Development & Integration of Social Data Connectors for Business Intelligence

Université de Technologie de Belfort-Montbéliard

David Hoeffel

January 31st 2016

Abstract

This document aims at presenting the theories & models developed as an Associate I.T. Engineer Intern during my stay at ActSocial (now part of Linkfluence) from August 2015 to January 2016.

My participation on various projects led me to question myself on how to process big data in a way to offer a real-time visual exploration experience.

Topics covered include : REST programming, MVC architecture, Crawlers, UI/UX, Sentiment analysis, Influencer Activation Platfroms, AW Services, Multithreading & Distributed systems.

Acknowledgements

Marc Rivoira For giving me the opportunity to conduct my internship at ActSocial, and reaching out to me.

Liu Kai For teaching me everything about crawling into data, using developer tools and implementing new structures.

Liu Zhehui For always being here when I needed help, and taking the time to help me with troubleshooting.

Chen Jian, Chen Lei & Qian Tianyang For being such good teammates, by sharing knowledge, helping me and guiding me to become a better engineer.

Ludovic Wassermann, Jeremy Rigaud & Francois Daugny For introducing me to ActSocial and giving me excellent base knowledge in project management.

UTBM and UTSEUS For giving me the opportunity to study in Shanghai and meet so many interesting people!

Contents

Chapter 1

Introduction to the Project

1.1 We are All Part of Big Data

Today, almost every one of us has a social account on the web. Wether you are using Facebook, Weibo, LinkedIn, or even Instagram there is always an identity that corresponds to you. Another growing trend we've been noticing these days is the use of centralised authentification services. Why? Simply because it's convenient. When you have to create a new account on a shopping platform or a support forum and there is a big button offering you to sign up using Google or Facebook, it's a pretty tempting option - but yet again, another link to your social identity.

In the past people could only interact with things around themselves, leading to smaller impacts but also smaller reach and possibilities. With the birth of internet people can act on real entities they don't even see. To put it more simply, you don't just go to your retail store to order, pay and take your good using your voice, eyes and hands but you go on an online retail store. This leaves a trace about you : your tastes, your general location, but also your personality.

1.2 How Data Structures Define us

Based on simple informations such as the amount of products sold on e-commerce websites, hot topics on forums or likes on posts, statistics can be made. Of course, these informations are based on anonymous data, publicly accessible. That is where we start talking about sentiment analysis and machine learning. Sometimes, with sentiment analysis, it is very easy to know if someone is delighted or not - if the user is provided with an explicit option to indicate so. But when it's not the case we have to rely on machine learning.

Every data item is different and therefore unique : it has been created at a specific time, from a specific place, in a specific part of the web and in that respect it defines us, people, at the source of this data. Being able to reverse engineer that information to classify it can prove to be a difficult task. Information on the web is chaotic just as humain nature can be unpredictible at times, and this is the reason why accuracy indexes need to be used.

1.3 Representing Social Variables

The challenge of representing data comes only after a structured and classified system has been set up. Data can then be represented in a raw form, simply based on it's aggragated values, or in a processed form based on the output of machine learning algorithms or formulaes. Either way, data visualization is another challenge in itself involving giving the user the right information at the right time - otherwise it may hardly make sense. By building the right back-end structure, data can be easily manipulated for visualization allowing live feeds and updates, complex filters and representations.

When representing data the challenge is always to make it as simple to understand as possible. Graphs must tell a story of their own, a story that is intuitive and makes immediate sense for the viewer. For this feat to be possible large amounts of data are required and in depth tailored processing has to be made to help for further interpretation.

1.4 Building Insights from Knowledge

Even after using computing power to crack open the social sphere of virtual data, it all boils down to requiring a team of real human engineers to get the most out of the models that have been made. Sure, diagrams tell their own stories and help get general and detailed views based on a situation or referencial, but they don't bind information outside of their scope of analysis and sometimes lack bottom-line information businesses are looking for. Outside the areas covered by a product that can give the same type of service to anyone, adjustements and tayloring must been made by human beings in teams of researchers. Machines are very performant at doing an incredible amount of complex tasks but can't learn and proceed properly without correction or guidance. In fact, machines are very bad at acheiving abstract tasks, which humans are good at.

This whole process leading to the creation of detailed insights and the understanding of the big social data surrounding us is a challenge to set up, but proves to be of invaluable use on the long term. *The rest of this* *report is dedicated to explaining how this detailed process articulates itself around major tasks that have been accomplished during my internship. Since October 2015, ActSocial has been part of Linkfluence ; both companies will therefore be introduced.*

Chapter 2

Corporate Structure & Goals

2.1 ActSocial by Linkfluence

2.1.1 A Global Social Intelligence Group

Linkfluence was founded in 2006 in France, has now more than 100 employees in Europe and more than 300 clients worldwide including brands, agencies and public organizations. By getting the most out of social conversations on Facebook, Twitter, Instagram, LinkedIn, online media, blogs and forums, their goal is to generate new opportunities to manage corporate reputation, customer relationship and social network performance.

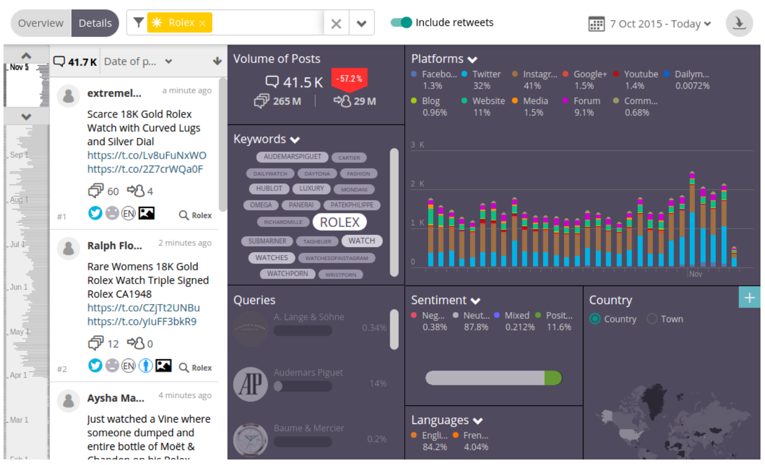
On October 15th, **ActSocial** was acquired by Linkfluence thus renamed "Linkfluence Asia". The objectives of this merge is to enable new listening features in Asia, and provide to businesses located in Asia with insights on the emerging market. A complete merge between tools, research teams and technologies has been launched to provide the best experience in social media listening. Today ActSocial counts about 50 people, and has offices in Singapore and Shanghai.

Figure 2.1: ActSocial & Linkfluence logos

Understanding Big Data can prove to be a complex task when it comes to digging out the information that is really valuable, and this is the reason why ActSocial & Linkfluence have brought together a team of social media experts along with highly efficient products. ActSocial's motto is "Insight, Action, Influence" true to the three areas of services provided : offering social media websites listening and providing Businesses with detailed analysis based on the areas of the industries they wish to cover, empowering them to change the conversation.

2.1.2 Two products, one service

Along with their services, ActSocial & Linkfluence offer a software that can track and analyze millions of publications from millions of sources on a daily basis. With Radarly companies can view, analyze and interact with social feedback from their customers in order to take the best actions for the future.


extremel... a minute ago
Scarce 18K Gold Rolex Watch with Curved Lugs and Silver Dial https://t.co/Lv8uFuNxWO https://t.co/2Z7crWQa0F

Ralph Flo... 2 minutes ago
Rare Womens 18K Gold Rolex Watch Triple Signed Rolex CA1948 https://t.co/CZjTt2UNBu https://t.co/yluFF3bkR9

Aysha Ma... 4 minutes ago
Just watched a Vine where someone dumped and entire bottle of Moët &

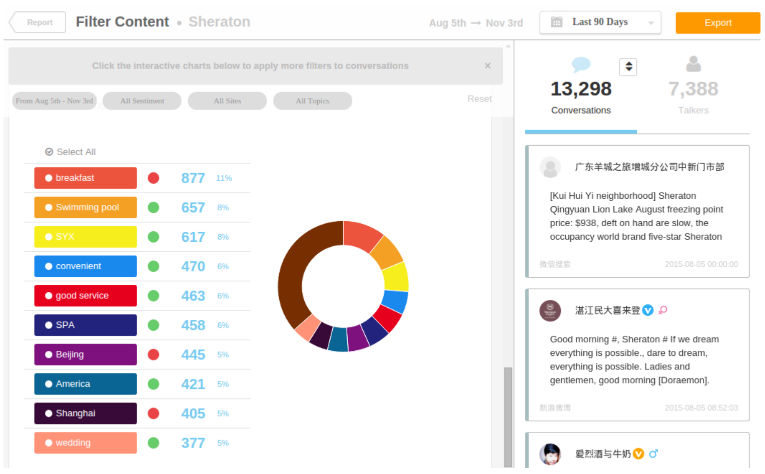
Platforms	
Facebo...	1.3%
Twitter	32%
Instagr...	41%
Google+	1.5%
Youtube	1.4%
Dailym...	0.0072%
Blog	0.96%
Website	11%
Media	1.5%
Forum	9.1%
Comm...	0.68%

Sentiment	
Neg...	0.38%
Neut...	87.8%
Mixed	0.212%
Posit...	11.6%

Languages	
Engli...	84.2%
Fren...	4.04%

Figure 2.2: Linkfluence's Software Radarly

The software provides a global view of the selected industry with key values and trends, in which users can dig in for deeper analysis. Insights provide a set of graphs to view, compare and get a more detailed analysis on keywords and sentiments. The last feature being intergrated into both software is to be able to engage with major influencers by tagging and interacting with them.



breakfast	🔴	877	11%
Swimming pool	🟢	657	8%
SYX	🟢	617	8%
convenient	🟢	470	6%
good service	🟢	463	6%
SPA	🟢	458	6%
Beijing	🔴	445	5%
America	🟢	421	5%
Shanghai	🔴	405	5%
wedding	🟢	377	5%

13,298 Conversations
7,388 Talkers

广东羊城之旅增城分公司中新门市部

[Kui Hui Yi neighborhood] Sheraton Qingyuan Lion Lake August freezing point price: $938, deft on hand are slow, the occupancy world brand five-star Sheraton

微信搜索 2015-08-05 00:00:00

湛江民大喜来登

Good morning #, Sheraton # If we dream everything is possible., dare to dream, everything is possible. Ladies and gentlemen, good morning [Doraemon].

新浪微博 2015-08-05 08:52:03

爱烈酒与牛奶

Figure 2.3: ActSocial's Software

Using both tools, research experts can provide businesses with valuable reports based on these insights. The finals steps in the process are to engage with their customers on a larger scale for some companies, by implementing an Influencer Activation Platform (IAP).

2.1.3 Technology & Engineering

The I.T. service is comprised of 6 engineers split into three major teams depending on the work context, and the abilities of each member : Liu Kai – CTO, Liu Zehui – Lead Engineer, Qian Tianyang – Lead Support Engineer, Chen Jian – Back-End Engineer, Chen Lei – Support Engineer, David Hoeffel – Front-End Engineer

Support Responsible for maintaining the crawler base, handling Zendesk bug tickets and network issues. Main tools used are Kibana, ActSocial's home-made crawler monitoring tool, and ruby for programming.

Infrastructure Entitled to discuss and shape up major structural changes by integrating new concepts and new technologies, data storage, transmission and architecture.

Back-end & Front-end Tasks to carry out design intergrations or data process changes on different projects.

2.2 Internship Role & Responsibilities

Subject: Collection, processing & representation of social media data for business intelligence.

During my internship I was lead to working on various different projects, ranging from website development to creating a threaded database content extractor. All objectives have allowed me to envision the global scope of being a web engineer, architect and developper in an environment handling Big Data. The main projects I contributed to are listed below.

2.2.1 Achievement Goals

- Developing and designing new websites
- Integrating enhanced features to ActSocial's web listening software
- Optimizing data crawling algorithms
- Setting up new crawling structures and managing data streams
- Data Access and Streaming in and out of the Crawler architecture

August & September Assisting in the development of client's Influencer Activation Platforms for Desktop and mobile. Front-end programming and interface intergrations on ActSocial's software and websites. Carrying out the version evolution of ActSocial's listening software, by intergrating a new custom date range feature.

October & November Back-end programming on the crawler base involving optimization, site exploration and rule updates. Reshaping ActSocial's business website. Implementing Kestrel queuing. Benchmarking Linkfluence's webpages from Asia. Performance tests for Influencer Activation platforms, managing AWS services for fast content delivery and virtual computing.

December & January Technology merge with Linkfluence, change in crawler architecture with a Scala proof of concept. Push of Asian crawling data from ActSocial to Linkfluence's software Radarly. Redesign of ActSocial's website to match with Linkfluence Asia. Work on a threaded data exporter from ActSocial's databases to Xml files, and a Sina Weibo stream reader with Akka.

2.2.2 Management Tools

Pivotal Tracker is our drop-off chest of new development features, bugs and ideas for ActSocial. Bugs on the front-end or back-end can be listed, categorized and assigner in real time in the team. Our icebox section let us list hypothetical enhancements that would see use on the long term. All our version release and deadlines are listed on the dashboard for a concise and efficient production rate.

Github & Gitlab that store all our projects in various branches. We decided to use a Git also as a log of our changes and to synchronize our numerous commits on a daily basis. All projects use different technologies and need a specific setup, described and maintained throughout the Git repositories. IAP projects also have their own repositories before deploying them on the clients server. Crawler intelligence is regrouped in numerous files in their corresponding project repository.

Deploying in a test environment was necessary for the teams to benchmark and test different apps before release. The client and our designer also connected with us to input their feedback, making a cycle of corrections before final release.

Chapter 3

Crawling into New Sets of Data

3.1 Introduction

ActSocial set up a whole infrastructure allowing topics to be detected and classified automatically, with semantic data grouped into categories using machine learning algorithms. The sentiment of Chinese content can be determined with a precision of 75%. Social media users creating the most impactful content related to a specific industry or product category are then identified. But before being able to represent social data it must first be collected. The web is immense and full of social interactions happening every second, setting the task of gathering data in real time not to be an easy one. Therefore, in order to make such a thing possible we have to use Crawlers to perform various tasks of indexing, detection and automatic updates for us.

3.1.1 Crawler Structure

Crawlers are automated internet bots that perform web indexing tasks. They create a bridge between web data and local networks and are used to pilot the storage of content into databases. The target goal is to ultimately extract social content from websites, store it and analyze it for representation through diagrams. When fetching data from the web, our crawlers focus on social content only : posts from users online. Once a page is opened the goal of the crawler is to find a list of posts, and gather their body, author and date so that all relevant social information can be stored and analyzed. Crawlers are all shaped up the same way:

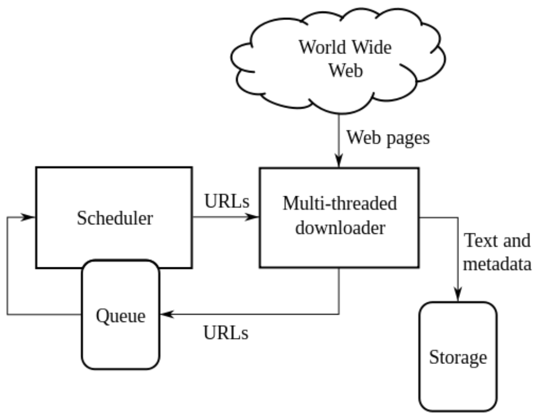
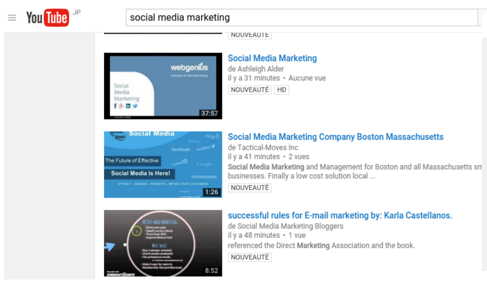


Figure 3.1: Architecture of a web crawler

To gather and update as much content as possible, a threaded architecture is used to enqueue Topic URLs that need to be analyzed. The URLs in queue are then popped by other threads and processed by a Scheduler which is used to define crawling times, priorities and make sure the URLs aren't malformed or pointing to non-processable or irrelevant web pages. In the end, a parser (not shown on graph) gathers text, pictures and other metadata from the webpage's url - stores the content in a database and updates the url's record. Other crawler jobs have been chosen to be handled such as the detection of usernames, dates, comment items and pictures so that a more detailed record of the social content can be made.

3.1.2 Terminology

ActSocial's Crawlers use the notions of Channels and Threads. Channels represent a section of a website containing links to discussion Threads. Those Threads feature content that needs to be collected ; each one can be broken down into a series of comments. Sites with social activity considered are Blogs, Forums, Social Network Services (Twitter, Facebook, Instagram etc.), E-Commerce websites and more.



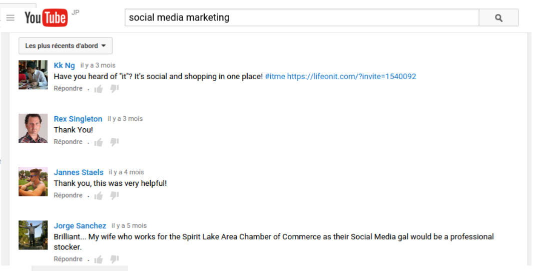
NOUVEAUTÉ

Social Media Marketing
de Ashleigh Alder
il y a 31 minutes • Aucune vue
NOUVEAUTÉ HD

Social Media Marketing Company Boston Massachusetts
de Tactical-Moves Inc
il y a 41 minutes • 2 vues
Social Media Marketing and Management for Boston and all Massachusetts sm businesses. Finally a low cost solution local ...
NOUVEAUTÉ

successful rules for E-mail marketing by: Karla Castellanos.
de Social Media Marketing Bloggers
il y a 48 minutes • 1 vue
referenced the Direct Marketing Association and the book.
NOUVEAUTÉ

(a) A search Channel with Threads



Kk Ng il y a 3 mois
Have you heard of "it"? It's social and shopping in one place! #itme https://lifeonit.com/?invite=1540092
Répondre

Rex Singleton il y a 3 mois
Thank You!
Répondre

Jannes Staels il y a 4 mois
Thank you, this was very helpful!
Répondre

Jorge Sanchez il y a 5 mois
Brilliant... My wife who works for the Spirit Lake Area Chamber of Commerce as their Social Media gal would be a professional stocker.
Répondre

(b) Post items within a Channel link

Crawlers also obey different policies such as a selection policy for selecting pages to analyze, or re-visit policy defining when to check for changes in those pages. With the use of these accurate information is automatically gathered when new posts appear. In the crawler system defined, two types of Channels exist : those which are Category-based and those Keyword-based. Using keywords to define channels is of use when a website posesses a custom search engine ; it can consequently return a list of Threads containing social data.

Because every website has a different structure, it's not easy to precisely sort and classify crawled data. Channel urls are stored in a database as records with attributes to define how and when they must be crawled. Each crawler at ActSocial relies of CSS selector rules and follows a specific crawling scheme:

- Get a site's channels
- For each channel:
  - Get the channel page content
  - Get the channel links to topics
  - For each topic:
    - Get the topic page content
    - Get the post items
    - For each post item:
      - Get the post date and time
      - Get the body text & content
      - Get the author information

3.2 Preliminary data treatment

3.2.1 Parsing content

When Crawling in a Thread, we decided not to store only the raw HTML content of the page : each post item has to be selected and the elements within it stored seperatly. In order to perform such tasks, web pages need to be parsed from their url. Many methods are available to accomplish that and return an element on which CSS selectors can be used to target specific items. The selection of content based on tags is how Ruby analyzers were built at ActSocial - so they can call a set of CSS rules stored into variables, and apply them on a parsed web page.

```
doc = Nokogiri::HTML(open(uri),nil,charset)
post_dates[] = doc.css('div.post_date_divs')
```

When parsing a document, encoding also has to be taken into account, especially for Asian languages. Charsets such EUC-JP (for Japanese), EUC (for Korean), GBK or GB18030 (for Chinese) are used to parse the web pages correctly. Furthermore, Crawlers need to use custom request headers to specify cookies, referer uri and other settings when opening a web page to parse.

```
agent.request_headers = {
      "Referer"=>"www.host.com/channel_id",
      "Host"=>"www.host.com",
      "Cookie" => "cookie1;cookie2;cookie3;"
    }
```

3.2.2 Analyzing & Storing content

Once all the relevant data has been selected from a page document, it needs to be converted in its correct type. Posts have a body, a date and an author, and all tags need to be processed accordingly. Author names have to be stripped from the HTML elements, and dates are converted from strings to actual date objects with the correct regional format. This helps to shape a concrete comment object, that can be saved as record in the database.

```
DateTime.parse(date_str)
username.strip.to_s
cssQueryImage.first['src']
```

Before storing a web page's content into the database and indexing it, some controls must be made to ensure the url's reliability and make sure the content isn't already saved in the database. When Thread urls are collected, some parameters can be appended to them and need to be removed from the url to prevent the creation of duplicate references in the database. Posts contents are then saved as snapshots of a web page in buckets from AWS Scalable storage service (S3), and new records are added to SimpleDB, a document-oriented database that stores part of the content analyzed from a web page.

To transmit post items to Linkfluence's own analyzer used for their listening software Radarly, posts are streamed to a Kestrel queue as JSON elements. Setting this up improved transfer speeds and entity sizes. Using JSON also has the advantage of being best-suited as a data-exchange format over XML, and provides structure that is easy to process and to read.

3.2.3 Queueing Systems

After our merge with Linkfluence, it was necessary to figure out a way to inject the post items we had gathered from our crawlers into their pipeline

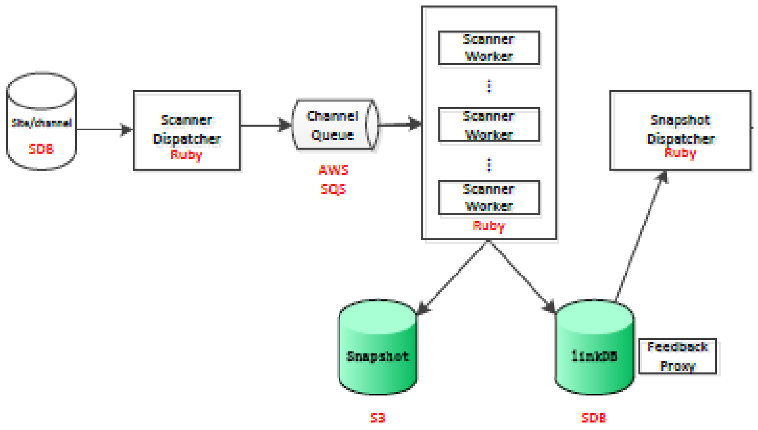


Figure 3.3: Crawler Dispatcher and Snapshotting

so that they could use the data with their listening product Radarly. We used queues since they allow the storage of items in a definite order, and is known to be one of the most efficient methods of transmitting data between two different systems in a fast and reliable way. Queueing systems are used as libraries for easy access and monitoring, they work well with crawlers that send text data structures. Four queues were created to match our crawler post item's main categories : bbs (forum), blog, review and all other less prolific types.

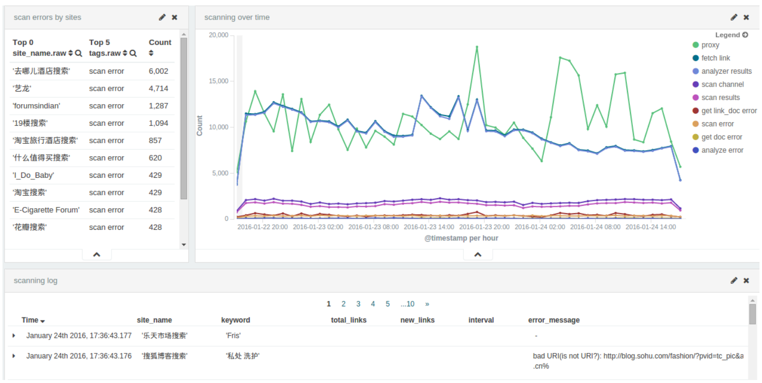
Indexed social posts are naturally processed using the First In First Out scheduling policy, with Kestrel that supports Memcached text-based commands, useful for displaying specific queue attributes. Memcached and Thrift were used for advanced monitoring commands and in order to perform basic actions to test the fair delivery of the queued items. To transfer the smallest item possible, post items are converted to JSON objects and then Zipped.

```
string = post_hash.to_json
msg = ActiveSupport::Gzip.compress(string)
kestrel_client=KestrelClient.new("server-example.com:port");
GET queueName /peek
```

3.2.4 Performance Monitoring

Testing is necessary in order to ensure that the crawlers return the right data and algorithms have been created to check the integrity of Selector rules, as well as the success of post indexing. Those tests are made directly in the rails console with terminal output. The crawler base is monitored through Kibana, that allows to display statistics in real time on crawlers.

The IT team has also set up a web interface to help manage these crawlers. For a selected site, we can manage its rules, channels and details.



Top 0 site_name.raw	Top 5 tags.raw	Count
'去哪儿酒店搜索'	scan error	6,002
'艺龙'	scan error	4,714
'forumsindian'	scan error	1,287
'19楼搜索'	scan error	1,094
'淘宝旅行酒店搜索'	scan error	857
'什么值得买搜索'	scan error	620
'I_Do_Baby'	scan error	429
'淘宝搜索'	scan error	429
'E-Cigarette Forum'	scan error	428
'花瓣搜索'	scan error	428

Time	site_name	keyword	total_links	new_links	interval	error_message
January 24th 2016, 17:36:43.177	'乐天市场搜索'	'Fris'				-
January 24th 2016, 17:36:43.176	'搜狐博客搜索'	'私处 洗护'				bad URI(is not URI?): http://blog.sohu.com/fashion/?pvid=tc_pic&a .cn%

Figure 3.4: Kibana Monitoring

Logstash & Elastic Search are both required to generate a usable output for Kibana. Deployed on multiple servers, Logstash collects running application messages and sends a filtered version of the data to ElasticSearch. Filters help define the types and alias of the data handled by Logstash. This way, our whole system can be monitored with each query, future, array being logged. Finally, ElasticSearch acts as a search engine used to look for content from our systems of log data displayed in Kibana. Kibana is deployed on a cluster called ELK that handles the search engine and graphical display UI of the data stored into each Logstash from Working crawler servers.

Although working well most of the time, the architecture wasn't easily maintainable and involved to manually set a lot of specific rules for every new site crawler that needed to be used. Using CSS selectors has the advantage of being easy to implement with a HTML document parser, but requires adaptation for every single website. The decision was taken to build a more Serializable crawler algorythm using Scala language to accomplish tasks of collection, indexing them and storage. This new structure is has a much faster processing speed, and was the opportunity to enhance some of the processes used.

Chapter 4

Implementing a Distributed System Architecture

4.1 A large-scale complex system

4.1.1 Architecture needs

Data Capture is just a first step in the whole process that leads to visualizing it. Between these two milestones, numerous phases take place such as Data Classification, Sentiment identification and tagging for social content - and on the other end a Zombie Detection for social accounts on SNS platforms. This common knowledge allows the identification of real influencers on specific areas of the market. The major goal here was to focus on an efficient data capture system with pre-processed data allowing easy classification and further processing.

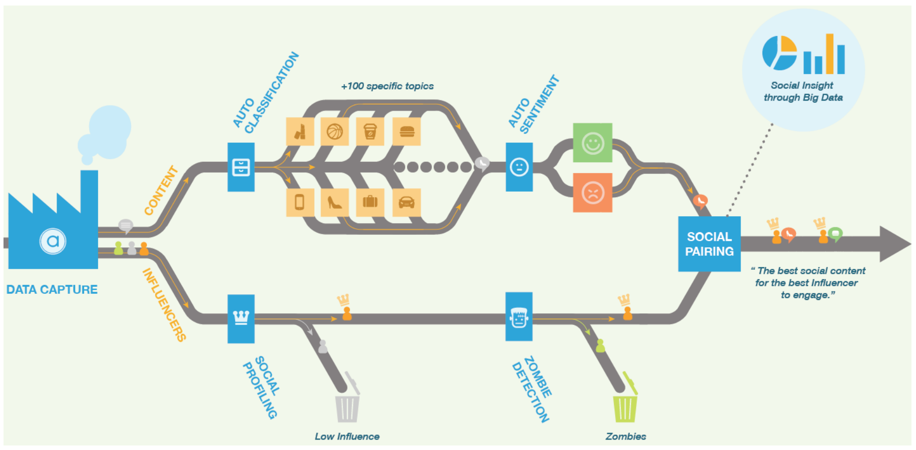


Figure 4.1: From Data Collection to Representation

A Crawler needs to handle huge amounts of tasks, the end goal being as fast as possible when analyzing web pages. Its functions can be seperated into four categories : fetching, filtering, parsing, and streaming. We want the crawlers to be as fast as possible, so we built an architecture were there is as minimum wait as possible. For this to happen a few requirements must be met, such as having a distributed system, load-balancing, and using efficient toolkits such as Akka to provide message-driven actions and a constantly running system. Our Crawler system is made of 3 clusters : one hosting Spark, one Hadoop, and one containing Worker Servers communicating with each other using Akka.

The system also needs to be fail-proof, meaning that if a server crashes, data gets jammed or slowed replacement actors must be elected as the deficient nodes are being restarted, and parallel processing is a key point in ensuring continuous data treatment. Distributed Systems help achieve that goal as they are the coordinators of messages between different processes executed within the network. Crawling into web pages also means managing a uri pool of visited uris, updating their content or indexing new links. Webpages that have already been crawled into should only have their content updated.

4.1.2 Concurrency & Actor Systems

The Scala code for our crawlers is then compressed into a JAR file with SBT Assembly and deployed on four worker servers. Each crawler instance process features Akka actors that communicate between each other with messages - Servers and Clusters are managed by Zookeeper for work load and availability. Zookeeper becomes a very powerful tool able to monitor Queues, and all working machines within a cluster. Sparks allows the implementation of RDD features within the Big Data Crawler System. To ensure the running state of most of the services, we issue the "jps" command to list all the Java Virtual Machine instances on our target systems.

A high system throughput requires a high amount of concurrent operations. To make it possible, a series of messages are sent between actors of the system to coordinate them. Akka is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM. We decided to use the Akka system as a way to manage at a high level our processes and interactions between them. It allows the creation of Actors, which become instances of a class able to run concurrently, using methods to coordinate itself with other Actors. Therefore, every Actor has a "receive" method, executing itself at the reception of a message. The use of this system also implies two other features : the integration of finite state machines and thread-safe objects.

```
State(S) x Event(E) -> Actions (A), State(S')
```

Within Akka Actors are used Promises that result to Future objects, allowing for a variable to be created whenever the Promise has achieved its fetching and processing data. This allows to create multiple objects that are returned whenever the Actor has finished data treatment. A Future referes to a variable storing the result of an asynchronous process or method called Promise. The Completion of that Promise can happen at any time, and sets of actions are usually defined at that moment depending on the Future value. Future objects are used in Akka and instanced by the reception of a message. It has the advantage of being retuned when data is bound, asynchronously from the crawling process in execution. An actor model a primitive of concurrent computation: in response to a message that it receives, it can make local decisions, create more actors, send more messages, and determine how to respond to the next message received.

Akka supports multiple different models of concurrency, and language bindings exist in Java and Scala, making its intergration much easier. In the Akka system designed, the Dispatcher is the central piece of the message flow. It can be Idle, Active or Busy whether it is starting, receiving or passing a message. Actors need to be initiated with a state : startWith(Idle,Empty) and change state upon the reception of messages handled by cases. Messages are sent using "!" or "?" and trigger names events to other actors of the system.

Crawler Dispatcher:

```
when (Idle, stateTimeout = idleTime) {
    case Event(Frontier.Msg(msg), _) =>
      handleDequeuedMsg(msg)
      log.info("[dispatch] changeStateTo Active '' 0")
      goto(Active)
    case Event(StateTimeout | Frontier.Empty, _) =>
      feedQueue ! Frontier.Next
      stay
  }
```

Crawler Frontier:

```
def receive: Actor.Receive = {
    case Next =>
      Future {
        next match {
          case Some(msg) =>
            Msg(msg)
          case _ =>
            Empty
        }
    } pipeTo sender
```

4.2 Network Interactions

4.2.1 Cluster Management

In order to maintain a load balance and synchronize all Crawling servers, Zookeeper was used as a distributed coordinator. In our system, it keeps the different actors sychronized. Each time an Actor wishes to perform a task, it addresses its request to Zookeeper that will coordinate all clusters in the system. If a cluster is down, Zookeeper will automatically redirect the load on another cluster. If a master server is down, Zookeeper can also re-elect a master server to coordinate sub-tasks. Each cluster that had been set up contains a set of servers represented by Amazon EC2 instances. The Spark cluster contains two Workers and one Master. The Akka cluster contains four Workers. The Hadoop cluster contains four storage servers running respectively Hadoop, HBase, ElasticSearch and Kibana. Logstash is ran on the three Spark servers. The Kestrel Queue is used to output data from our system to Linkfluence's servers in France.

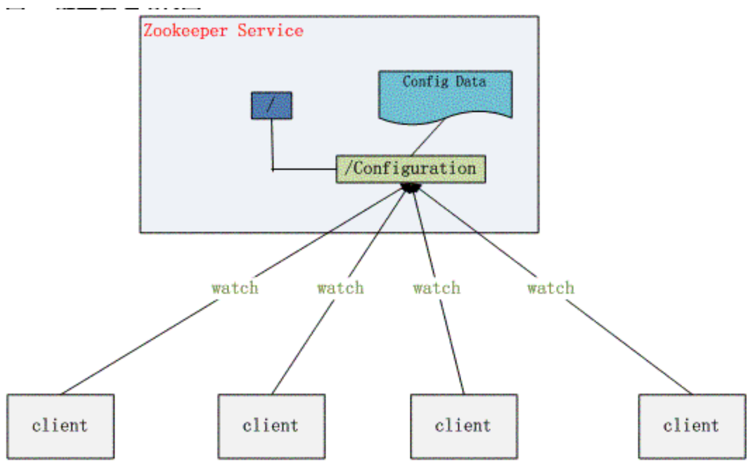


Figure 4.2: Zookeeper Configuration Spread

4.2.2 Crawling & Message Flow

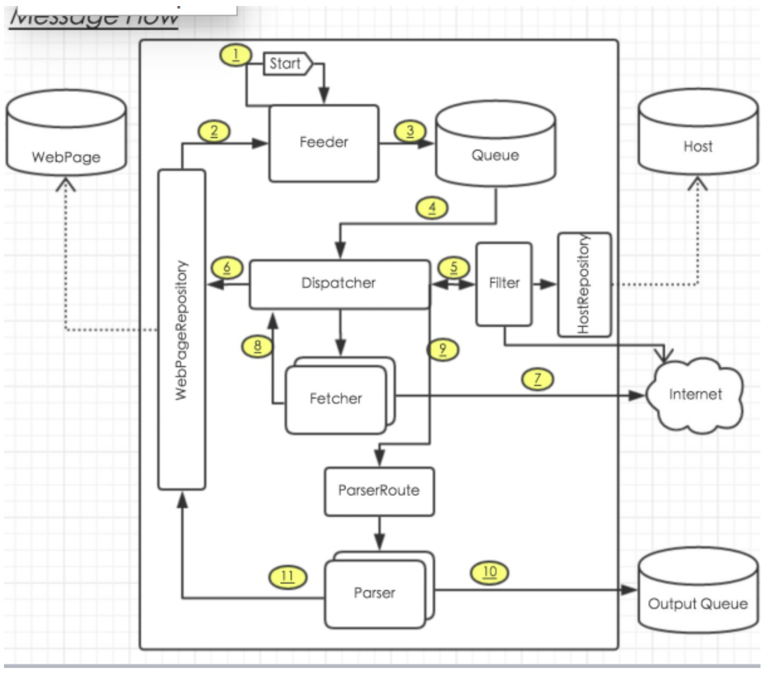


Figure 4.3: New Crawler Message Flow Diagram

The first actor in the crawler topology is the Feeder. It's job is to collect uris from the database and enqueue them periodically. There are multiple feeders running simultaneously to quickly push uris to an initial queue of webpage urls.

The Dispatcher has multiple roles : the first is to dequeue items for the source queue only if Fetchers are available ; the second is to transmit the url to the Filter for inspection before handing it to a Fetcher. After the Fetcher has done its job, the Dispatcher then passed the fetched data to a ParseRoute that will direct the result to an available Parser.

The Filter checks if a given url is a webpage object that can be crawled. It also compares the url with custom rules from the source website, including the Robots Exclusion Protocol specifying source repositories not to crawl.

The Fetcher retrives the header and the content at the specified url, and informs the Dispatcher back of its availability.

The Parser's job is to parse the web content to the JSON format, stream it to an output queue and update the status of the processed webpage in the database with a next crawl date and priority. The Parser instances different types of content Extractors to respectively extract document Metas, Html, Images, Links and Rss content if needed from webpages.

Spark offers features of a RDD (Resilient Distributed Dataset), meaning that data can be persisted in memory and recomputed if part of it is lost. Data can be read and processed concurrently, and operations on it are seperated into tasks that are being executed only when required (laziness). When handling crawler operations, to ensure that there is waiting time between our Parser, Dispatcher and Filter an RDD helps us save a lot of time by managing data in our system.

4.2.3 Distributed Storage

In order to store the large amounts of data collected by the crawlers, Hadoop was the perfect choice to allow quick access of large files and scalability. There is so much data in the crawler databases that almost every request returns large amounts of files. Hadoop allows a fail-safe cluster of data replication and can manage servers that are down. The Hadoop Distributed File System, or HDFS, uses MapReduce to process files in batch mode - meaning that files arent returned individually but by big sets when a request is made. MapReduce has the advantage of being thought for scalability. Since batches of data are requested when they need to be represented, the Mapper is going to find what we seek, and the Reduce process will recombine each part of the objects requested back together from each server in the Hadoop Cluster. HBase is a distributed scalable NoSQL database, and it supports random read and write access to data.

Since the plan was to store large amounts of objects with a structure that can't be predefined, the requirements for our storage system were for it to be : schema free, support basic SQL and scalable. Crawlers have multiple sources, ranging from simple forums to SNS platforms like Twitter that don't return the same types of structures, and are subject to policy changes in the future. Due to the changeability of these data structures, it was necessary to use a Scalable and Schema-free database supporting a minimum level of SQL queries. SimpleDB was chosen as it is an Amazon Service providing all these features in an easy-to-use package. Data is stored into SimpleDB as documents, meaning that we never need to care about the structure of the objects our objects. We than iterate the Database with simple SQL queries to return desired social topics, users or post objects.

Chapter 5

Processing Information into Vizualisations

5.1 Data Exploration

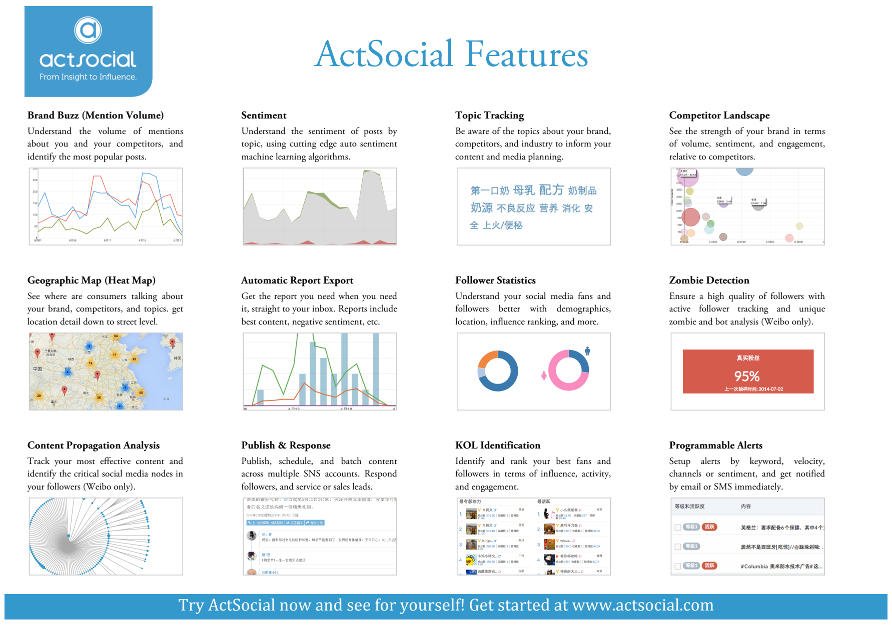
The challenge of representing data goes through two processes : machine learning and visualisation frameworks. To build visualizations withing ActSocial's listening software the d3.js library was used as it quickly offers a set of dynamic graphs adapted to different representations. Representing data is the final step of data gathering, it comes after data has been filtered, sorted, processed using algorithms. Data visualizations need to be dynamic for the end-user to truly comprehend what he is looking at. It is with that mindset that Radarly and ActSocial's listening software were designed.

5.1.1 Interactivity & Preprocessing

Machine learning is a set of algorithms used to predict statements based on sample data. This is how sentiments can be tagged based on adjectives. Nevertheless, Chinese sentiment analysis goes through additionnal character matchings as there are no space between words, and two characters may have different meanings based on their combination.

After being processed, many visualisations can be made from data. A graph factory has been coded in the back-end code to instanciate and link data objects to different graphic views. It is also possible to handle exports for multiple datasets, with each graph having an equivalent spreadsheet structure. Every display page is generated for a specific time range, data being queried from the database based on predefined periods as follows:

```
Database -> Controller queries -> Data Object -> Graph instance -> View Generation
```


ActSocial Features

Brand Buzz (Mention Volume)
Understand the volume of mentions about you and your competitors, and identify the most popular posts.

Sentiment
Understand the sentiment of posts by topic, using cutting edge auto sentiment machine learning algorithms.

Topic Tracking
Be aware of the topics about your brand, competitors, and industry to inform your content and media planning.

Competitor Landscape
See the strength of your brand in terms of volume, sentiment, and engagement, relative to competitors.

Geographic Map (Heat Map)
See where are consumers talking about your brand, competitors, and topics. get location detail down to street level.

Automatic Report Export
Get the report you need when you need it, straight to your inbox. Reports include best content, negative sentiment, etc.

Follower Statistics
Understand your social media fans and followers better with demographics, location, influence ranking, and more.

Zombie Detection
Ensure a high quality of followers with active follower tracking and unique zombie and bot analysis (Weibo only).

Content Propagation Analysis
Track your most effective content and identify the critical social media nodes in your followers (Weibo only).

Publish & Response
Publish, schedule, and batch content across multiple SNS accounts. Respond followers, and service or sales leads.

KOL Identification
Identify and rank your best fans and followers in terms of influence, activity, and engagement.

Programmable Alerts
Setup alerts by keyword, velocity, channels or sentiment, and get notified by email or SMS immediately.

Try ActSocial now and see for yourself! Get started at www.actsocial.com

actsocial
From Insight to Influence.
第一口奶 母乳 配方 奶制品
奶源 不良反应 营养 消化 安
全 上火/便秘
真实粉丝
95%
上一次抽样时间: 2014-07-02
最有影响力
最活跃
等级和活跃度
内容

Figure 5.1: Listening Software Features

5.1.2 Development milestones

Intensive bug search sessions have been conducted in order to identify and list front-end and back-end operations to make as fixes. This led to further alterations involving the way data is presented with the implementation of new features. A new software release was planned to allow the selection of a custom range of time to represent social data online, involing major changes that had to be done.

Offline tasks retreive every day social data from the database using predefined periods of time, implying that even if a new period of time is requested by the user, the data can't be accurately pulled in real time. This limits queries to the database of 1 per day. For the user to be able to track trends with greater precision, setting up a custom date range picker would be necessary. Data is sorted by the controller according to predifened periods of time : 90-60-30 Days ago, 1 Months ago, 1 week ago, Yesterday, Today. A period is bound to a specific range of dates by the controller. The moment.js library was used to Parse, validate, manipulate, and display dates in JavaScript. Every instance of time is generated with this library that offers an object that can be easily manipulated. Creating a custom date range requires to query the database in order select data according to the period selected. The instruction of creating a new period is therefore given to the database if that range doesn't exist yet. Custom user periods are then added to the list of periods in the database and scheduled to be generated from the following day on. This is a constraint that had to be dealt with as queries can't be made in real-time do to the heavy amount of data in the database.

If a custom period is within the ranges of period categories that are available, the controller can simply filter social data, thus immediatly display it to the user without querying the database again. Daterangepicker.js is a JavaScript component for choosing date ranges. It's designed to work with the Bootstrap CSS Framework and has the asset of being able to combine category-based period selection with a custom range option, presented in a user-friendly way.

When trying to represent social data with d3.js, some conditions can go wrong especially with the use of location or site fitlers on a large scale. Making sure there were no inconsistencies in the data, with bug-free fast and responsive graphs. Wrong data models linked to graphs or incorrect behaviours and display artifacts were fixed with an in-depth troubleshooting through controller data, and structure manipulation by the front-end.

5.2 Influencer Activation Platforms

ActSocial possesses intelligence related to key influencers and trending topics. The idea is to use this data to help businesses identify pain points and plan their next move. The core idea of such a platform is to create value by targeting new potential influencers amongst a company's customer base. By gaining the loyalty of their customers, businesses can create more word-of-mouth opportunities driven by people who have a real impact on social media.

5.2.1 Features

Influencer Activation Platforms aren't meant to persist on the web, but rather to create temporary opportunities for customers to gain rewards, integrate a company's culture and apply to some jobs. The engagement process goes as the following : users have missions they can accomplish such as reading or commenting tasks. They are rewarded for each individual action they make with points that allow them to redeem gifts. The goal of such a platform is to create favorable relationships between potential customers and give birth to real influencers that can testify of the quality of a brand on the web.

Every Mission has a timed deadline and gives a defined amount of points for achieving it. Their are meant to entice social activity between participants and make them spread the word about the tasks they are involved into. Throughout my internship three IAPs were made for SAP, Actualyse and StyleXStyle. Those three clients shared the same project, but heavy customization was made by branching the code.

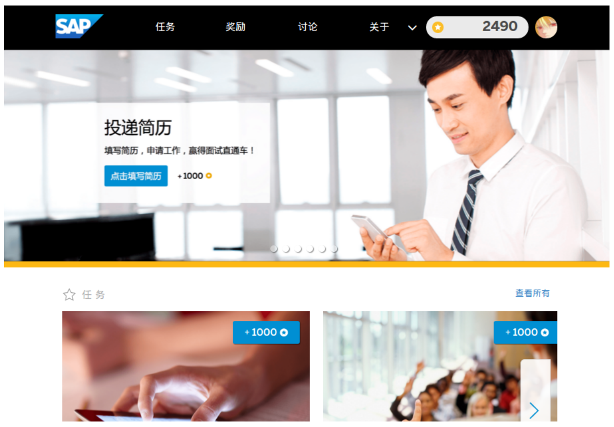


Figure 5.2: SAP China's Influencer Activation Platform

5.2.2 Development milestones

With several clients to manage, style settings were regrouped into one SASS file for easier customization of fonts, colors and logos. Taking into account mobile devices also required CSS alterations to match different screen widths and network speeds. The use of Facebook or Weibo authentification provided us with API keys and secrets for our web applications. For development purposes we used the OAuth library to connect to SNS as if we were authentifying directly from the web, making Weibo believe that we are authenticating to him via the online SAP url port 80 (but actually we are from localhost IP with port 3000). Rails is not authorized to use port numbers under 1000, because they are system reserved - it was necessary to use Apache2 to make a Proxy pass since it has the rights to use any port on the system.

```
    sudo a2enmod proxy_http
    Creating a new virtual host:
    sudo cp 000-default.conf saplabschina.conf
    sudo a2ensite saplabschina
```

Finally, custom javascript methods were coded to manage ajax data submission of forms, custom jQuery effects or dynamic web page elements. About a total of six projects were affected by changes made in the front-end. In order to maintain the most generic code the I18n format, libraries such as Polyglot were used within HBS files to support multiple languages.

Chapter 6

Conclusion

6.1 Challenges

After gathering some basic knowledge about data mining within modern MVC architectures and frameworks, I worked mainly as a front-end engineer with responsibilities on back-end ruby programming.

Influencer Activation Platforms At my arrival I worked on developping an Influencer Activation Platform (IAP) for SAP, third project of this type carried out by ActSocial. Some technologies and frameworks were already in place such as the use of Ruby on Rails, with an MVC Angular and Bootstrap intergrated architecture. I mainly participated in setting the technologies used as a new standard for future projects and shaping the interface as our development process went on for our SAP and Actualyse clients.

ActSocial's listening website ActSocial's listening sofware was coded using technologies such as Node, Bower, Backbone and Bootstrap. Along with the team, we referenced and categorized bugs and enhancements for ActSocial's dashboard. Each of us worked on different assigned tasks coordinated using PivotalTracker. My responsibilites were to set up a front-end structure capable of intergrating data structure objects from the back-end, and communicate with other engineers in order to implement design changes.

Social data crawlers ActSocial created a base of hundreds of crawler channels selecting topics on social websites on a daily basis, and storing them into databases. Data is processed with a global ruby thread analyzer using customized rules for every website. Maintaining and improving the crawler base involved working with ruby and home-made web-tools in order to retreive new data in the correct data structures and databases.

Policy updates & new projects The company's business website was set up using Jekyll, Bootstrap and Jquery. Restructuring the homepage had me working along with a user experience engineer based in Los Angeles to integrate Linkfluence's updated policy and changes to ActSocial's website. Commissioned to create a document database exporter, work also had to be made with a colleague to read and store content from an AWS SimpleDB to JSON formatted files and involved the use of Threads and popular libraries.

6.2 Outcome

The time I spent as an intern at Actsocial started with a lot of challenges and ended being full of great lessons and experiences. I felt encouraged by my colleagues who helped me learn a lot about distributed systems and how Big Data is handled and collected from start to finish. This experience led me to improve my development skills on many levels and discover a lot of new languages, framework and general technical knowledge. The great team communication lead us to work together and collaborate on many projects with various needs in features or set up.

After the merge with Linkfluence, new objectives appeared with the combination needs of ActSocial's data with Linkfluence and led to new projects to stream our knowledge and content to the French IT team. It is still the beginning of an expanding company which aims to analyze and step firmly in Asia, and make use of the Chinese and strong social networks there. The global outcome of my internship can be summed up with the milestones below:

1. ActSocial's business site redesigned
2. Kestrel queueing system integrated
3. Version update for Actsocial's listening software
4. New crawler concept set up
5. Crawler base maintained and improved
6. Influencer Activation Platforms deployed
7. SimpleDB Distributed Exporter developped
8. Sina Stream Reader project ongoing

With these projects carried out these 6 months, considerable progress have been made on web platforms set up by ActSocial, also on their listening software and their global crawler system. I have learnt a lot of valuable lessons and technical knowledge on database management, AWS, server setup, crawlers and general web development - web architectures and frameworks. Participating in the writing of technical articles on the team blog has also proven to be an enlightening sharing experience, and helped in the understanding of technical concepts. Today these projects are still being improved and new objectives have been planned with Linkfluence to exploit new data from Asian social networks in the following months.

Bibliography

[1] Our technical articles
`http://blog.leanote.com/actsocial`

[2] Actsocial Crunchbase Profile
`https://www.crunchbase.com/organization/actsocial`

[3] http://queues.io/

[4] http://www.shmula.com/queueing-theory/

[5] http://www.daterangepicker.com/

[6] http://jmt.sourceforge.net/

[7] http://momentjs.com/

[8] http://redis.io/

[9] http://gruntjs.com/

[10] https://www.npmjs.com/

[11] https://nodejs.org/api/

[12] http://yeoman.io/

[13] http://bower.io/

[14] https://rvm.io/

[15] https://jekyllrb.com/

[16] http://www.scala-sbt.org/

[17] http://kafka.apache.org/

[18] http://www.json.org/xml.html

[19] http://docs.scala-lang.org/style/

[20] http://nutch.apache.org/

[21] http://doc.akka.io/docs/akka/2.4.0/scala/fsm.html

[22] https://www.thoughtworks.com/radar

[23] https://thrift.apache.org/

[24] http://api.rubyonrails.org/

[25] http://docs.aws.amazon.com/

[26] https://www.nomachine.com

[27] http://watir.com/

[28] http://www.w3schools.com/

Appendices

Appendix A

Primary Crawler Data Stream

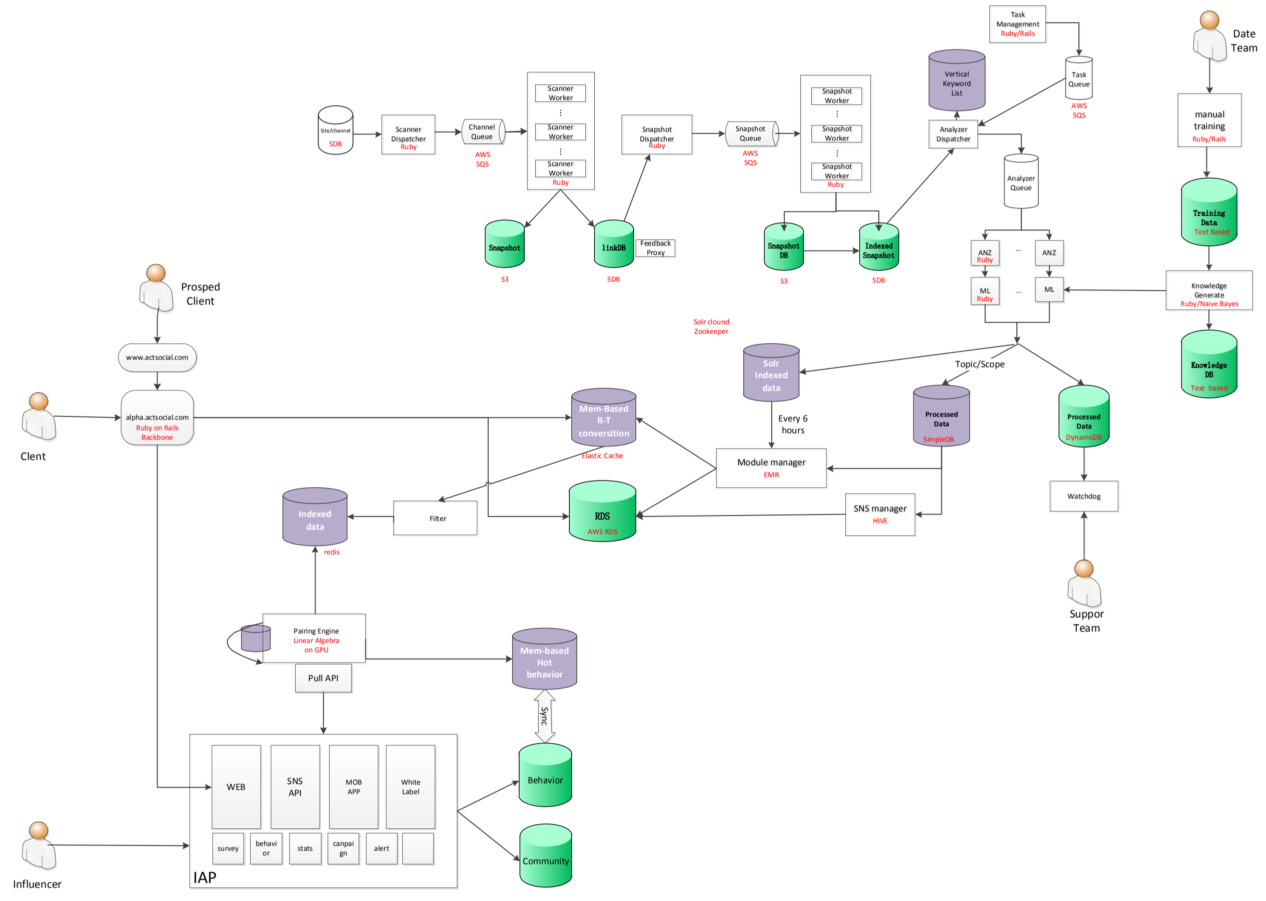
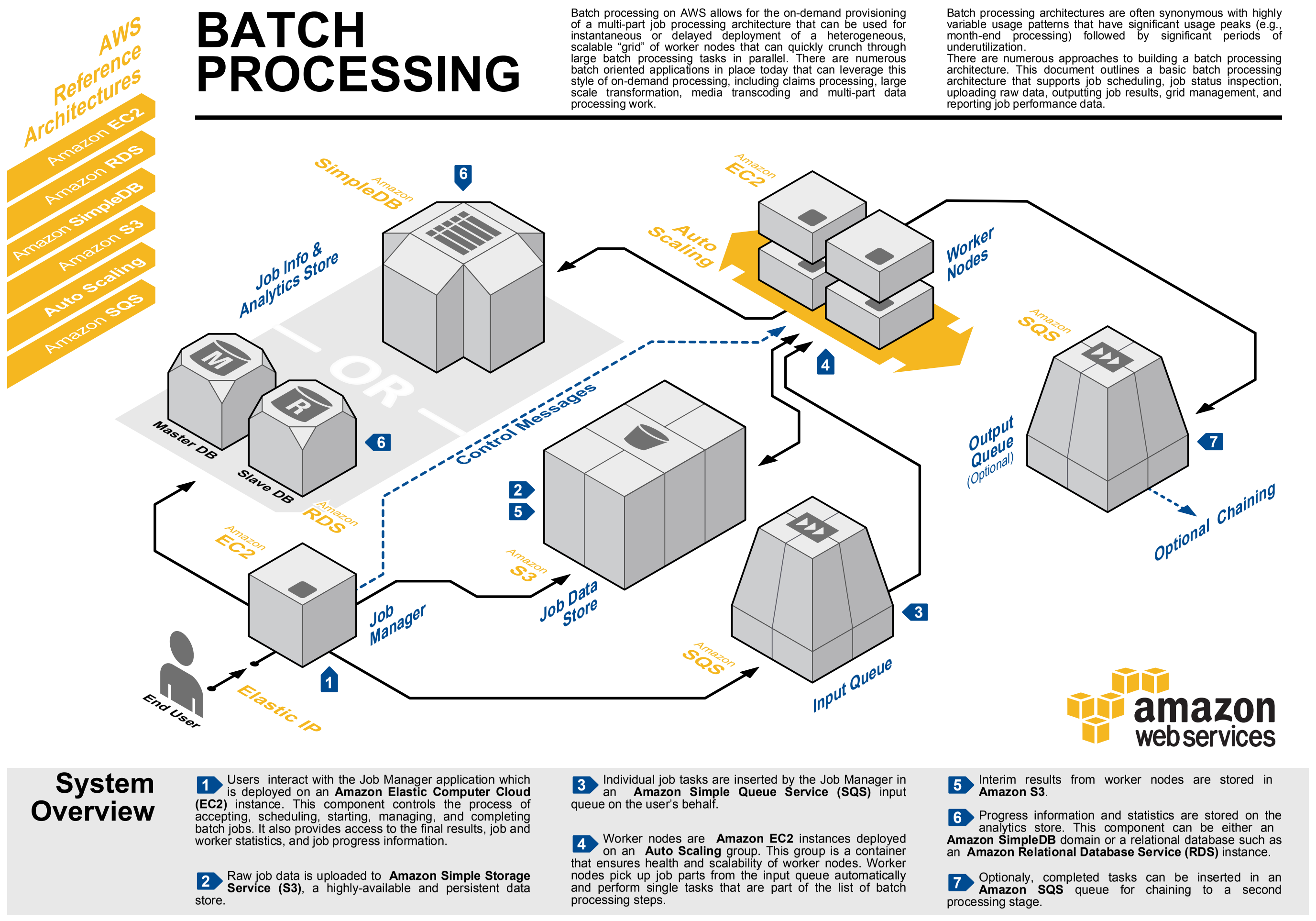


Figure A.1: Data Stream from collection to representation within the network.

Appendix B

Batch Processing Crawler Architecture



BATCH PROCESSING

Batch processing on AWS allows for the on-demand provisioning of a multi-part job processing architecture that can be used for instantaneous or delayed deployment of a heterogeneous, scalable "grid" of worker nodes that can quickly crunch through large batch processing tasks in parallel. There are numerous batch oriented applications in place today that can leverage this style of on-demand processing, including claims processing, large scale transformation, media transcoding and multi-part data processing work.

Batch processing architectures are often synonymous with highly variable usage patterns that have significant usage peaks (e.g., month-end processing) followed by significant periods of underutilization.
There are numerous approaches to building a batch processing architecture. This document outlines a basic batch processing architecture that supports job scheduling, job status inspection, uploading raw data, outputting job results, grid management, and reporting job performance data.

System Overview

1 Users interact with the Job Manager application which is deployed on an **Amazon Elastic Computer Cloud (EC2)** instance. This component controls the process of accepting, scheduling, starting, managing, and completing batch jobs. It also provides access to the final results, job and worker statistics, and job progress information.

2 Raw job data is uploaded to **Amazon Simple Storage Service (S3)**, a highly-available and persistent data store.

3 Individual job tasks are inserted by the Job Manager in an **Amazon Simple Queue Service (SQS)** input queue on the user's behalf.

4 Worker nodes are **Amazon EC2** instances deployed on an **Auto Scaling** group. This group is a container that ensures health and scalability of worker nodes. Worker nodes pick up job parts from the input queue automatically and perform single tasks that are part of the list of batch processing steps.

5 Interim results from worker nodes are stored in **Amazon S3**.

6 Progress information and statistics are stored on the analytics store. This component can be either an **Amazon SimpleDB** domain or a relational database such as an **Amazon Relational Database Service (RDS)** instance.

7 Optionaly, completed tasks can be inserted in an **Amazon SQS** queue for chaining to a second processing stage.

Figure B.1: Interactions between Worker nodes and Storage cluster.

Appendix C

Social Network Service Crawling Tasks

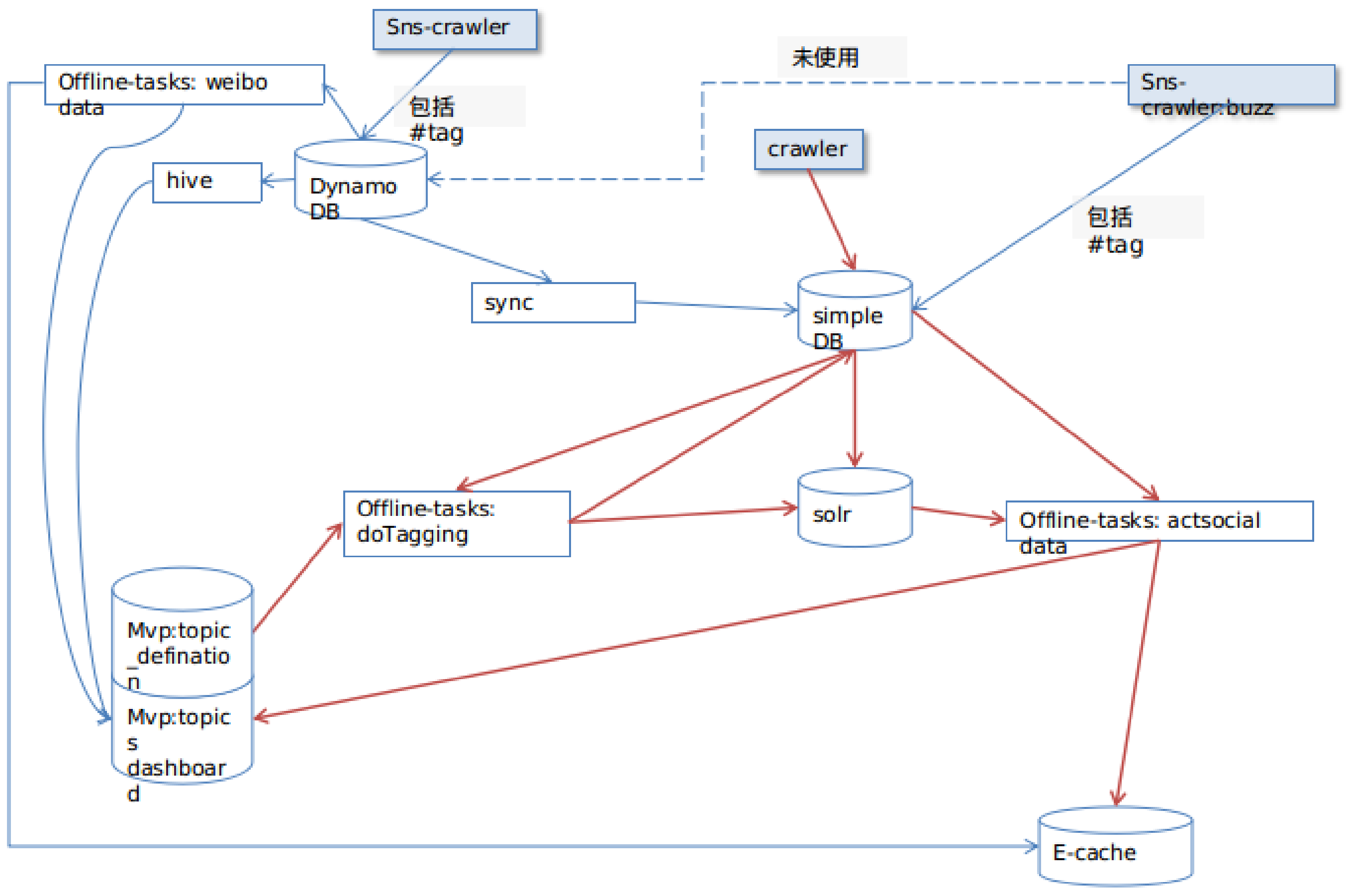


Figure C.1: Weibo SNS interactions between AWS & Offline tasks.

Appendix D

Crawling Technology Architecture

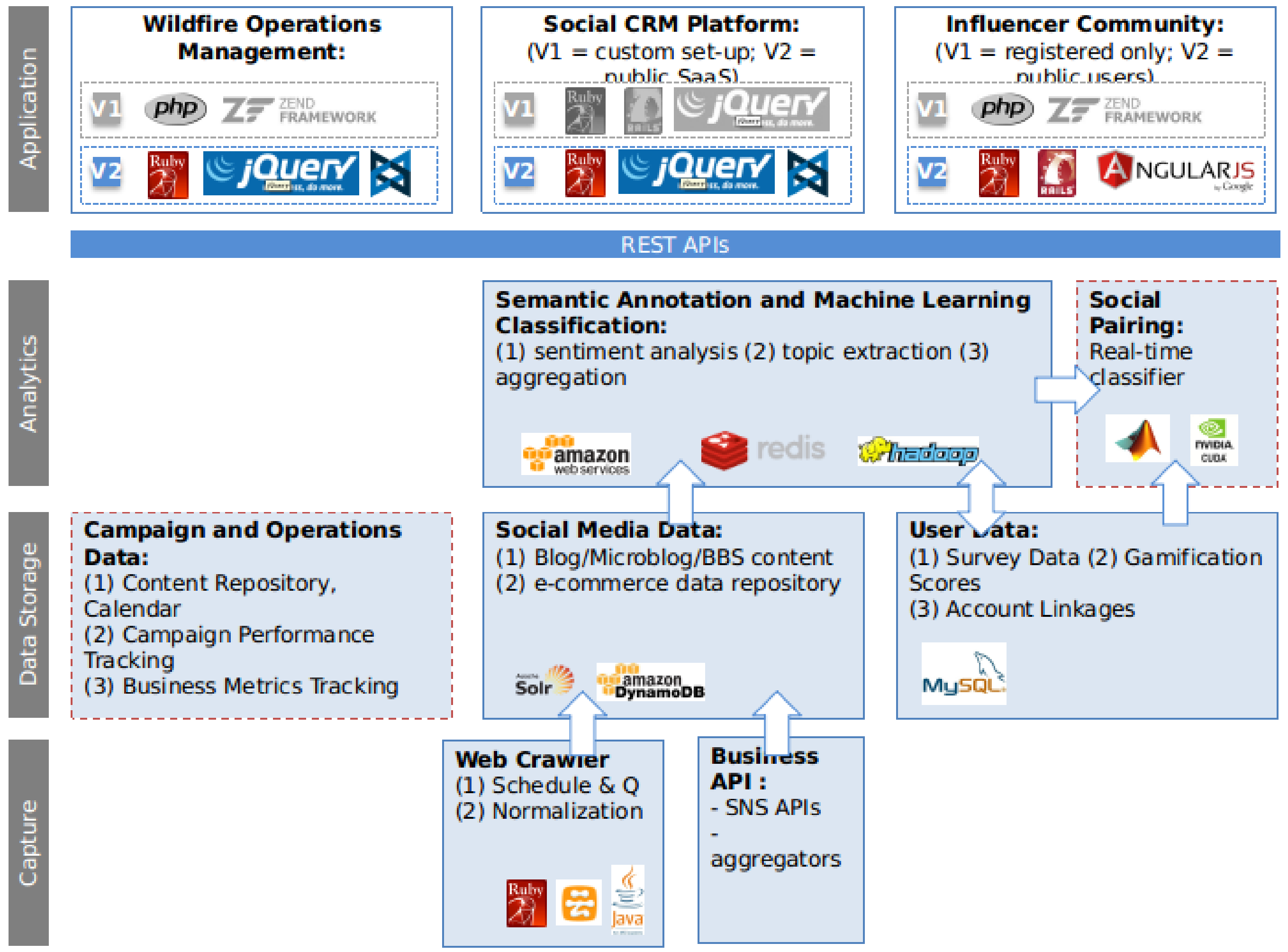

Application
Wildfire Operations Management:
V1
V2
Social CRM Platform:
(V1 = custom set-up; V2 = public SaaS)
Influencer Community:
(V1 = registered only; V2 = public users)
REST APIs
Analytics
Semantic Annotation and Machine Learning Classification:
(1) sentiment analysis (2) topic extraction (3) aggregation
Social Pairing:
Real-time classifier
Data Storage
Campaign and Operations Data:
(1) Content Repository, Calendar
(2) Campaign Performance Tracking
(3) Business Metrics Tracking
Social Media Data:
(1) Blog/Microblog/BBS content
(2) e-commerce data repository
User Data:
(1) Survey Data (2) Gamification Scores
(3) Account Linkages
Capture
Web Crawler
(1) Schedule & Q
(2) Normalization
Business API :
- SNS APIs
- aggregators

Figure D.1: High-level preliminary technology stack from data capture to representation.