Singapore Management University

Institutional Knowledge at Singapore Management University

Research Collection School Of Information Systems

School of Information Systems

12-2018

A cloud-based data gathering and processing system for intelligent demand forecasting

Colin K. L. TAY
Singapore Management University, colintay.2017@mais.smu.edu.sg

Kyong Jin SHIM
Singapore Management University, kjshim@smu.edu.sg

Citation

TAY, Colin K. L. and SHIM, Kyong Jin. A cloud-based data gathering and processing system for intelligent demand forecasting. (2018). *2018 IEEE International Conference on Big Data (Big Data): Seattle, WA, December 10-13: Proceedings*. 5451-5453. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/4338

2018 IEEE International Conference on Big Data (Big Data)

https://doi.org/10.1109/BigData.2018.8622546

A Cloud-Based Data Gathering and Processing System for Intelligent Demand Forecasting

Colin K. L. Tay
School of Information Systems
Singapore Management University
Singapore
colintay.2017@mais.smu.edu.sg

Kyong Jin Shim
School of Information Systems
Singapore Management University
Singapore
kjshim@smu.edu.sg

***Abstract*—Demand forecasting has been a challenging problem especially for products with short life cycles such as electronic goods and fashion items. Additionally, in the presence of limited past or historical data as well as the need for fast turnaround for forecast, producing timely and accurate demand forecast can be extremely challenging. In this study, we describe a cloud-based data gathering and processing system for intelligent demand forecasting.**

***Keywords*—demand forecasting, web crawling, cloud-based data gathering, data engineering**

I. INTRODUCTION & DATASET

Today's modern retail environments are highly dynamic and fast-paced. Books, fashion items, and electronic goods are examples of products with short product cycle [1, 2]. Demand forecasting for such products can be difficult due to limited historical demand and sales data on which forecasting models are built. Further, accurate demand forecasting requires not only the past trends but also data concerning external factors that a) are known to have influenced demand in the past such as weather, epidemics, incidents, and other economic indicators, and b) are predicted to influence future demand (e.g. new phenomena). In both cases, it is imperative that such information can be crawled dynamically and efficiently in order for retailers and also logistics and supply chain solution providers to strategize their business operations.

In this study, we propose and describe a cloud-based data gathering and processing system for intelligent demand forecasting. The proposed system allows for efficient and dynamic data crawling and information querying. The system crawls and scrapes external data that supplement historical demand and sales data towards more accurate demand forecasting. The system's architecture allows for scheduled job executions as well as on-demand data crawling.

II. SYSTEM ARCHITECTURE

The data crawlers, written in Python 3.6, crawl weather information, known public holidays globally, news articles (from a pre-defined list of news sites), and Twitter tweets. The latter two can be configured to crawl news articles and conversations pertaining to certain industries, sectors, brands, products, and individuals. The types of external data that can potentially help improve demand forecasting vary from one domain to another. For instance, in the past, reports of certain E. coli outbreak or other epidemics had a significant impact on consumer demand for certain food products or health products. The current architecture is highly modular – new crawler programs can be easily added onto the system by specifying new data sources, referencing new crawling scripts, and specifying crawling rules.

Our system leverages Celery Distributed Task Queue architecture [3]. The Scheduler allows for scheduled data crawling. Crawled data are stored in Elasticsearch [5] and it can be queried efficiently via a user interface by the business user. The current system prototype leverages Cloudera's Hadoop distribution. Our system architecture does not restrict users from using other external tools to interface with the crawled data for information querying and further data analyses.

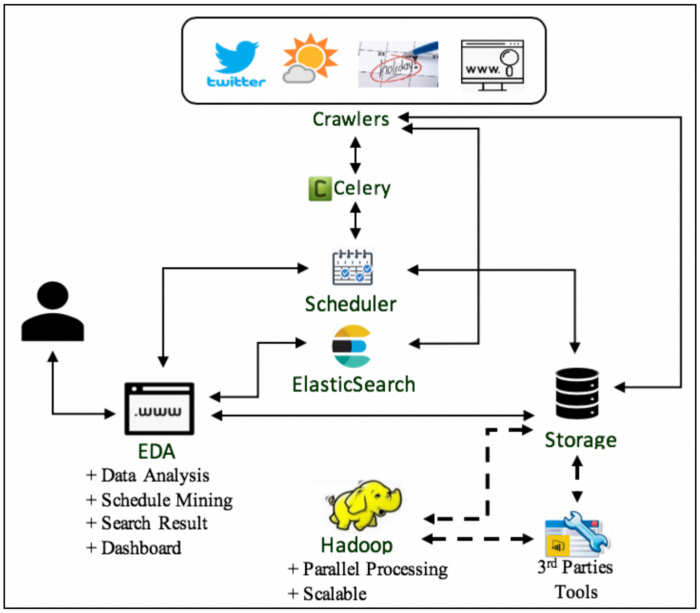


Fig. 1. Architecture of Proposed Cloud-Based Data Gathering and Processing System.

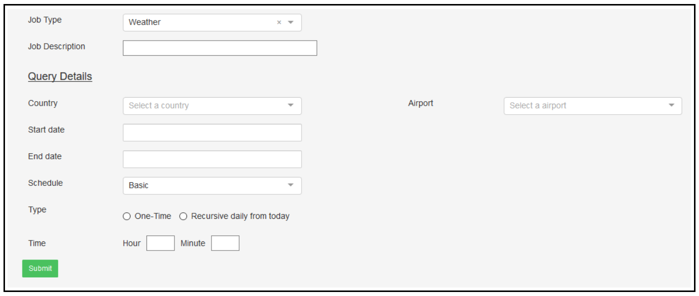
A. *Web Crawler Module*

In order to accommodate different types of data crawlers in the future, our system is designed to handle a heterogenous group of crawlers running concurrently. We achieve this through non-locking threads and non-dependence architecture. We use Celery Distributed Task Queue [3]. Four data crawlers currently in place in our system are shown in Table I.

TABLE I. DATA CRAWLERS AND DATA SOURCES

1	Data Type	Weather
	Source	Wunderground.com
	Metrics Collected	Daily mean temperature, minimum temperature, maximum temperature, weather condition (rainy, sunny, etc.)
	Additional Metrics	7 days weather forecast from the query date
	Mandatory Fields	Country, Airport ICAO code, Start and End date
2	Data Type	Public Holidays
	Source	TimeandDate.com
	Metrics Collected	Date, Day of the week, Holiday name, Holiday type, Country
	Additional Metrics	Nil
	Mandatory Fields	Country and year of interest
3	Data Type	News Articles
	Source	A pre-defined list of URLs
	Metrics Collected	An entire article that matches the keywords
	Additional Metrics	Nil
	Mandatory Fields	User-defined URL, keywords, depth and option to explore within the domain
4	Data Type	Social Media (Twitter)
	Source	Twitter
	Metrics Collected	Tweet ID, User, Created Datetime, Tweet text, Retweets count, Followers_Count
	Additional Metrics	Nil
	Mandatory Fields	Twitter username and keywords

All data crawlers can be configured using Basic Mode and Advanced Mode. The Basic Mode allows the user to schedule jobs on-demand – either as a one time task or as a recursive daily job.

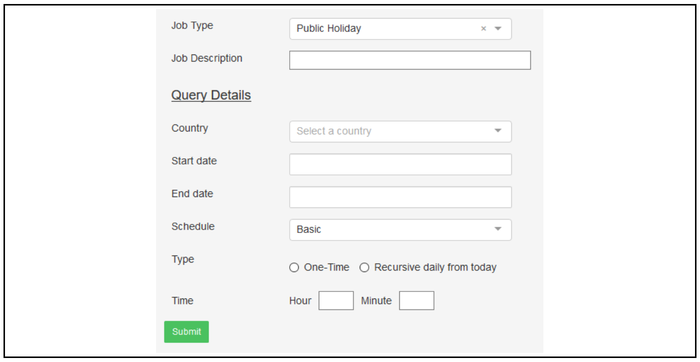


Job Type: Weather
Job Description
Query Details
Country: Select a country
Airport: Select a airport
Start date
End date
Schedule: Basic
Type: ○ One-Time ○ Recursive daily from today
Time: Hour Minute
Submit

Fig. 2. Weather Data Crawler for Airports

Figure 2 shows the web user interface for a Basic Mode weather data crawler for airports. The user can choose a particular airport from a pre-defined list of airports, and the weather data crawler will run, extract and show weather information on cities around the selected airport. This module uses Python's BeautifulSoup package to retrieve historical and forecast weather information from a public website. The user can further specify start and end dates, and the crawler will extract and display weather information around the specified dates.

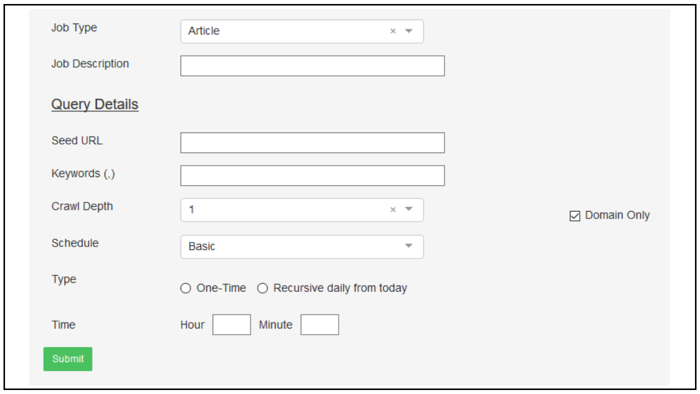
Another crawler, Public Holiday data crawler, extract global public holidays. Figure 3 illustrates various parameters that the user can specify for the crawler. Noting that public holidays are typically published annually ahead of time, our crawler extracts public holiday information by year since the number of records returned per year is relatively small.



Job Type: Public Holiday
Job Description
Query Details
Country: Select a country
Start date
End date
Schedule: Basic
Type: ○ One-Time ○ Recursive daily from today
Time: Hour Minute
Submit

Fig. 3. Public Holiday Data Crawler

News article data crawler utilizes Python's Newspaper package. This crawler takes a URL and a set of keywords specified by the user in the user interface shown in Figure 4, and it performs a recursive search within the specified domain. As many news sites and blogs have links to external websites such as advertisements, we incorporate a safeguarding mechanism whereby the crawler only retrieves and searches links belonging to the same domain as the original search domain. As the crawler extracts relevant webpages, it stores the search results in Elasticsearch.



Job Type: Article
Job Description
Query Details
Seed URL
Keywords (,)
Crawl Depth: 1
☑ Domain Only
Schedule: Basic
Type: ○ One-Time ○ Recursive daily from today
Time: Hour Minute
Submit

Fig. 4. Public Holiday Data Crawler

Lastly, our social media data crawler retrieves relevant tweets from Twitter via Twitter's Rest API and Streaming API. The user can specify trending keywords or hashtags in the user interface that are relevant to the user's industry or sector. The crawler will then retrieve recent tweets mentioning those keywords or hashtags. Similarly to other crawlers, the social media crawler can be scheduled to run on a regular basis or on demand.

B. *Scheduler & Task Distributor*

The scheduling feature of our system allows the user to configure crawlers' execution schedules in advance. We explored numerous options, ranging from OS commands such as 'cron' or 'at' to various development libraries such as Quartz, APScheduler and Celery. Our system uses both APScheduler and Celery libraries. APScheduler is capable of handling simple and complex schedules while Celery can be used as a task distributor to allocate tasks to crawlers.

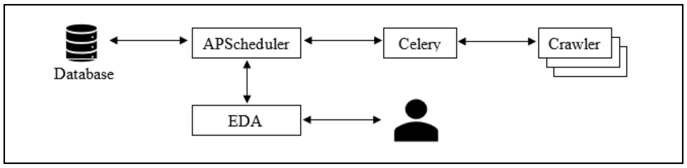


Fig. 5. Workflow between EDA, APScheduler, Celery and Crawler

As shown in Figure 5, APScheduler is backed by a database to persist the schedule information that the user defines. As such, there can be only one instance of the APScheduler running at a time to ensure data consistency. Our system also implements asynchronous tasking between EDA and APScheduler to prevent thread locking as the number of concurrent users increases. In the event when a task is to be scheduled, APScheduler will trigger a message to Celery where a cluster of crawlers will pick up the task to run and update the task status respectively thereafter.

Our system implements multiple asynchronous communication between modules. Celery is one of the most popular and established as compared to the rest such as RQ or TaskTiger. We use Celery as it is interoperable with the rest of the modules in our system and thereby minimizing system dependencies and most importantly, Celery can coordinate and distribute tasks across several machines.

C. *Search Engine*

Given that there is a significant amount of text content that needs to be stored, efficient search within these huge datasets with the traditional SQL queries can be challenging. As such, alternative technologies such as Apache Solr [4] and Elasticsearch [5] were explored to overcome the traditional database limitations. Upon further investigation, our system uses Elasticsearch due to the ease of deployment along with other modules. We chose Elasticsearch over Solr as the latter required an additional component (i.e. Zookeeper) to operate while Elasticsearch required only the standard installer. Furthermore, clustering can also be achieved easily with Elasticsearch with the standard installation on various computers in the same network, having the same cluster name. Next, Elasticsearch is schema-less meaning it does not require the definition of the domains (e.g. index, type, field, field type) in advance. Lastly, the Elasticsearch aggregation functions can extract interesting analytical insights (i.e. terms, min, max, avg, sum, cardinality, histogram) from the dataset, and it will be useful for Exploratory Data Analysis in alignment with demand forecasting.

D. *Storage & Hadoop*

Our system supports multiple storage options. The current prototype houses several MySQL and Microsoft SQL Server database instances. Nevertheless, the user can potentially use other databases in the future as the system architecture utilizes an Object-Relational Mapping (ORM) architecture where the user can switch to other databases such as Postgres, GreenPlum, Vertica and Impala without significant changes to the current system architecture.

We explored three popular commercial distributions of Hadoop (i.e. Cloudera, HortonWorks, and MapR) and chose Cloudera in accordance with several other projects in our organization looking to leverage cloud-based platform for big data analytics. All of the system modules were deployed onto a Cloudera Hadoop platform and configured accordingly.

E. *Intelligent Demand Forecasting*

When it comes to forecasting future demand, historical data may prove to be insufficient especially in the presence of volatile market dynamics. The primary objective of this study is to provide a cloud-based one stop portal for the investigation of external factors that influence demand volatility. Our cloud-based data gathering and processing system crawls, scrapes, and stores external data that can supplement historical demand and sales data towards more accurate future demand forecasting. For instance, a series of bad weather days combined with a planned road construction in a particular geographic area has been found to be positively correlated with a sudden surge of demand for particular motor parts.

III. FUTURE DIRECTIONS

We are continuing to add more data sources that can potentially affect global markets. We plan on gathering more data pertaining to natural disasters, accidents, crimes, etc. We plan to continue to find correlation between external factors as found in the crawled data and past and future demand across multiple industries and sectors.

REFERENCES

[1] Maaß, Dennis & Spruit, Marco & de Waal, Peter. (2014). Improving short-term demand forecasting for short-lifecycle consumer products with data mining techniques. Decision Analytics. 1. 4. 10.1186/2193-8636-1-4.

[2] Top Challenges in Demand Forecating. (2018 November 11). Retrieved from https://www.relexsolutions.com/top-challenges-in-demand-forecasting/

[3] Celery: Distributed Task Queue. http://www.celeryproject.org/

[4] Solr: http://lucene.apache.org/solr/

[5] Elasticsearch: https://www.elastic.co/