# User Analytics with UbeOne: Insights into Web Printing

Georgia Koutrika, Qian Lin, Jerry Liu
HP Labs, Palo Alto, USA
{firstname.lastname@hp.com}

## ABSTRACT

As web and mobile applications become more sensitive to the user context, there is a shift from purely off-line processing of user actions (log analysis) to real-time user analytics that can generate information about the user context to be instantly leveraged by the application. *Ubeone* is a system that enables both real-time and aggregate analytics from user data. The system is designed as a set of lightweight, composeable mechanisms that can progressively and collectively analyze a user action, such as pinning, saving or printing a web page. We will demonstrate the system capabilities on analyzing a live feed of URLs printed through a proprietary, web browser plug-in. This is in fact the first analysis of web printing activity. We will also give a taste of how the system can enable instant recommendations based on the user context.

## 1. INTRODUCTION

User log analysis helps understand content consumption and user behavior. For example, analyzing people's searches helps search engines, such as Google, understand user search intent and improve search results [7, 8]. Analyzing people's clicks (Google News [6]), purchases (Amazon [5]) or movies viewed (Netflix [2]) can help generate personalized recommendations for items that the users would like to read, buy, and so forth. However, user log analysis is performed mostly off-line. As new applications become more responsive to the user context, there is a shift to real-time user analytics that can generate information about the user context that could be instantly leveraged by an application [1]. In many scenarios from generating contextual recommendations and ads [4] to fast application launching [9], waiting for the data to be analyzed to take an action may be unacceptable.

Furthermore, nowadays people interact with online content in many different ways. An individual may post a status update from his mobile, read an article on his tablet, edit a document from his laptop, send a recipe to the printer, and so forth. User actions such as searching, browsing, and purchasing, have been scrutinized but what about actions such as pinning a page in Pinterest or printing a page from the cloud using EFI PrintMe [3]?


*Proceedings of the VLDB Endowment, Vol. 6, No. 12*


Observing these challenges, we are building a user analytics system called *Ubeone*. Its key features can be summarized as follows:

- (*Analytics mechanisms*). At a high level, the system offers mechanisms for analyzing individual items consumed as well as aggregating information at the user level. The mechanisms are designed for analyzing strong user consumption signals coming from actions such as pinning, bookmarking or printing a web page or clipping an online coupon. Such actions, contrary to noisier actions like searching or clicking through a web site, can (even alone) provide important clues about the user.
- (*Additive processing and composeability*). The system is designed as a set of lightweight, composeable mechanisms that can progressively and collectively analyze a user action. To be as lightweight as possible, each mechanism is designed for a specific task and builds on the work of other mechanisms. Putting the system at work, one can select and wire available mechanisms together depending on the application. The same mechanism may be executed on different parts of the web page, such as the url, the title or the metadata. As data flows through the mechanisms, at each stage, each mechanism contributes in an additive way to the final output by generating additional knowledge, adding structure, consolidating data or generating more precise results. The system allows for efficient and flexible processing that can adapt to the requirements of the content and the application.
- (*Multi-aspect user analytics*). We consider that a user action, such as pinning a web page, can reveal different things about the user. For example, it can show interest in food recipes, locations that the user is visiting, upcoming trips or purchases. We have implemented mechanisms that can learn different pieces of information about a user as part of the user profile.
- (*Print analytics*). *Ubeone* is already used on generating analytics on a live feed of pages printed on the web. This is a proprietary, anonymized, data set containing URLs printed by users who consented to provide interaction data through a widely-distributed browser plug-in. This kind of print log analysis is the first of its kind and will provide the demo participants with the opportunity to gain insights into web printing.

We will demonstrate how different mechanisms contribute to the analytics and how they are flexible with different types of content (for example, in handling private pages or pages with poor textual content). We will show real-time graphs and insights into web printing using an online analytics dashboard. Finally, we will give a taste of how the system can be used in a application scenario that generates instant recommendations based on the user context.

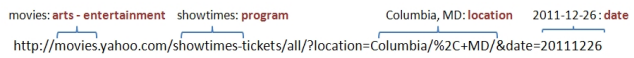



**Figure 1: Illustration of annotations**

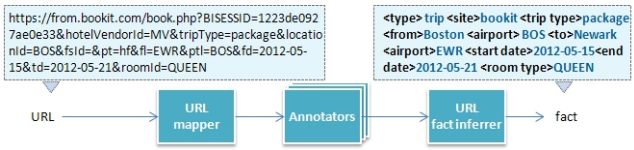



**Figure 2: Example workflow utilizing the URL of a page**

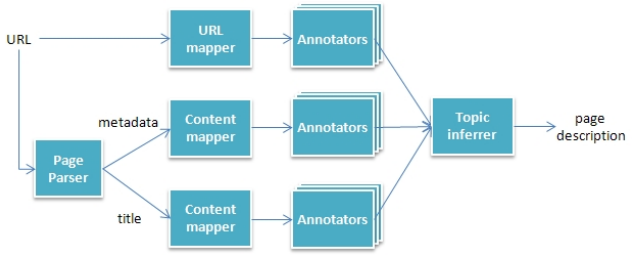



**Figure 3: Example workflow utilizing different parts of a page**

## 2. SYSTEM OVERVIEW

*Ubeone* comprises a set of mechanisms for analyzing text-based items, such as web pages, coupons, and products, and for aggregating information at the user level. Our system is driven by two major challenges: (a) how to process each item and generate useful knowledge as items 'arrive' in the system, and (b) how to enable different types of processing in a flexible and scalable way. To tackle these challenges, each mechanism is built so that it performs a single specific task and can build on the output of other mechanisms. In this way, mechanisms can be lightweight enough and generate additional results fast. Furthermore, mechanisms can be composed and combined to form processing workflows to tackle specific processing requirements.

Next, we present the main mechanisms. Then, we will show how these mechanisms can be composed into bigger processing flows. For simplicity in the presentation, we will talk about processing of web pages thereafter. The system is designed with extensibility in mind and we are constantly adding more mechanisms.

### 2.1 Analytics Mechanisms

Conceptually, the analytics mechanisms can be roughly divided into: mappers, annotators, inferrers, and aggregators.

A *mapper* generates a representation of its input that may be useful to other mechanisms. For instance, one mapper generates a term vector representation of its input while another one generates n-grams. A URL often represents a textual summary of the actual content or functionality of the web page. Hence, the URL mapper parses the URL into a structured object that captures the URL components. This object carries some initial semantics reflecting the site structure and query string. For instance, the URL *"http://movies.yahoo.com/showtimes-tickets/all/?location=Columbia %2C+MD &date=20111226"* could be mapped to an object like: {*sitename: yahoo movies; level1: showtimes-tickets; level2: all; location : columbia, md; date: 20111226*}.

An *annotator* typically builds on the output of a mapper and adds meaning through semantic annotations. Such annotations include attaching categories, attributes, and other meta data to the extracted information. Different annotators exist in the system. For instance, an approximate annotator attaches annotations by partially matching parts of its input to an external knowledge source. This source can contain taxonomies, named entities, and so forth. A confidence score is computed for each annotation based on how good the partial match is. On the other hand, an exact annotator matches exactly to entries in the external source. Finally, the pattern-based annotator uses patterns to generate annotations that describe dates, addresses, and so forth. Annotators have their own policy in resolving conflicts. For example, the approximate annotator may choose among two annotations the one with the maximum coverage.

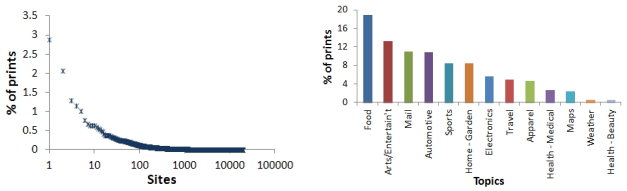
Figure 1 illustrates the idea of annotations using as input a url. For example, we see that the annotators have recognized 'movies' as belonging to the 'arts-entertainment' category. They have also recognized dates and location information. Note that the figure is used for illustration purposes and it does not show how the url and the annotations are actually represented internally.

The role of an annotation is to provide necessary and meaningful information for other mechanisms to use to measure the value of each annotation and utilize them accordingly. Therefore, in the system, an annotation is described by several fields that show which part of the input carries the annotation, the annotation label and id (in case the annotations are based on external knowledge), the type of annotation (e.g., topics, types, time, locations, etc), the type of the input (e.g., the url, the title of a web page, the metadata, etc) and a confidence score for the annotation.

An *inferrer* builds on the output of annotators to generate additional knowledge and add structure. For example, the URL fact inference mechanism is responsible for identifying user facts, such as purchases, upcoming trips, and so forth, based on information contained in a URL. A user fact is a structured record that has a fixed field type that shows the type of the user fact (such as trip, purchase, user location) and a number of variable fields that are defined and populated dynamically based on the information given by the annotations. For example, by analyzing a URL that contains airport codes and dates, the final user fact will represent that the user has booked a trip and information about this trip. Fact inference uses a relaxed grammar parser to define rules on how a certain type of user fact is identified and populated. For example, for the URL *"http://www.kayak.com/flights/BOS-SFO/2012-09-04/2012-09-11"*, a user fact could look like this:
⟨ type*: TRIP*, start date*: 2012-09-04*, end date*: 2012-09-11*, start location*: Boston*, start type*: airport*, end location*: San Francisco*, end type*: airport code*, travel*: flight* ⟩.

In a similar spirit, other inferrers extract structured information from the content of a web page. For example, for food recipes, ingredients, preparation times and guidelines are recognized as components of the recipe. Another yet type of inferrer is the web page topic inferrer that makes a more informed decision of the topics of a web page by combining data from different mechanisms. For example, it may combine annotations on different parts of a web page. This inferrer is a lazy topic classification mechanism.

An *aggregator* combines information obtained from analyzing different pages to build a user profile entity. Our user profiles bear two novel characteristics. First, they are multi-aspect, i.e., they capture different user aspects such as sites or brands that the user likes, categories and topics of interest, user activities (such as purchases and trips) and locations of interest. The aspects captured in the user profile depend on the type of items consumed. For example, from URLs that the user saves, we can learn about purchases and favorite




(a) Distribution of prints (b) Popular Topics

**Figure 4: Print insights**

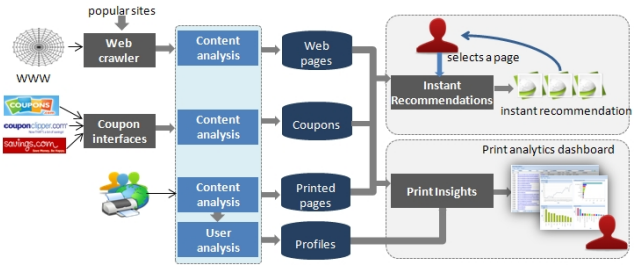



**Figure 5: Demonstration system and apps**

sites, from maps the user prints, we can learn locations of interest, and so forth. Second, our profiles are multi-dimensional, i.e., each user aspect is captured as a multi-dimensional vector, where dimensions include frequency, recency and interestingness. An aggregator is responsible for one aspect of the user profile and often for one dimension in that aspect.

## 2.2 Processing Workflows

Depending on the application and the characteristics of the input feed, we can decide to compose different mechanisms together in a layered way. One can think of each layer progressively adding 'more pieces to the puzzle'. It is possible that the same mechanism is used more than once in a workflow, for example to process different parts of the web page, such as the url, the title, the metadata, and so forth. To illustrate the above, we will discuss particular example workflows that are part of the current deployment of the system with the web print feed.

Figure 2 shows a workflow that uses only the URL of a web page to generate annotations and facts. We can use the URL to generate a more concise summary of a web page and the content to generate a more elaborate one. In many cases, the URL captures in a human-readable form the site's organization and a textual summary of the actual content of the web page. For instance, the URL *http://www.cnn.com/2012/11/06/politics/election-2012/index.html* hints us about the content of the page. This workflow provides a light-weight, non-obtrusive way to analyze web pages without looking inside their contents. Often, it may be the only way to derive any information about a page whenever its content is expired or changed (e.g., a news page, a trip reservation) or not easily analyzable (e.g., a web page with images).

Figure 3 shows a workflow that progressively combines information from different parts of the page through the topic inferrer to generate a description for the page. The topic inferrer compares the annotations of the previous layer to decide on the topics of the page. For example, a topic is boosted in the final output if it is supported by annotations coming from different parts of a page.

# 3. PRINT ANALYTICS

We are using our system on a live, proprietary, real-world data feed containing the anonymized URLs printed by users who consented to provide interaction data through a widely-distributed browser plug-in. This feed provides us with examples of real-world printing behavior. Each URL comes with a timestamp and a hashed IP.

We use *Ubeone* to generate print analytics that are unique since print logs have not been studied in this way before. These analytics shed some light into questions such as: the types of content people print, the kind of sites that attract web printing and the reasons reasons people print pages from the web.

Our analysis reveals several interesting insights into printing. Figure 4 shows some plots over the period Feb 2012 - Feb 2013. Figure 4(a) displays the print distribution for web sites. Note that the x-axis is in logarithmic scale. The plot indicates a power law distribution, i.e., there are very few sites from which users print a lot of content and then users visit a wide variety of sites and print content from. The latter comprise the long tail in the figure. Figure 4(b) displays popular topic categories.

# 4. DEMO DESCRIPTION

**Print Analytics Dashboard.** The primary source of data for this demo is the proprietary anonymized data feed of URLs printed by users through the browser plugin. We are using *Ubeone* to analyze the content printed and generate aggregate statistics. On top of the system, we have built a dashboard that offers different views of the data generated and allows a user to obtain various types of statistics. For example, the user may start from the distribution of popular print topics, then, focus on a specific topic and examine the distribution of sites and time and regional trends for this topic.

**Instant Recommendations.** *Ubeone* is built having in mind applications that need to respond to a user action almost instantly. For example, when the user is printing a trip reservation, coupons and content that would be helpful for this trip could be recommended and potentially be printed together with the trip document. As another example, if a user is interested in a food recipe, we could recommend related food recipes or recipe sites on the fly. To demonstrate how *Ubeone* enables such scenarios we have built instant content recommendations: we use content-based filtering to recommend content that is related to the content the user has chosen leveraging the output of the analysis performed on this content.

**Implementation.** Figure 5 shows the high-level processing pipeline for our demonstration system and the two demo applications. Each URL printed through the browser plugin is analyzed and the results are stored in the printed pages database. Any pages that may contain sensitive or personal data are identified early (by performing a light-weight URL analysis) and are not processed. Additionally, we apply our user analytics mechanisms to generate user profiles that capture user interests over sites and topics, user activities and locations of interest. Note that we do not have users, we only have hashed IPs. For the purposes of the demo, we will treat them as users. Figure 6 shows an example profile in the system.

For the recommendations, we additionally use two other sources of data. We use a list of popular web sites over different categories, such as news (e.g., CNN, BBC) and food (e.g., *allrecipes.com*, *foodnetwork.com*), and we periodically fetch all new content from these web sites. These pages are also analyzed and stored in our repository. We also get coupons from three providers (*coupons.com*, *couponclipper.com* and *savings.com*). For the first two providers, there is a REST API while for the latter there is an FTP feed.

In our current system, we use several external sources including hierarchies of topics and locations, dictionaries of important entities, and so forth. Note that the content analysis mechanisms do not

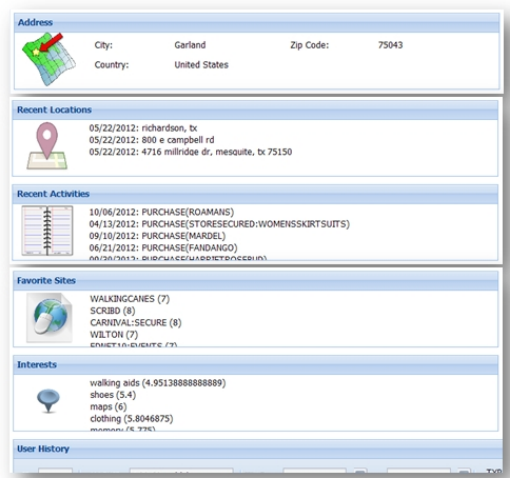



**Figure 6: Example user analysis**

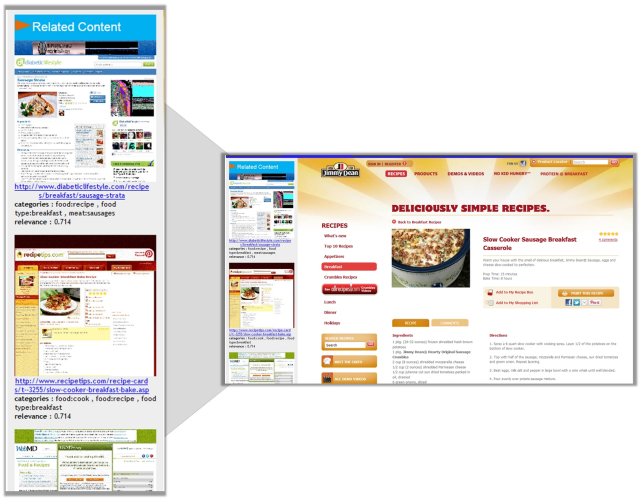



**Figure 7: Example instant recommendations**

depend on external sources used.

We use all three sources, i.e., web pages crawled, coupons, and pages printed by users, to make recommendations. We show recommendations in the following way. When a user clicks a web page, this page is instantly analyzed and the top three items are computed and shown in the context of the current page. The user may choose if he wants to see recommendations from other pages on the Web, or coupons that are relevant in the context of the current page or related pages that other users have found interesting and printed them. Figure 7 shows an example: on the right side, the web page that the user has currently selected is shown while the left side contains the recommendations for related web pages.

The main algorithms are provided as services and they are implemented in Java. Our crawling, content and user analysis workflows are also implemented as standalone jobs in Java. The recommendations are triggered by the user in the system and hence they are online. The demo interfaces are using JSP. Finally, we store processed data in Vertica.

## 5. INTERACTION WITH THE DEMO

Our demo is fully interactive and will allow participants to (a) see how the analytics mechanisms are applied to a real user data feed, (b) gain insights to the results of print analysis, and (c) see how instant recommendations can be generated on the fly.

Demo participants can use the dashboard to explore how different types of pages (e.g., food, entertainment, news, and travel) are analyzed, and understand how the mechanisms collectively generate distinct pieces of information about a web page. They can also explore different profiles and see how different user aspects can be learned. For example, we will see a 'foodie' profile versus a 'online shopper' profile. Demo participants will see an analysis of each profile showing user activity and trends.

Furthermore, the dashboard provides different views of the data computed, and a participant can navigate through the various options and graphs in several different dimensions (e.g., topics, locations, and types) to get insights into web printing. For instance, they can select a category and see the top sites for this category or choose a site and see what types of pages are visited from this site. They will be able to drill down in any of the dimensions and see aggregate results that focus on any combination of values in these dimensions. We will also use the interface to demonstrate some interesting trends such as those shown in Figure 4. This analysis provides interesting insights into print analytics that the participants can discover.

Demo participants can also explore the real-time analytic capabilities of the system through the instant recommendation interface. A person can select a URL and see how the system on-the-fly changes the content of the URL to show web page and coupon recommendations.

## 6. REFERENCES

[1] Beyond location: The rise of user context in apps. *http://www.guardian.co.uk /technology/ appsblog/2011 /apr/19/bluevia-location-based-services*, 2011.

[2] Netflix recommendations. *http://techblog.netflix.com /2012/04/netflixrecommendationsbeyond5stars.html*, 2012.

[3] EFI PrintMe. *http://w3.efi.com/Fiery/Products/Office/EFI-PrintMe*, 2013.

[4] G. Adomavicius and A. Tuzhilin. *Recommender Systems Handbook*, chapter Context-Aware Recommender Systems, pages 217–253. Springer Science+Business Media, 2011.

[5] G. Linden, B. Smith, and J. York. Amazon.com item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, Jan-Feb 2003.

[6] J. Liu, P. Dolan, and E. R. Pedersen. Personalized news recommendation based on click behavior. In *15th International Conference on Intelligent User Interfaces (IUI)*, pages 31–40, 2010.

[7] S. K. Tyler and J. Teevan. Large scale query log analysis of re-finding. In $3^{rd}$ *ACM International Conference on Web Search and Data Mining (WSDM)*, pages 191–200, 2010.

[8] R. W. White, P. N. Bennett, and S. T. Dumais. Predicting short-term interests using activity-based search context. In $19^{th}$ *ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1009–1018, 2010.

[9] T. Yan, D. Chu, D. Ganesan, A. Kansal, and J. Liu. Fast app launching for mobile devices using predictive user context. In $10^{th}$ *International Conference on Mobile systems, applications, and services (MobiSys)*, pages 13–19, 2012.