Demo: Ping - A customizable, open-source information filtering system for textual data

Thanasis Chantzios, Lefteris Zervakis, Spiros Skiadopoulos, Christos Tryfonopoulos
University of the Peloponnese, Tripolis, Greece
{tchantzios,zervakis,spiros,trifon}@uop.gr

ABSTRACT

Information filtering has emerged as a prominent paradigm of timely information delivery; in such a setup, users submit profiles that express their information needs to a server which is responsible for notifying them when information that matches their profiles becomes available. Traditionally, information filtering research has focused mainly on providing algorithmic solutions that enhance the efficiency and effectiveness of the filtering process, and has largely neglected the development of tools/systems that showcase the usefulness of the paradigm. In this work, we put forward Ping, a fully-functional content-based information filtering system aiming *(i)* to showcase the realisability of information filtering and *(ii)* to explore and test the suitability of the existing technological arsenal for information filtering tasks. The proposed system is entirely based upon open-source tools and components, is customisable enough to be adapted for different textual information filtering tasks, and puts emphasis in user profile expressivity, intuitive UIs, and timely information delivery. To assess the customisability of Ping, we deployed it in two distinct application scenarios, and assessed its performance under both scenarios.

CCS CONCEPTS

• **Information systems** → **Information systems applications**; **Document filtering**; • **Software and its engineering** → *Publish-subscribe / event-based architectures*;

KEYWORDS

Information filtering, user profiles and alert services, document filtering, interactive systems and tools

ACM Reference Format:
Thanasis Chantzios, Lefteris Zervakis, Spiros Skiadopoulos, Christos Tryfonopoulos. 2019. Demo: Ping - A customizable, open-source information filtering system for textual data. In *DEBS '19: The 13th ACM International Conference on Distributed and Event-based Systems (DEBS '19), June 24–28, 2019, Darmstadt, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3328905.3332512

1 INTRODUCTION

In the modern digital era, the creation and availability of new information has increased exponentially. A plethora of information sources, such as news delivery sites, weather reporting services, and digital libraries, constantly make new content available at an overwhelming pace. To assist users in coping with the vast amount of newly generated information and the cognitive overload associated with it, the *Information Filtering* (IF) paradigm was introduced. In an IF scenario, users are asked to (implicitly or explicitly) express their information needs through appropriate interfaces, tools and languages and submit *profiles* (or continuous queries) to a system or service. In this way, users create subscriptions that are continuously matched (by the system/service) against newly published content, and generate notifications whenever new content that matches users' information needs is published.

Over the past decades, IF research has mainly focused on providing efficient and effective algorithmic solutions [1, 3, 5, 7, 9–11, 13] that worked well in the controlled research environment, but were never actually used "in the battlefield" as components of a larger IF system. This lack of IF tools that would integrate promising solutions and allow developers to use them for building added-value IF services over textual sources or streams, resulted in the lack of prominent IF systems that would act as demonstrators for the usefulness of the IF paradigm. Thus, currently, the only prominent demonstrator of the potential of IF is Google Alerts [6], a proprietary closed-source service built upon the Google ecosystem. Although many users nowadays (mis-)use Google Alerts to monitor the web for marketing (e.g., brand mentions), social listening (e.g., comment follow-up), or even citation counting purposes (e.g., in the context of GoogleScholar), there is clearly a need for an extensible, customisable open-source IF system that could be modified to fit domain-specific IF tasks. Such a system, would act in favour of IF research by *(i)* showcasing the usefulness of the IF paradigm to end-users, *(ii)* providing a basis for developers to build added-value IF services in a number of different domains, *(iii)* testing the current technological arsenal in IF, and *(iv)* providing data (that are in scarcity in the IF domain) regarding system usage or user profiling.

In this work, we present a full-fledged, customisable, open-source IF system, coined Ping, that makes use of state-of-the-art tools and web technologies; we concentrate on providing an operational system that is designed and implemented on IF-specific requirements. To this end, the presented system is equipped with profile administration (e.g., creation, modification, submission), publication management capabilities (e.g., collection, filtering), different content delivery options (e.g., email or on-site notifications), (interval- or batch-based) monitoring of different types of textual data, and an intuitive user interface. The front-end of the system is built upon modern Internet technologies, while the back-end relies on the well-established Apache Solr[1] platform. Ping is designed with flexibility and customisability in mind; developers may use it to easily create textual IF engines for different domains, parameterise and

[1] http://lucene.apache.org/solr/

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
DEBS '19, June 24–28, 2019, Darmstadt, Germany
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6794-3/19/06.
https://doi.org/10.1145/3328905.3332512

deploy it for IF-specific tasks over their own textual information sources, or use it as a building block for added-value services. To demonstrate the customisability of Ping, we deployed and experimented (see Section 3) with it in two different textual IF scenarios: the DBLP[2] database for scientific publications and the textual part of the DBpedia[3] knowledge graph. Using Ping, we easily created an IF system that allows users to express their information needs and stay notified for new and interesting publications.

To this end, our contributions may be summarised as follows.

- We present Ping, a novel, fully-functioning IF system build entirely upon open-source components; the proposed system is able to support complex IF tasks in a variety of domains. To the best of our knowledge, this is the first open-source textual IF system that can *(i)* be deployed as a standalone solution on different textual IF tasks and domains or *(ii)* be used as a building block for other added-value services.
- We showcase the realisability of the developed system on two different domains (textual IF on scientific publications and crowd-sourced encyclopaedia articles), and experimentally assess its performance.

More information about the project, along with a fully-functional working deployment over DBLP publications may be found at: http://195.251.39.222/pingsys. After publication we will provide the source code of Ping under a GNU General Public License.

2 THE PING SYSTEM

The main idea behind Ping is to enable users to stay updated in a timely fashion, with new and interesting content that satisfies their needs. Ping achieves this objective by: *(i)* providing users with profile submission, *(ii)* information filtering, and *(iii)* notification delivery capabilities. In the following, we discuss the underlying functionalities of Ping, overview the system architecture and implementation details, and present how Ping may be deployed over different sources along with the main parameterisation options.

Profile submission. Ping enables users to express their information needs using several profiles. Each profile is formed by a set of constraints involving terms (expressed in any attribute of the incoming information) combined with textual operators. Profiles are submitted using a simple and intuitive user interface. For example in Figure 1(a) a user of Ping submits a profile expressing her interest in receiving new content that contain in their *title* the terms "retrieval" and "information" or "data" and "mining". The user may also specify additional constraints on the *authors* and *venue* attributes. Before submitting the profile to Ping for indexing, the user receives *a projection of the average notifications that will be produced* (per month) based on the constraints presented in the profile. This projection is an *estimate* from existing data and is used to help users assess the generality of their profiles. If needed, the user may refine the profile constraints; when, the profile submission is finalised, Ping indexes the profile in the data store.

Information filtering. The information filtering process is triggered every time new content becomes available by the monitored repository. In order to locate the newly produced information, Ping resides on periodically monitoring the source and retrieving all new published content that becomes available in XML format. When new publications arise, the Ping system commences the filtering process. At first, Ping retrieves and indexes all new content by making use of the Apache Solr framework. Subsequently, Ping retrieves all user profiles from its profile store and prepares them for execution under Solr. Finally, Ping executes each profile against the Solr framework that indexes the newly available content, thus determining which profiles match the new publication set. When the information filtering process terminates, Ping sends appropriate notifications to the users.

We chose to implement Ping's information filtering functionality over the Solr framework, which is primarily designed for information retrieval tasks, as there are no publicly available frameworks that natively support information filtering tasks. To this end, Ping implements the information filtering functionality by executing each profile separately against the newly published content that is indexed by the Solr framework. This fact alone highlights the need for and the importance of developing efficient IF-specific machinery to facilitate higher-level IF systems.

Notification delivery. When a profile is satisfied by an incoming publication, Ping delivers an appropriate notification to the user, through the notification center (Figure 1(b)). In the notification center, users may find notifications about information that matched their profiles coupled with direct links to the corresponding information. Finally, the notification center provides notification management capabilities (e.g., storage, dismissal) and advanced visualisation capabilities (e.g., notification timeline).

System architecture. The Ping system has been entirely designed on and developed using open-source software; it employs the Linux, Apache, MySQL and PHP (LAMP) stack as the back-end infrastructure, while the front-end modules have been developed using HTML, CSS and JavaScript. The information filtering capabilities of Ping are supported by the Apache Solr framework. Figure 1(c) presents a high-level view of the Ping architecture and the different modules that comprise the system.

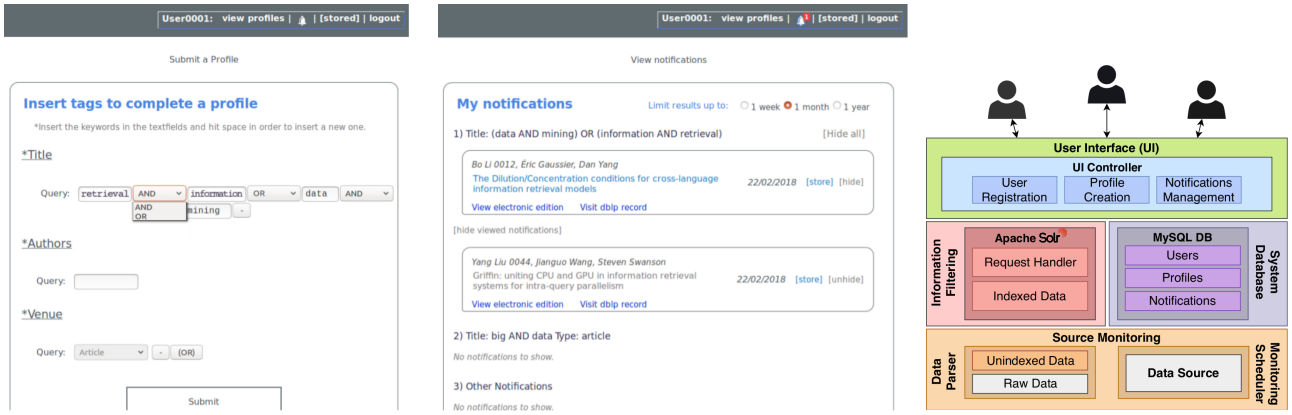
The *Source Monitoring* module is responsible for extracting new available content from information sources; this procedure is implemented through either continuous or periodic (i.e., at predefined intervals) monitoring by means of a wide range of data parsers that are able to accommodate different types of data sources.

The Apache Solr search framework lies at the heart of the *Information Filtering* module of Ping. This module receives all new content and indexes it under the Solr server. Additionally, the module retrieves all user profiles from the database and submits them trough appropriate service calls (using a REST API) to the Solr server for execution. Finally, this module receives the results from the Solr server and generates the appropriate notifications which are then handed over to the *System Database* module.

The *System Database* module is responsible for all the necessary storage and retrieval operations at the system back-end; it stores and manages all user account credentials, user profiles and associated notifications, relevant publications and other user-related data.

The *UI Controller* module is responsible for coordinating the operations of the Ping system and enabling a seamless communication between the UI and the underlying architectural elements.

[2] https://dblp.uni-trier.de/
[3] https://wiki.dbpedia.org



(a) Profile submission (b) Notification center (c) System architecture

Figure 1: Ping usage and architecture

Thus, the *UI Controller* serves as a mediator that receives and passes on data to various other modules. This module is also responsible for visualising all user interactions including *(i)* user registration and account management activities, *(ii)* profile creation, submission and editing, and *(iii)* notification delivery and management.

Deployment over different sources. The Ping system offers a variety of customisability options for its deployment over different data sources. The system administrator may easily set several parameters. Such parameters include *(i)* the type of the monitored data source, *(ii)* the source monitoring rate, *(iii)* the attributes that will correspond to the monitored data, *(iv)* the data manipulation rules (e.g., the employment of tokenisation or stemming) for the different data types of interest, and *(v)* restrictions on the generated notifications and preferred delivery method. The deployed online version of Ping (Figure 1) is setup to work with scientific publications from DBLP, with the following parameter setup: *(i)* monolithic XML files, *(ii)* a monitoring rate of 24 hours, *(iii)* attributes corresponding to the *title*, *authors*, and *venue type*, *(iv)* tokenisation and stemming enabled, and (v) on-site notifications only.

3 EXPERIMENTAL EVALUATION

In this section, we present a series of experiments that assess Ping over two distinct deployment scenarios.

Data and profile sets. In order to evaluate the filtering performance of Ping and demonstrate its real-world capabilities (and ease of customisability), we designed and experimented with two deployment scenarios. In the first scenario, we deploy Ping over the DBLP database for scientific publications. Thus, we utilise the DBLP corpus that contains all entries published during 2018 and consists of $786K$ publications, with a vocabulary size of $162K$ unique terms. As DBLP is a focused domain, we designed an additional deployment scenario that assesses the performance of Ping under a more general domain. To this end, we deployed and evaluated Ping also over the textual part of DBpedia, which covers a wide range of topics, with a total vocabulary of $3.2M$ unique terms.

Since no databases of profiles for neither scenarios were available to us, we synthetically generated one for each deployment scenario (similarly to [9, 11, 13]). These two profile databases are formed by conjuncts of different terms; each term conjunct is selected equiprobably among the multi-set of words forming DBLP and DBpedia respectively. Finally, for each profile set we randomly selected $50K$ publications and used them for the filtering task.

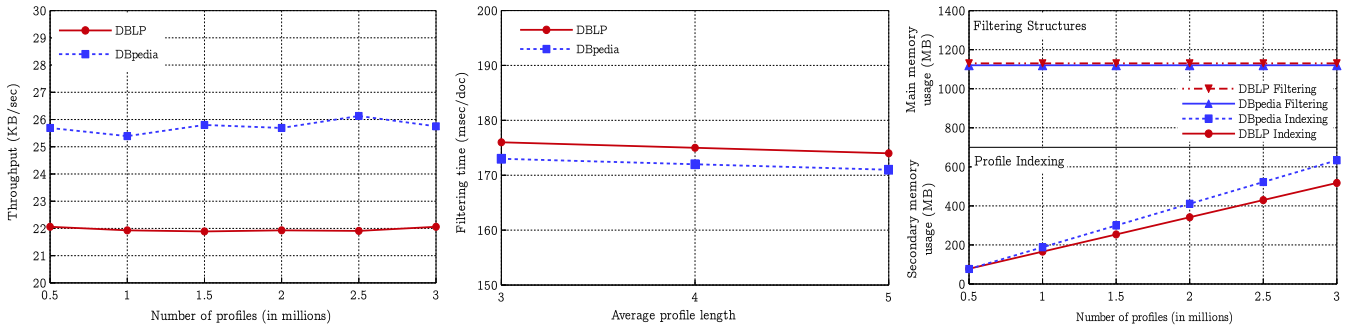
Technical configuration. A machine with Intel Xeon CPU E5-2650 $2.00GHz$, $32GB$ RAM, and Ubuntu Linux 18.04 was used to host the two deployment scenarios of Ping. The Apache Solr server was assigned $2GB$ of main memory. Time measurement report wall-clock time and the results of each experiment is averaged over 10 runs, to eliminate fluctuations in time measurements.

Evaluation results. Figure 2 presents the most interesting results regarding the deployment and evaluation of Ping.

Figure 2(a) shows the throughput in KB/sec needed to filter $50K$ incoming publications against a profile database of different sizes for an average number of 5 terms in the profile ($P_L = 5$), under both deployment scenarios. We observe that the throughput of Ping is consistent across the increasing profile databases for both cases. Moreover, the lower throughput of Ping under the DBLP domain, is attributed to the restricted vocabulary ($126K$ terms) of the topic-specific domain. A restricted profile vocabulary increases the probability to match a user profile against an incoming publication; these matches increase filtering time and hence reduce throughput.

Figure 2(b) shows the time (in milliseconds) that Ping requires to filter an incoming document in a database storing $DB_P = 3M$ profiles, when the average number of terms P_L varies. The performance of the system (slightly) improves as the average profile size increases since longer profiles (i.e., profiles with more constraints) exhibit a lower probability to match an incoming publication (high selectivity), and thus require less time to be evaluated by Ping.

Finally, Figure 2(c) presents the (primary and secondary) memory requirements of Ping for different sizes of the profile database. We observe that, in both deployment scenarios Ping exhibits low memory requirements, while transitioning from a topic-specific to a more general domain has minimum impact on memory needs. Ping indexes solely the incoming publication set during filtering time and executes each profile against it, while after the filtering phase completes intermediate results and publications are discarded. This approach allows Ping to exhibit constant memory requirements at filtering time, and allows it to efficiently support both topic-specific and general domains with low memory requirements.



(a) Filtering throughput for P_L = 5 **(b) Filtering time for DB_P = 3M** **(c) Memory usage for P_L = 5**

Figure 2: Evaluation of Ping for the DBLP and DBpedia deployment scenarios

The performance evaluation results reported in Figure 2 suggest that Ping may be utilised to efficiently support different textual IF tasks with high throughput and a relatively small memory footprint. The demonstrated efficiency, alongside the inherent profile expressivity delivered from the Solr engine, the different customisation options, and the open-source nature of the system, make it a promising solution for complex textual IF tasks in many domains.

4 RELATED WORK

In the last decades, content-based IF research has focused on providing efficient solutions on semi-structured data. These approaches utilised XPath as the (continuous) query language, and XML as the standard for publication representation. The first work that adopted the XPath/XML data model is presented in [1], where the XFilter system is described. XFilter employed Finite State Machines (FSMs) to represent each XPath query, while FSMs were utilized during publication time to determine the queries satisfied by incoming documents. In the same spirit YFilter [3] adopted a single Nondeterministic Finite Automaton (NFA) to represent the query set and share results between common parts of the XPath expressions. Similarly, in [5], the authors adopt a message batch processing approach to enhance the filtering process of publish/subscribe systems. Recently in [2], the authors adopt YFilter over a distributed computation environment, and implement YFilter's NFA approach over the Hadoop framework. In [8] the authors propose GPX-Matcher, a novel algorithmic solution that employs encoding for XPath and XML. GPX-Matcher exploits this encoding to capture commonalities among XPath profiles and uses them during matching time.

In the context of systems, SIFT [12] was the first content-based IF system that collected news and articles, and allowed users to subscribe into updates through web-forms and email. In SIFT users could utilise the Boolean or VSM models to express their information needs; in order to support the high volume of profiles and incoming publications, SIFT employed a tree-based solution [11] for the Boolean model and inverted index data structures [10] for VSM queries. Recently, [7] utilised graph structures to locate and index the subsumption relationships between continuous VSM queries and deliver the top-k most relevant notifications. InfoFilter [4] is another content-based system that aims at pattern detection over incoming text streams. In InfoFilter users can express their profiles through a pattern specific language, while the system created pattern detection graphs that were utilised during filtering time. Finally, the most widely used IF service to date is Google Alerts [6]. In Google Alerts, users can subscribe to new content by choosing topics of their interest or submitting VSM queries, while they can determine a wide range of parameters such as information delivery rate and channels over which information is retrieved.

5 CONCLUSIONS AND OUTLOOK

In this paper, we presented Ping, a customizable textual IF system built entirely over open source software, and experimentally assessed its performance in two different deployment scenarios. Our immediate plans involve extending the system by implementing a version for multi-core and cluster environments, integrating more formats for source monitoring, and incorporating non-textual IF (e.g., structural or graph constraints). Apart from the source code that will be released upon publication, we also plan to openly provide the research community with any (anonymised) profile dataset that may be constructed by the end-usage of Ping, in an effort towards realistic benchmarks for IF research.

ACKNOWLEDGMENTS

This work was supported in part by project ENIRISST from the General Secretary for ERDF & CF, under Operational Programme Competitiveness, Entrepreneurship and Innovation 2014-2020 (EPAnEK) of the Greek Ministry of Economy and Development.

REFERENCES

[1] M. Altinel and M.J. Franklin. 2000. Efficient Filtering of XML Documents for Selective Dissemination of Information. *VLDB*.
[2] P. Antonellis, C. Makris, and G. Pispirigos. 2015. Distributed XML Filtering Using HADOOP Framework. *ALGOCLOUD*.
[3] Y. Diao, M. Altinel, M.J. Franklin, H. Zhang, and P. Fischer. 2003. Path Sharing and Predicate Evaluation for High-performance XML Filtering. *ACM TODS* (2003).
[4] L. Elkhalifa, R. Adaikkalavan, and S. Chakravarthy. 2005. InfoFilter: a system for expressive pattern specification and detection over text streams. *ACM SAC*.
[5] P.M. Fischer and D. Kossmann. 2005. Batched Processing for Information Filters. *ICDE*.
[6] Google Inc. 2019. Google Alerts. https://www.google.com/alerts. Accessed: 25 Feb. 2019.
[7] W. Rao, L. Chen, S. Chen, and S. Tarkoma. 2014. Evaluating continuous top-k queries over document streams. *World Wide Web* (2014).
[8] M. Sadoghi, I. Burcea, and H.-A. Jacobsen. 2011. GPX-matcher: a generic boolean predicate-based XPath expression matcher. *EDBT*.
[9] C. Tryfonopoulos, M. Koubarakis, and Y. Drougas. 2009. Information filtering and query indexing for an information retrieval model. *ACM TOIS* (2009).
[10] T.W. Yan and H. Garcia-Molina. 1994. Index Structures for Information Filtering under the Vector Space Model. *ICDE* (1994).
[11] T.W. Yan and H. Garcia-Molina. 1994. Index Structures for Selective Dissemination of Information Under the Boolean Model. *ACM TODS* (1994).
[12] T.W. Yan and H. Garcia-Molina. 1999. The SIFT Information Dissemination System. *ACM TODS* (1999).
[13] L. Zervakis, C. Tryfonopoulos, S. Skiadopoulos, and M. Koubarakis. 2017. Query Reorganization Algorithms for Efficient Boolean Information Filtering. *IEEE TKDE* (2017).