engagement in communities of a place. We designed algorithms to extract users interests passively, meaning without their intention to collaborate. We used the data we obtained to design WeBrowse: a working information filtering system for communities of a place that works passively. We deployed it and our users gave us positive feedback. For this reason, we think that there is room for a system like WeBrowse and in general for passive content curation. Finally, using our clicks extraction algorithms we constructed a dataset of online news browsing. We collected 80 million page views and we conducted a characterization of web news consumption.

Appendix A

Extended Performance Description

In this appendix, we give an extended description of the performance of the filters for detection of user-URLs and candidate-URLs. We give a complete description in Sec. 3.1. In this section we show the results of the experiments that we conduct to determine how to set the parameters of the filters. We present results in tables from A.1 to A.4. . We test the different cases. First, we enable and after, we disable the F-Children-Type filter. We vary the value of the maximum number of parameters that the F-filter allows a URL to be considered user-URL. We also switch the filter off (see the row "OFF" of the tables). In the columns, we report the number of children required by the F-Children to select a user-URL. We vary this value from zero (i.e. F-Children off) to five. In the rows, we repost the maximum number of paramenters allowed. For each combination of number of children and maximum parameter we show the precision and the recall of the heuristics. We use the same definition of Recall and Precision that we use in Sec 3.4. *Recall* is the number of true positives (or URLs correctly labeled as user-URLs) divided by the number of positives (which is the total set of user-URLs). *Precision* is the number of true positives divided by the number of true positives plus false positives (which is the number of URLs our heuristics say are user-URLs).

First we present the result for user-URLs detection in Table A.1 and Table A.2. We notice that in both tables the precision increases in two cases i) when we increase the number of children for a URL required to be a user-URL and ii) when we increase the number of maximum parameters allowed to be user-URL. This behavior is expected indeed increasing this values we impose more strict conditions to identify user-URLs. On the other hand, when precision increases the recall drops drastically.

We show the performance for candidate-URL detection algorithms in Table A.3 and in Table A.4. Also in these table we observe a similar behavior. The recall increases when we increase the number of maximum parameter allowed and decreases when we increase the number of children. However, we notice that in these table the precision is always one. This is because the conditions to be elected candidate-URL are very strict. In practice, in our deployments, when we activate the candidate-URL filters we have as a nice side effect 100% of precision in detecting user URLs.

Table A.1: Performance of the online filters for user-URL detection. F-Children-Type is on. F-Ad is off. We vary the number of parameters in F-parameter and the number of children in F-children

Max parameter	Children	0	1	2	3	4	5
0	Precision	0.854691075515	0.854691075515	0.930535455861	0.955342902711	0.96	0.965583173996
	Recall	0.839325842697	0.839325842697	0.730681818182	0.680681818182	0.630857142857	0.580459770115
1	Precision	0.805143422354	0.805143422354	0.904884318766	0.931914893617	0.935085007728	0.945578231293
	Recall	0.897464167585	0.897464167585	0.784838350056	0.733258928571	0.680539932508	0.629671574179
2	Precision	0.771006463527	0.771006463527	0.892725030826	0.923392612859	0.930827067669	0.940298507463
	Recall	0.919603524229	0.919603524229	0.806236080178	0.752508361204	0.695505617978	0.641402714932
3	Precision	0.745098039216	0.745098039216	0.889434889435	0.91961852861	0.928035982009	0.938741721854
	Recall	0.920704845815	0.920704845815	0.806236080178	0.752508361204	0.695505617978	0.641402714932
4	Precision	0.719387755102	0.719387755102	0.887545344619	0.919463087248	0.929098966027	0.939739413681
	Recall	0.931718061674	0.931718061674	0.817371937639	0.763656633222	0.706741573034	0.652714932127
5	Precision	0.689766317486	0.689766317486	0.883413461538	0.917112299465	0.927835051546	0.938311688312
	Recall	0.932461873638	0.932461873638	0.817575083426	0.763919821826	0.707070707071	0.653107344633
6	Precision	0.688473520249	0.688473520249	0.876869965478	0.908280254777	0.918881118881	0.940902021773
	Recall	0.962962962963	0.962962962963	0.847608453838	0.793986636971	0.737373737374	0.683615819209
7	Precision	0.685802948022	0.685802948022	0.874856486797	0.907124681934	0.917597765363	0.939440993789
	Recall	0.962962962963	0.962962962963	0.847608453838	0.793986636971	0.737373737374	0.683615819209
8	Precision	0.672492401216	0.672492401216	0.87100456621	0.906091370558	0.917712691771	0.939534883721
	Recall	0.964052287582	0.964052287582	0.84872080089	0.795100222717	0.738496071829	0.939534883721
9	Precision	0.668174962293	0.668174962293	0.864253393665	0.89824120603	0.908965517241	0.929555895865
	Recall	0.9651416122	0.9651416122	0.849833147942	0.796213808463	0.739618406285	0.685875706215
10	Precision	0.660700969426	0.660700969426	0.864253393665	0.89824120603	0.908965517241	0.929555895865
	Recall	0.9651416122	0.9651416122	0.849833147942	0.796213808463	0.739618406285	0.685875706215
OFF	Precision	0.636037329505	0.636037329505	0.850779510022	0.885997521685	0.895380434783	0.923896499239
	Recall	0.9651416122	0.9651416122	0.849833147942	0.796213808463	0.739618406285	0.685875706215

Table A.2: Performance of the online filters for user-URL detection. F-Children-Type is off. F-Ad is off. We vary the number of parameters in F-parameter and the number of children in F-children

Max parameter	Children	0	1	2	3	4	5
0	Precision	0.854691075515	0.854691075515	0.905289052891	0.944517833554	0.948369565217	0.954609929078
	Recall	0.839325842697	0.839325842697	0.8297632469	0.806087936866	0.787810383747	0.759593679458
1	Precision	0.805143422354	0.805143422354	0.86847826087	0.912075029308	0.918590522479	0.930025445293
	Recall	0.897464167585	0.897464167585	0.883849557522	0.860619469027	0.837209302326	0.809523809524
2	Precision	0.771006463527	0.771006463527	0.847107438017	0.891741071429	0.90243902439	0.919315403423
	Recall	0.919603524229	0.919603524229	0.906077348066	0.882872928177	0.859513274336	0.83185840708
3	Precision	0.745098039216	0.745098039216	0.835881753313	0.884828349945	0.896193771626	0.914841849148
	Recall	0.920704845815	0.920704845815	0.906077348066	0.882872928177	0.859513274336	0.83185840708
4	Precision	0.719387755102	0.719387755102	0.828343313373	0.878393051031	0.892290249433	0.910394265233
	Recall	0.931718061674	0.931718061674	0.917127071823	0.893922651934	0.870575221239	0.842920353982
5	Precision	0.689766317486	0.689766317486	0.820335636723	0.872844827586	0.889390519187	0.907253269917
	Recall	0.932461873638	0.932461873638	0.917218543046	0.894039735099	0.870718232044	0.843093922652
6	Precision	0.688473520249	0.688473520249	0.815764482431	0.866459627329	0.882034632035	0.909090909091
	Recall	0.962962962963	0.962962962963	0.948123620309	0.923841059603	0.900552486188	0.872928176796
7	Precision	0.685802948022	0.685802948022	0.812677388836	0.862886597938	0.878232758621	0.904925544101
	Recall	0.962962962963	0.962962962963	0.948123620309	0.923841059603	0.900552486188	0.872928176796
8	Precision	0.672492401216	0.672492401216	0.80223380597	0.854230377166	0.869009584665	0.895809739524
	Recall	0.964052287582	0.964052287582	0.949227373068	0.924944812362	0.901657458564	0.874033149171
9	Precision	0.668174962293	0.668174962293	0.797222222222	0.848331648129	0.862724392819	0.888888888889
	Recall	0.9651416122	0.9651416122	0.950331125828	0.926048565121	0.902762430939	0.875138121547
10	Precision	0.660700969426	0.660700969426	0.794280442804	0.847474747475	0.862724392819	0.888888888889
	Recall	0.9651416122	0.9651416122	0.950331125828	0.926048565121	0.902762430939	0.875138121547
OFF	Precision	0.636037329505	0.636037329505	0.771505376344	0.832341269841	0.850156087409	0.882943143813
	Recall	0.9651416122	0.9651416122	0.950331125828	0.926048565121	0.902762430939	0.875138121547

Table A.3: Performance of the online filters for candidate-URL detection. F-Children-Type is off. F-Ad is off. We vary the number of parameters in F-parameter and the number of children in F-children

Max parameter	Children	0	1	2	3	4	5
0	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.120627261761	0.120627261761	0.120627261761	0.120627261761	0.119420989144	0.118214716526
1	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.126658624849	0.126658624849	0.126658624849	0.126658624849	0.125452352232	0.124246079614
2	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.127864897467	0.127864897467	0.127864897467	0.127864897467	0.126658624849	0.125452352232
3	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.127864897467	0.127864897467	0.127864897467	0.127864897467	0.126658624849	0.125452352232
4	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.129071170084	0.129071170084	0.127864897467	0.126658624849
5	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.129071170084	0.129071170084	0.127864897467	0.126658624849
6	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.129071170084	0.129071170084	0.127864897467	0.126658624849
7	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.129071170084	0.129071170084	0.127864897467	0.126658624849
8	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.129071170084	0.129071170084	0.127864897467	0.126658624849
9	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.129071170084	0.129071170084	0.127864897467	0.126658624849
10	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.129071170084	0.129071170084	0.127864897467	0.126658624849
OFF	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.129071170084	0.129071170084	0.127864897467	0.126658624849

Table A.4: Performance of the online filters for candidate-URL detection. F-Children-Type is on. F-Ad is off. We vary the number of parameters in F-parameter and the number of children in F-children

Max parameter	Children	0	1	2	3	4	5
0	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.120627261761	0.120627261761	0.115802171291	0.114595898673	0.107358262967	0.10615199035
1	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.126658624849	0.126658624849	0.121833534379	0.120627261761	0.113389626055	0.112183353438
2	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.127864897467	0.127864897467	0.123039806996	0.121833534379	0.114595898673	0.113389626055
3	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.127864897467	0.127864897467	0.123039806996	0.121833534379	0.114595898673	0.113389626055
4	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.124246079614	0.123039806996	0.115802171291	0.114595898673
5	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.124246079614	0.123039806996	0.115802171291	0.114595898673
6	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.124246079614	0.123039806996	0.115802171291	0.114595898673
7	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.124246079614	0.123039806996	0.115802171291	0.114595898673
8	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.124246079614	0.123039806996	0.115802171291	0.114595898673
9	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.124246079614	0.123039806996	0.115802171291	0.114595898673
10	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.124246079614	0.123039806996	0.115802171291	0.114595898673
OFF	Precision	1.0	1.0	1.0	1.0	1.0	1.0
	Recall	0.129071170084	0.129071170084	0.124246079614	0.123039806996	0.115802171291	0.114595898673

Bibliography

[1] K. C. Laudon et J. P. Laudon, *Management information systems*, vol. 6 (Prentice Hall Upper Saddle River, NJ, 2000). 1

[2] A. J. Sellen, R. Murphy, et K. L. Shaw, "How knowledge workers use the web," dans "ACM SIGCHI," (2002). 1

[3] D. H. Widyantoro, T. R. Ioerger, et J. Yen, "An adaptive algorithm for learning changes in user interests," dans "Proceedings of the eighth international conference on Information and knowledge management," (ACM, 1999), p. 405–412. 1

[4] S. Jung, J. Kim, et J. L. Herlocker, "Applying collaborative filtering for efficient document search," dans "Proceedings of the 2004 IEEE/WIC/ACM International Conference on Web Intelligence," (IEEE Computer Society, 2004), p. 640–643. 1

[5] U. Hanani, B. Shapira, et P. Shoval, "Information filtering: Overview of issues, research and systems," User modeling and user-adapted interaction **11**, 203–259 (2001). 1

[6] N. Hegde, L. Massoulié, et L. Viennot, "Self-organizing flows in social networks," Theoretical Computer Science **584**, 3–18 (2015). 1, 1.1

[7] A. May, A. Chaintreau, N. Korula, et S. Lattanzi, "Filter & follow: How social media foster content curation," dans "ACM SIGMETRICS Performance Evaluation Review," , vol. 42 (ACM, 2014), vol. 42, p. 43–55. 1, 1.1

[8] C. Padoa, D. Schneider, J. M. De Souza, et S. P. J. Medeiros, "Investigating social curation websites: A crowd computing perspective," dans "Computer Supported Cooperative Work in Design (CSCWD), 2015 IEEE 19th International Conference on," (IEEE, 2015), p. 253–258. 1, 1.1

[9] E. Gilbert, "Widespread Underprovision on Reddit," dans "ACM CSCW," (2013). 1

[10] D. Neal, "Digg is dogged by conservative pressure groups," http://goo.gl/M9Plt. [Online; August 2016]. 1

[11] C. Zhong, S. Shah, K. Sundaravadivelan, et N. Sastry, "Sharing the loves: Understanding the how and why of online content curation." dans "ICWSM," (2013). 1.1

[12] A. May, A. Chaintreau, N. Korula, et S. Lattanzi, "Filter & follow: How social media foster content curation," dans "ACM SIGMETRICS Performance Evaluation Review," , vol. 42 (ACM, 2014), vol. 42, p. 43–55. 1.1

[13] M. Sutton, "Content curation tools: The ultimate list." http://www.curata.com/blog/content-curation-tools-the-ultimate-list/. URL : http://www.curata.com/blog/content-curation-tools-the-ultimate-list/, accessed July, 2016. 1.1

[14] "Google trends," http://www.google.com/trends/. 1.1

[15] P. Singer, F. Flöck, C. Meinhart, E. Zeitfogel, et M. Strohmaier, "Evolution of Reddit: from the front page of the Internet to a self-referential community?" dans "ACM WWW," (2014). 1.1, 1.2, 2.1.4

[16] E. Gilbert, "Widespread underprovision on reddit," dans "Conference on Computer supported cooperative work," (ACM, 2013). 1.1, 1.2, 2.1.4

[17] S. Dale, "Content curation the future of relevance," Business Information Review **31**, 199–205 (2014). 1.1

[18] S. Rosenbaum, "Curation nation: How to win in a world where consumers are creators," (2011). 1.1

[19] A. Wolff et P. Mulholland, "Curation, curation, curation," dans "Proceedings of the 3rd Narrative and Hypertext Workshop," (ACM, 2013), p. 1. 1.1

[20] G. Adomavicius et A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," IEEE transactions on knowledge and data engineering **17**, 734–749 (2005). 1.1

[21] J. A. Konstan, "Introduction to recommender systems," dans "Proceedings of the 2008 ACM SIGMOD international conference on Management of data," (ACM, 2008), p. 1373–1374. 1.1

[22] J. S. Breese, D. Heckerman, et C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," dans "Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence," (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998), UAI'98, p. 43–52. URL : http://dl.acm.org/citation.cfm?id=2074094.2074100. 1.1

[23] B. Sarwar, G. Karypis, J. Konstan, et J. Riedl, "Item-based collaborative filtering recommendation algorithms," dans "Proceedings of the 10th International Conference on World Wide Web," (ACM, New York, NY, USA, 2001), WWW '01, p. 285–295. URL : http://doi.acm.org/10.1145/371920.372071. 1.1

[24] S. M. Kywe, E.-P. Lim, et F. Zhu, "A survey of recommender systems in twitter," dans "Proceedings of the 4th International Conference on Social Informatics," (Springer-Verlag, Berlin, Heidelberg, 2012), SocInfo'12, p. 420–433. URL : http://dx.doi.org/10.1007/978-3-642-35386-4_31. 1.1

[25] J. Bennett, C. Elkan, B. Liu, P. Smyth, et D. Tikk, "Kdd cup and workshop 2007," SIGKDD Explor. Newsl. **9**, 51–52 (2007). URL : http://doi.acm.org/10.1145/1345448.1345459. 1.1, 2.1.6

[26] S. M. McNee, J. Riedl, et J. A. Konstan, "Being accurate is not enough: how accuracy metrics have hurt recommender systems," dans "CHI'06 extended abstracts on Human factors in computing systems," (ACM, 2006), p. 1097–1101. 1.1

[27] A. I. Schein, A. Popescul, L. H. Ungar, et D. M. Pennock, "Methods and metrics for cold-start recommendations," dans "Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval," (ACM, 2002), p. 253–260. 1.2

[28] Y. Hu, Y. Koren, et C. Volinsky, "Collaborative filtering for implicit feedback datasets," dans "2008 Eighth IEEE International Conference on Data Mining," (Ieee, 2008), p. 263–272. 1.2

[29] S. Sahebi et W. W. Cohen, "Community-based recommendations: a solution to the cold start problem," dans "Workshop on recommender systems and the social web, RSWEB," (2011). 1.2, 2.1.3

[30] L. Quijano-Sanchez, J. A. Recio-Garcia, B. Diaz-Agudo, et G. Jimenez-Diaz, "Social factors in group recommender systems," ACM TIST (2013). 1.2, 1.3, 2.1.3

[31] A. Sharma et D. Cosley, "Do social explanations work?: studying and modeling the effects of social explanations in recommender systems," dans "ACM WWW," (2013). 1.2, 1.3, 2.1.3

[32] M. S. Amin, B. Yan, S. Sriram, A. Bhasin, et C. Posse, "Social referral: leveraging network connections to deliver recommendations," dans "ACM RecSys," (2012). 1.2, 1.3, 2.1.3

[33] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, et S. Ofek-Koifman, "Personalized recommendation of social software items based on social relations," dans "ACM RecSys," (2009). 1.2, 1.3, 2.1.3

[34] X. Yang, Y. Guo, Y. Liu, et H. Steck, "A survey of collaborative filtering based social recommender systems," Computer Communications **41**, 1–10 (2014). 1.2, 1.3

[35] A. P. McAfee, "Enterprise 2.0: The dawn of emergent collaboration," MIT Sloan management review **47** (2006). 1.2

[36] A. Jackson, J. Yates, et W. Orlikowski, "Corporate blogging: Building community through persistent digital talk," dans "System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on," (IEEE, 2007), p. 80–80. 1.2, 2.1.5

[37] J. DiMicco, D. R. Millen, W. Geyer, C. Dugan, B. Brownholtz, et M. Muller, "Motivations for social networking at work," dans "ACM CSCW," (2008). 1.2, 2.1.5

[38] F. Hideyuki, "Innovationcafe an In-house Social Network Service (SNS) Used in NEC," NEC Technical Journal **2** (2007). 1.2, 2.1.5

[39] Z. Du, X. Fu, C. Zhao, et T. Liu, "University campus social network system for knowledge sharing," dans "IEEE ICSAI," (2012). 1.2, 2.1.5

[40] S. T. Kim, C. K. Lee, et T. Hwang, "Investigating the influence of employee blogging on it workers' organisational citizenship behaviour," International Journal of Information Technology and Management (2008). 1.2, 2.1.5

[41] H. Hasan et C. C. Pfaff, "The wiki: an environment to revolutionise employees' interaction with corporate knowledge," dans "ACM OZCHI," (2006). 1.2, 2.1.5

[42] D. R. Millen, J. Feinberg, et B. Kerr, "Dogear: Social bookmarking in the enterprise," dans "Proceedings of the SIGCHI conference on Human Factors in computing systems," (ACM, 2006), p. 111–120. 1.2

[43] B. D. Friedman, M. J. Burns, et J. Cao, "Enterprise social networking data analytics within alcatel-lucent," Bell Labs Technical Journal **18** (2014). 1.2, 2.1.5

[44] M. Butkiewicz, H. Madhyastha, et V. Sekar, "Characterizing web page complexity and its impact," IEEE/ACM ToN **PP**, 1–1 (2013). 1.3

[45] "Reddit statistics," URL : https://www.reddit.com/about. 1.3

[46] M. Stephens, *A History of News* (Oxfort University Press, Bloomington, IN, USA, 2006). 1.4.3

[47] S. Kourogi, H. Fujishiro, A. Kimura, et H. Nishikawa, "Identifying attractive news headlines for social media," dans "Proceedings of the 24th ACM International on Conference on Information and Knowledge Management," (ACM, 2015), p. 1859–1862. 1.4.3, 2.3

[48] A. Olteanu, C. Castillo, N. Diakopoulos, et K. Aberer, "Comparing events coverage in online news and social media: The case of climate change," dans "Proceedings of the Ninth International AAAI Conference on Web and Social Media," (2015), EPFL-CONF-211214. 1.4.3

[49] P. J. Boczkowski, "The divergent online news preferences of journalists and readers," Communications of the ACM **53**, 24–25 (2010). 1.4.3, 2.3, 5.5

[50] Z. Dezsö, E. Almaas, A. Lukács, B. Rácz, I. Szakadát, et A.-L. Barabási, "Dynamics of information access on the web," Physical Review E **73**, 066132 (2006). 1.4.3, 2.3

[51] N. B. Ellison *et al.*, "Social network sites: Definition, history, and scholarship," Journal of Computer-Mediated Communication **13**, 210–230 (2007). 2.1.1

[52] M. G. Noll et C. Meinel, "Web search personalization via social bookmarking and tagging," dans "The semantic web," (Springer, 2007), p. 367–380. 2.1.1

[53] H. Halpin, V. Robu, et H. Shepherd, "The complex dynamics of collaborative tagging," dans "Proceedings of the 16th international conference on World Wide Web," (ACM, 2007), p. 211–220. 2.1.1

[54] S. A. Golder et B. A. Huberman, "Usage patterns of collaborative tagging systems," Journal of information science **32**, 198–208 (2006). 2.1.1

[55] G. D. Linden, J. A. Jacobi, et E. A. Benson, "Collaborative recommendations using item-to-item similarity mappings," (2001). US Patent 6,266,649. 2.1.1

[56] J. Liu, P. Dolan, et E. R. Pedersen, "Personalized news recommendation based on click behavior," dans "ACM IUI," . 2.1.2, 2.1.5

[57] A. I. Schein, A. Popescul, L. H. Ungar, et D. M. Pennock, "Methods and metrics for cold-start recommendations," dans "Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval," (ACM, 2002), p. 253–260. 2.1.3

[58] X. N. Lam, T. Vu, T. D. Le, et A. D. Duong, "Addressing cold-start problem in recommendation systems," dans "Proceedings of the 2nd international conference on Ubiquitous information management and communication," (ACM, 2008), p. 208–211. 2.1.3

[59] M. J. Pazzani, "A framework for collaborative, content-based and demographic filtering," Artificial Intelligence Review **13**, 393–408 (1999). 2.1.3

[60] Z. Huang, H. Chen, et D. Zeng, "Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering," ACM Transactions on Information Systems (TOIS) **22**, 116–142 (2004). 2.1.3

[61] D. Billsus et M. J. Pazzani, "Learning collaborative information filters." dans "Icml," , vol. 98 (1998), vol. 98, p. 46–54. 2.1.3

[62] B. Sarwar, G. Karypis, J. Konstan, et J. Riedl, "Application of dimensionality reduction in recommender system-a case study," Rapp. techn., DTIC Document (2000). 2.1.3

[63] E. Bakshy, I. Rosenn, C. Marlow, et L. Adamic, "The role of social networks in information diffusion," dans "Proceedings of the 21st international conference on World Wide Web," (ACM, 2012), p. 519–528. 2.1.4

[64] H. Kwak, C. Lee, H. Park, et S. Moon, "What is Twitter, a social network or a news media?" dans "ACM WWW," (2010). 2.1.4

[65] E. Bakshy, J. M. Hofman, W. A. Mason, et D. J. Watts, "Everyone's an influencer: quantifying influence on twitter," dans "Proceedings of the fourth ACM international conference on Web search and data mining," (ACM, 2011), p. 65–74. 2.1.4

[66] E. Gilbert, "Widespread underprovision on reddit," dans "Proceedings of the 2013 conference on Computer supported cooperative work," (ACM, 2013), p. 803–808. 2.1.4

[67] G. Adomavicius et A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," IEEE transactions on knowledge and data engineering **17**, 734–749 (2005). 2.1.4

[68] X. Su et T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," Advances in artificial intelligence **2009**, 4 (2009). 2.1.4

[69] H. V. Madhyastha et M. Maiya, "Towards comprehensive social sharing of recommendations: augmenting push with pull," dans "ACM HotNets," (2013). 2.1.4

[70] A. P. McAfee, "Emergent collaboration," (2006). 2.1.5

[71] J. Liu et L. Birnbaum, “Localsavvy: aggregating local points of view about news issues,” dans “ACM LOCWEB,” (2008). 2.1.5

[72] D. Zhao et M. B. Rosson, “How and why people twitter: the role that micro-blogging plays in informal communication at work,” dans “Proceedings of the ACM 2009 international conference on Supporting group work,” (ACM, 2009), p. 243–252. 2.1.5

[73] C. Steinfield, J. M. DiMicco, N. B. Ellison, et C. Lampe, “Bowling online: social networking and social capital within the organization,” dans “Proceedings of the fourth international conference on Communities and technologies,” (ACM, 2009), p. 245–254. 2.1.5

[74] A. Acquisti et R. Gross, “Imagined communities: Awareness, information sharing, and privacy on the facebook,” dans “International workshop on privacy enhancing technologies,” (Springer, 2006), p. 36–58. 2.1.5

[75] W. Sherchan, S. Nepal, et C. Paris, “A survey of trust in social networks,” ACM Computing Surveys (CSUR) **45**, 47 (2013). 2.1.5

[76] F. Stutzman, R. Gross, et A. Acquisti, “Silent listeners: The evolution of privacy and disclosure on facebook,” Journal of privacy and confidentiality **4**, 2 (2013). 2.1.5

[77] B. Krishnamurthy, “Privacy and online social networks: can colorless green ideas sleep furiously?” IEEE Security & Privacy **11**, 14–20 (2013). 2.1.5

[78] G. Shani et A. Gunawardana, “Evaluating recommendation systems,” dans “Recommender systems handbook,” (Springer, 2011), p. 257–297. 2.1.6

[79] H. Steck, “Item popularity and recommendation accuracy,” dans “Proceedings of the fifth ACM conference on Recommender systems,” (ACM, 2011), p. 125–132. 2.1.6

[80] S. M. McNee, J. Riedl, et J. A. Konstan, “Being accurate is not enough: how accuracy metrics have hurt recommender systems,” dans “CHI’06 extended abstracts on Human factors in computing systems,” (ACM, 2006), p. 1097–1101. 2.1.6

[81] C.-N. Ziegler, S. M. McNee, J. A. Konstan, et G. Lausen, “Improving recommendation lists through topic diversification,” dans “Proceedings of the 14th international conference on World Wide Web,” (ACM, 2005), p. 22–32. 2.1.6

[82] M. Ge, C. Delgado-Battenfeld, et D. Jannach, “Beyond accuracy: evaluating recommender systems by coverage and serendipity,” dans “Proceedings of the fourth ACM conference on Recommender systems,” (ACM, 2010), p. 257–260. 2.1.6

[83] B. P. Knijnenburg, M. C. Willemsen, Z. Gantner, H. Soncu, et C. Newell, “Explaining the user experience of recommender systems,” User Modeling and User-Adapted Interaction **22**, 441–504 (2012). 2.1.6

[84] M. Jia, S. Ye, X. Li, et J. Dickerson, “Web site recommendation using http traffic,” dans “IEEE ICDM,” (2007). 2.2

[85] P. Barford et M. Crovella., "Generating representative web workloads for network and server performance evaluation," dans "ACM SIGMETRICS," (1998). 2.2

[86] H.-K. Choi et J. O. Limb, "A behavioral model of web traffic," dans "IEEE ICNP," (1999). 2.2, 3.1

[87] S. Ihm et V. S. Pai, "Towards Understanding Modern Web Traffic," dans "ACM IMC," (2011). 2.2, 3.3

[88] G. Xie, M. Iliofotou, T. Karagiannis, M. Faloutsos, et Y. Jin, "ReSurf: Reconstructing web-surfing activity from network traffic," dans "IFIP Networking," (2013). 2.2, 3.3, 3.4

[89] F. Schneider, B. Ager, G. Maier, A. Feldmann, et S. Uhlig, "Pitfalls in http traffic measurements and analysis," dans "PAM," (2012). 2.2

[90] M. Eirinaki et M. Vazirgiannis, "Web mining for web personalization," ACM Transactions on Internet Technology (TOIT) **3**, 1–27 (2003). 2.2

[91] Q. Yang, H. H. Zhang, et T. Li, "Mining web logs for prediction models in www caching and prefetching," dans "Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining," (ACM, 2001), p. 473–478. 2.2

[92] C. Neasbitt, R. Perdisci, K. Li, et T. Nelms, "Clickminer: Towards forensic reconstruction of user-browser interactions from network traces," dans "Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security," (ACM, 2014), p. 1244–1255. 2.2

[93] C. Neasbitt, B. Li, R. Perdisci, L. Lu, K. Singh, et K. Li, "Webcapsule: Towards a lightweight forensic engine for web browsers," dans "Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security," (ACM, 2015), p. 133–145. 2.2

[94] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, et P. Steenkiste, "The cost of the s in https," dans "Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies," (ACM, 2014), p. 133–140. 2.2

[95] T. Nelms, R. Perdisci, M. Antonakakis, et M. Ahamad, "Webwitness: Investigating, categorizing, and mitigating malware paths," dans "24th USENIX Security Symposium (USENIX Security 15)," (USENIX Association, Washington, D.C., 2015), p. 1025–1040. URL : https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/nelms. 2.2

[96] L. Vassio, I. Drago, et M. Mellia, "Detecting user actions from http traces: Toward an automatic approach," dans "Wireless Communications and Mobile Computing Conference (IWCMC), 2016 International," (IEEE, 2016), p. 50–55. 2.2

[97] J. Dimmick, Y. Chen, et Z. Li, "Competition between the internet and traditional news media: The gratification-opportunities niche dimension," The Journal of Media Economics **17**, 19–33 (2004). 2.3

[98] D. Ahlers, "News consumption and the new electronic media," The Harvard International Journal of Press/Politics **11**, 29–52 (2006). 2.3

[99] D. Lagun et M. Lalmas, "Understanding user attention and engagement in online news reading," dans "Proceedings of the Ninth ACM International Conference on Web Search and Data Mining," (ACM, 2016), WSDM '16, p. 113–122. 2.3

[100] J. Lehmann, C. Castillo, M. Lalmas, et E. Zuckerman, "Finding news curators in twitter," dans "Proceedings of the 22nd International Conference on World Wide Web," (ACM, 2013), p. 863–870. 2.3

[101] C. Castillo, M. El-Haddad, J. Pfeffer, et M. Stempeck, "Characterizing the life cycle of online news stories using social media reactions," dans "Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing," (ACM, 2014), p. 211–223. 2.3, 5.2

[102] W. X. Zhao, J. Jiang, J. Weng, J. He, E.-P. Lim, H. Yan, et X. Li, "Comparing twitter and traditional media using topic models," dans "Advances in Information Retrieval," (Springer, 2011), p. 338–349. 2.3

[103] M. B. Zafar, P. Bhattacharya, N. Ganguly, S. Ghosh, I. Shibpur, et I. K. P. Gummadi, "On the wisdom of experts vs. crowds: Discovering trustworthy topical news in microblogs," (2016). 2.3

[104] P. Bhattacharya, S. Ghosh, J. Kulshrestha, M. Mondal, M. B. Zafar, N. Ganguly, et K. P. Gummadi, "Deep twitter diving: Exploring topical groups in microblogs at scale," dans "Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing," (ACM, 2014), p. 197–210. 2.3

[105] H. Kwak, C. Lee, H. Park, et S. Moon, "What is twitter, a social network or a news media?" dans "Proceedings of the 19th international conference on World wide web," (ACM, 2010), p. 591–600. 2.3

[106] J. Lehmann, C. Castillo, M. Lalmas, et E. Zuckerman, "Transient news crowds in social media." dans "ICWSM," (2013). 2.3

[107] D. Saez-Trumper, C. Castillo, et M. Lalmas, "Social media news communities: gatekeeping, coverage, and statement bias," dans "Proceedings of the 22nd ACM international conference on Conference on information & knowledge management," (ACM, 2013), p. 1679–1684. 2.3

[108] Y. Wang et G. Mark, "Trust in online news: comparing social media and official media use by chinese citizens," dans "Proceedings of the 2013 conference on Computer supported cooperative work," (ACM, 2013), p. 599–610. 2.3

[109] J. S. Morgan, C. Lampe, et M. Z. Shafiq, "Is news sharing on twitter ideologically biased?" dans "Proceedings of the 2013 conference on Computer supported cooperative work," (ACM, 2013), p. 887–896. 2.3

[110] A. M. Lee et H. I. Chyi, "When newsworthy is not noteworthy: Examining the value of news from the audience's perspective," Journalism Studies **15**, 807–820 (2014). 2.3

[111] D. Frey, "Recent research on selective exposure to information," Advances in experimental social psychology **19**, 41–80 (1986). 2.3

[112] S. A. Munson et P. Resnick, "Presenting diverse political opinions: how and how much," dans "Proceedings of the SIGCHI conference on human factors in computing systems," (ACM, 2010), p. 1457–1466. 2.3

[113] E. Bakshy, S. Messing, et L. A. Adamic, "Exposure to ideologically diverse news and opinion on facebook," Science **348**, 1130–1132 (2015). 2.3

[114] S. A. Munson, S. Y. Lee, et P. Resnick, "Encouraging reading of diverse political viewpoints with a browser widget." dans "ICWSM," (2013). 2.3

[115] S. Park, S. Kang, S. Chung, et J. Song, "Newscube: delivering multiple aspects of news to mitigate media bias," dans "Proceedings of the SIGCHI Conference on Human Factors in Computing Systems," (ACM, 2009), p. 443–452. 2.3

[116] "Adblock Plus," http://easylist.adblockplus.org/. [Online; July 2013]. 3.1

[117] A. Finamore, M. Mellia, M. Meo, M. M. Munafò, et D. Rossi, "Experiences of Internet traffic monitoring with Tstat," IEEE Network (2011). 3.3, 4.1, 5.1.1

[118] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munafò, K. Papagiannaki, et P. Steenkiste, "The Cost of the "S" in HTTPS," dans "ACM CoNext," (2014). URL : http://doi.acm.org/10.1145/2674005.2674991. 1

[119] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, R. Rockell, D. Moll, T. Seely, et C. Diot, "Packet-level traffic measurements from the Sprint IP backbone," dans "IEEE Network Magazine," , vol. 17 (2003), vol. 17. 4.1

[120] Y. Sun, A. K. C. Wong, et M. S. Kamel, "Classification of imbalanced data: a review," International Journal of Pattern Recognition and Artificial Intelligence (2009). 4.2.2

[121] "Reddit's source code"," https://github.com/iangreenleaf/reddit. 4.2.3

[122] "Survey Response Rates," http://goo.gl/NiW471. 7

[123] "Google Language Detection," http://code.google.com/p/language-detection/. 4.4.4

[124] Google, "Google news," https://news.google.com/. URL : https://news.google.com/, accessed May 6, 2015. 4.4.5

[125] M. Manson, "Number of facebook users in italy," (online May 2016). http://goo.gl/14gFl1. 5.2

[126] Statista, "Number of twitter users in italy," (online May 2016). http://goo.gl/liXla2. 5.2

[127] M. Ester, H.-P. Kriegel, J. Sander, et X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." . 5.7.1

[128] S. Fortunato, "Community detection in graphs," Physics reports **486**, 75–174 (2010). 5.10

[129] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, et E. Lefebvre, "Fast unfolding of communities in large networks," Journal of statistical mechanics: theory and experiment **2008**, P10008 (2008). 5.10

[130] S. Fortunato et M. Barthelemy, "Resolution limit in community detection," Proceedings of the National Academy of Sciences **104**, 36–41 (2007). 5.10.1

[131] F. Ricci, L. Rokach, et B. Shapira, *Introduction to recommender systems handbook* (Springer, 2011).

[132] P. Resnick et H. R. Varian, "Recommender systems," Communications of the ACM **40**, 56–58 (1997).

[133] T. Mahmood et F. Ricci, "Improving recommender systems with adaptive conversational strategies," dans "Proceedings of the 20th ACM conference on Hypertext and hypermedia," (ACM, 2009), p. 73–82.

[134] R. Burke, "Hybrid web recommender systems," dans "The adaptive web," (Springer, 2007), p. 377–408.

[135] M. Bilenko et R. W. White, "Mining the search trails of surfing crowds: identifying relevant websites from user activity," dans "Proceedings of the 17th international conference on World Wide Web," (ACM, New York, NY, USA, 2008), WWW '08, p. 51–60. URL : http://doi.acm.org/10.1145/1367497.1367505.

[136] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, et G. Gay, "Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search," ACM Transactions on Information Systems (TOIS) **25**, 7 (2007).