Large-Scale Real-Time Data Management for Engagement and Monetization

Simon Jonassen
Cxense ASA, Oslo, Norway
simon.jonassen@cxense.com

ABSTRACT

Cxense helps companies understand their audience and build great online experiences. Cxense Insight and DMP let customers annotate, filter, segment and target their users based on the consumed content and performed actions in real-time. With more than 5000 active websites, Insight alone tracks more than a billion unique users with more than 15 billions page views per month. To leverage the huge amounts of data in real-time, we have built a large distributed system relying on techniques familiar from databases, information retrieval and data mining. In this talk, we outline our solutions and give some insight into the technology we use and the challenges we face. This introduction should be interesting to undergraduate and PhD students as well as experienced researchers and engineers.

Categories and Subject Descriptors

H.3.4 [**Information Storage Systems**]: Systems and Software; H.3.5 [**Information Storage Systems**]: Online Information Services

Keywords

bigdata, online, real-time, web, analytics, personalization, targeting, ads, architecture, application, industry

1. DESCRIPTION

In the following we provide a short overview of our system, challenges and opportunities. This description can be used as an extended abstract/summary for the talk, supplementary to the slides, or as a brief standalone description.[1]

1.1 System Overview

Our solution can be deployed to a website with a single JavaScript tag. When the site is visited, the script requests a tracking pixel, thus informing us about, e.g., the visited url, referrer url or query, browser (name, version, time zone, language), device (type, brand, screen resolution), connection (type, provider) and geographical location (country, region, city). It also provides information from which we can uniquely identify users across domains and devices, detect bots and refreshes, and handle opt-out and do-not-track requests. Additionally, we are able to deduce the time spent on the page (dwell time), page visibility, exit links, etc.

Knowing which pages are visited we can easily discover newly published pages. When crawling a page we automatically detect the language and the page type (e.g., a topical front page, a search results page, an application such as a photo gallery, or an article). For articles, we automatically recognize and extract entities, including concepts, person and company names, and score them with a TF×IDF model.

Additionally, each customer can define a classification taxonomy to be applied to the extracted entities, or annotate pages with custom tags such as categories, targeted audience characteristics, authors, thumbnail images, article ids or canonical urls. Page views can be annotated with additional parameters, for example the user's registration and log-in status, or trigger user-profile and retargeting parameters. It is also possible to emit custom events associated with the user's actions such as hovering an ad or interacting with a menu or a video player. Finally, we also provide development kits (SDK) for easy integration with mobile and tablet apps or other connected devices such as smart TVs, set-top boxes and even radio-controlled toys.

For each user we maintain several profiles. A real-time profile tracks locations, devices, queries, intents, custom and retargeting parameters, etc. This profile is driven by the apriori information such as the item weight and recency of the action. Additionally, a long-term profile keeps user interests and preferred locations. This profile is driven by the aposteriori information such as the number of pages the user has read, the page popularity, or the dwell time. Customers can also provide their first-party data such as the user id, gender, date of birth, registration status or interests.

Cxense Insight UI is a Web-application providing a real-time overview of the current and historical traffic, content and audience via a set of predefined views and customizable dashboards. A UI widget may for example display trending interests, drop-off articles and gender distribution, or provide a comparison with the last week or month. The UI also facilitates interaction with the data – filtering by the content and audience characteristics, changing the time period, and aggregation for each site or across multiple sites.

[1]The presentation slides and other material are to be found at `http://s-j.github.io/cxense-at-lsdsir-2015`.

LSDS-IR'15, October 23, 2015, Melbourne, Australia.
ACM 978-1-4503-3781-6/15/10.
DOI: http://dx.doi.org/10.1145/2809948.2809953.

To manage all configurations and information we provide a RESTful API. All interactions that are possible in the UI/SDKs go via this API. In this sense, the customers can build their own interactive applications and programmatically manage their data in real-time. A subset of our API facilitates filtering and aggregation over billions of data points in matter of milliseconds. It allows the end-users to select the time period, sites, data types (such as the event, content and user profile parameters) and metrics (such as the number of events, unique users, urls and sessions, dwell time, and attention scores). Most importantly, it supports filtering by parameters from all types of data, boolean logical operators, time and count constrains on sub-filters, etc.

The API filters can be used to form segments for further reuse and user annotation (tagging). Segment membership can be used by third-party systems such as Google DFP and ExactTarget, or our own ad and recommendation solutions. Through machine learning we can also tag users that have properties similar to the users within a given segment.

Though some of the tasks are solved by the complementary products such as Content (recommendations and personalization), Display (display, video and mobile ads) and Search, all of the aforementioned can be used to measure and improve user engagement and monetization, drive personalized online experience, advanced search, recommendations and ads, subscription and conversion rate optimization.

1.2 Technology Overview

For our needs we have built a large and globally distributed system, mostly implemented in Java and running directly on Linux in form of continuous services and scheduled jobs. In this sense, we have a set of micro-services communicating either via HTTP or the file system, mostly using JSON input and output. Parts of our system are highly optimized in-house solutions such as the content processing graphs and data cubes, while some of it is more or less customized commodity software such as Elasticsearch, Apache Cassandra and Apache Spark.

Either way, our system is a symbiosis of techniques from information retrieval, databases and data mining. For example, linguistic processing and entity extraction are typical IR tasks, differentiating us from pure parameter-driven solutions, while being able to filter and aggregate billions of data points in real-time are typical database tasks, differentiating us from pure search and map-reduce based solutions.

The core of our system, known as the data cubes, is a column-store using succinct data structures and frame-of-reference compression to efficiently store the semi-structured data representing events, user and content profiles. It relies on idempotent feeding and snapshots, instead of transactions and recovery logs, and utilizes append-only data structures as much as possible. For aggregation we extensively use bit vectors, branch-free optimizations and intrinsic functions. This provides high throughput and short latency for both updates and aggregations.

The cubes can be fully in-memory or partially on-disk, partitioned by site, user, or both, and they can be specialized for specific tasks such as tracking segment membership. The query processor takes care of parsing the high-level (public) API requests, finding which cubes have the necessary information, scheduling the low-level (internal) API requests, combining the partial results and returning the final result.

1.3 Challenges and Opportunities

In the remainder of this summary we would like to share some of the interesting challenges we face.

We receive tens of thousands events per second from all over the world, sent from different platforms and browsers, often over unreliable network connections. The events must be delivered asynchronously and disregarding hardware and network failures. Moreover, dealing with large online news-publishers, we can observe traffic spikes above 200% on certain occasions such as presidential elections and cataclysms.

The incoming events must be reflected immediately. For example, impression capping for an ad-request may require us to find which of the given ads have been seen by the given user in a given period of time. This decision has to be done the next time the user loads a page and take 50 milliseconds or less. Another query may require us to summarize billions of data points to be displayed in the UI. Such queries can be issued each time a customer interacts with the UI, for example changes the view or refines the filter. Currently we serve tens of thousands aggregation requests per second.

We process webpages exhibiting a diverse specter with regards to their type, quality, language, configuration and server capacity/latency. Quite often the pages are moved or edited after being published, referred by multiple urls or served differently to different device types. Since our dashboards are typically used by the editorial team monitoring and optimizing the site performance in real-time, we must instantly reveal the new content and its changes. Presenting the most useful and precise information in the most meaningful way is another great challenge.

With a multitenant architecture we must prevent interference between the customers, provide a high and sustainable performance, and guarantee durability, reliability and availability. We tackle these challenges by for example running multiple data centers around the world and scheduling requests to the closest one, but we can also redirect the traffic in case of power outage, network failure or overload.

Security and privacy are two other issues worth mentioning. Some of our customers do not mind sharing their data, while the majority want to keep it private. We overcome this challenge with authenticated API requests and by managing each customer's data separately. We avoid storing sensitive information such as IP addresses or sharing our internal user identities. Nevertheless, we provide an efficient mechanism for linking with the customer's first-party data. As we cannot rely on third-party cookies, we deploy other techniques for user identification and cross-device tracking. Differentiating between the users sharing the same device and browser is an interesting challenge that we have yet to solve.

2. ABOUT THE PRESENTER

Simon Jonassen is a Senior Software Engineer at Cxense. He received his MSc and PhD degrees in computer engineering from the Department of Computer and Information Science of Norwegian University of Science and Technology in 2008 and 2013 respectively. His PhD thesis focused on efficient query processing in distributed search engines and was partly done in collaboration with Yahoo! Research Barcelona. Simon has joined Cxense in 2012 and is working on the API and the backend for Insight and DMP.

Acknowledgments: The presenter would like to thank the Oslo R&D team authored the solutions described herein.