Towards a Scalable Enterprise Content Analytics Platform

Kevin Beyer Vuk Ercegovac Rajasekar Krishnamurthy Sriram Raghavan Jun Rao Frederick Reiss
Eugene J. Shekita David Simmen
Sandeep Tata Shivakumar Vaithyanathan Huaiyu Zhu

{*kbeyer, vercego, rajase, rsriram, junrao, frreiss, shekita, simmen, stata, vaithyan, huaiyu*}*@us.ibm.com*

IBM Almaden Research Center
650 Harry Road, San Jose
California, 95120, USA

Abstract

With the tremendous growth in the volume of semi-structured and unstructured content within enterprises (e.g., email archives, customer support databases, etc.), there is increasing interest in harnessing this content to power search and business intelligence applications. Traditional enterprise infrastruture or analytics is geared towards analytics on structured data (in support of OLAP-driven reporting and analysis) and is not designed to meet the demands of large-scale compute-intensive analytics over semi-structured content. At the IBM Almaden Research Center, we are developing an "enterprise content analytics platform" that leverages the Hadoop map-reduce framework to support this emerging class of analytic workloads. Two core components of this platform are SystemT, a high-performance rule-based information extraction engine, and Jaql, a declarative language for expressing transformations over semi-structured data. In this paper, we present our overall vision of the platform, describe how SystemT and Jaql fit into this vision, and briefly describe some of the other components that are under active development.

1 Introduction

As the volume of semi-structured and unstructured content within enterprises continues to grow, there is increasing interest and commercial value in harnessing this content to power the next generation of search and business intelligence applications. Some examples of enterprise repositories with valuable unstructured content include email archives, call center transcripts, customer feedback databases, enterprise intranets, and collaboration and document-management systems.

The use of content from such repositories for enterprise applications is predicated on the ability to perform analytics. For example, transcripts of customer calls can yield valuable business insights in areas such as product perception, customer sentiment, and customer support effectiveness. However, analytics is essential to extract the appropriate information from the raw text of the transcripts and transform this information into a form

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

that can be consumed by BI analysis and reporting applications. As another example, increased legislation around corporate data governance is requiring enterprises to invest in applications for regulatory compliance and legal discovery [5]. Such applications require advanced analytics (such as automatic recognition of persons, organizations, addresses, etc.) to support search and retrieval over enormous email archives. Finally, as we describe in Section 3, sophisticated analytics on intranet Web pages is critical to effective search over complex enterprise intranets.

Notice that in each of these applications, the common underlying theme is the need to perform analytics on large amounts of unstructured content. For instance, email archives can range in size from a few tens to several hundreds of terabytes, depending on the size of the company and the applicable regulations. Similarly, we know of close to 100M distinct URLs within the IBM intranet, which translates to approx. 3TB of plain HTML content. Factoring in non-HTML content as well as the data stored in numerous enterprise content management systems will result in a couple of orders of magnitude increase in size.

Traditional enterprise infrastructure for analytics (ETL engines, warehouses and marts, data cubes, etc.) is geared towards OLAP-driven reporting over structured data and is not designed to meet the demands of large-scale analytics over unstructured content. To this end, at the IBM Almaden Research Center, we are developing an *enterprise content analytics platform* to support muti-stage analytic workflows over large volumes of unstructured and semi-structured content. In building this platform, we are leveraging the tremendous innovation in the industry around scale-out based data processing paradigms – in particular, the open source Hadoop implementation of the Map/Reduce framework pioneered by Google.

The design goals for our platform are motivated by the following two observations:

1. *Compute intensive information extraction:* A common aspect of analytics over unstructured content is the need for *information extraction* – to identify and extract structured data ("annotations") from text. For example, in the ES2 intranet search application described in Section 3, several hundred patterns involving regular expressions and dictionary matches are applied on the titles, URLs, and other features of intranet Web pages to extract high quality annotations for a search index. In general, information extraction is a compute intensive process involving several complex character-level operations such as tokenization and pattern matching. The ability to support scalable information extraction over large content repositories is a key design goal for our platform.

2. *Dynamic and evolving workflows:* Due to the inherent heterogeneous nature of text collections, analytic workflows will need to continuously evolve over time to adapt to changes in the incoming content. For instance, when documents in newer formats or languages are added to a content source, appropriate modifications to the analytic workflow will be needed to incorporate new parsers, introduce language-specific tokenizers, and appropriately modify information extraction rule patterns. As a sample data point, the analytic workflow that powers the ES2 intranet search application (cf. Section 3) has gone through hundreds of changes to its processing workflow over the past year, in response to changes in the type and nature of content being published to the IBM intranet.

In this paper, we present a high-level architecture of our platform and describe some of the components that are currently under active development.

2 Overview of the Platform

The enterprise content analytics platform is designed to facilitate the specification and evaluation of complex analytic data flows over massive volumes (terabytes to petabytes) of content from enterprise repositories. We are developing new data processing tools for this purpose: SystemT [9] to extract information from content and Jaql [8] to flexibly specify analysis workflows. Both tools are used declaratively to insulate the user from optimization decisions needed for efficient and scalable processing.

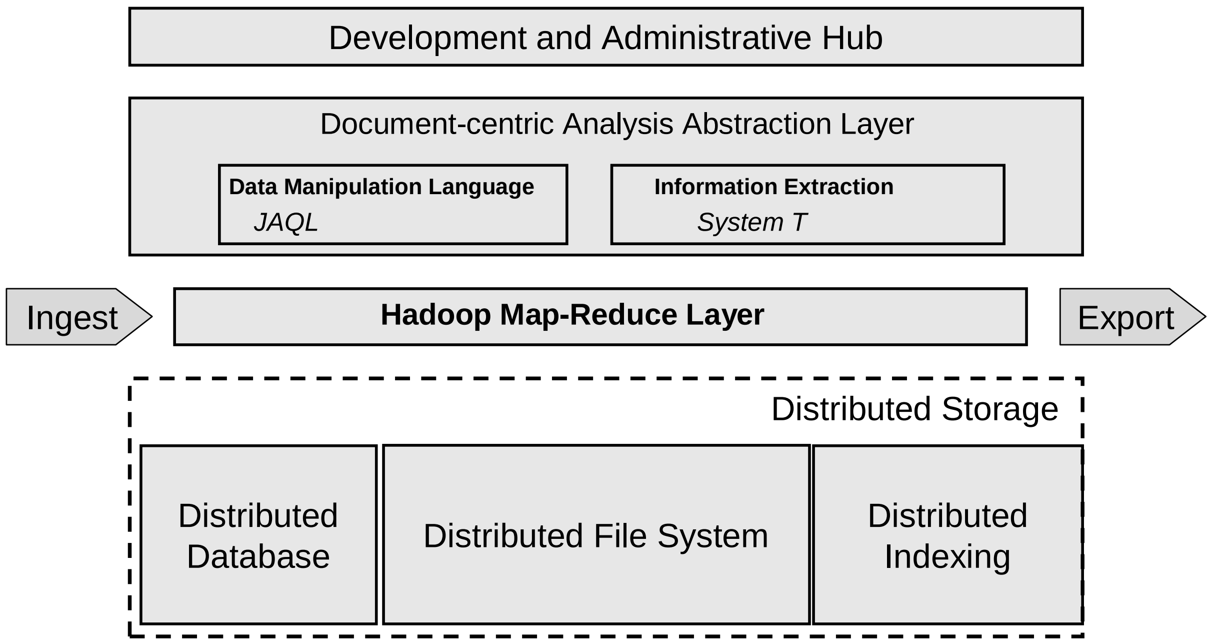


Figure 1: Architecture

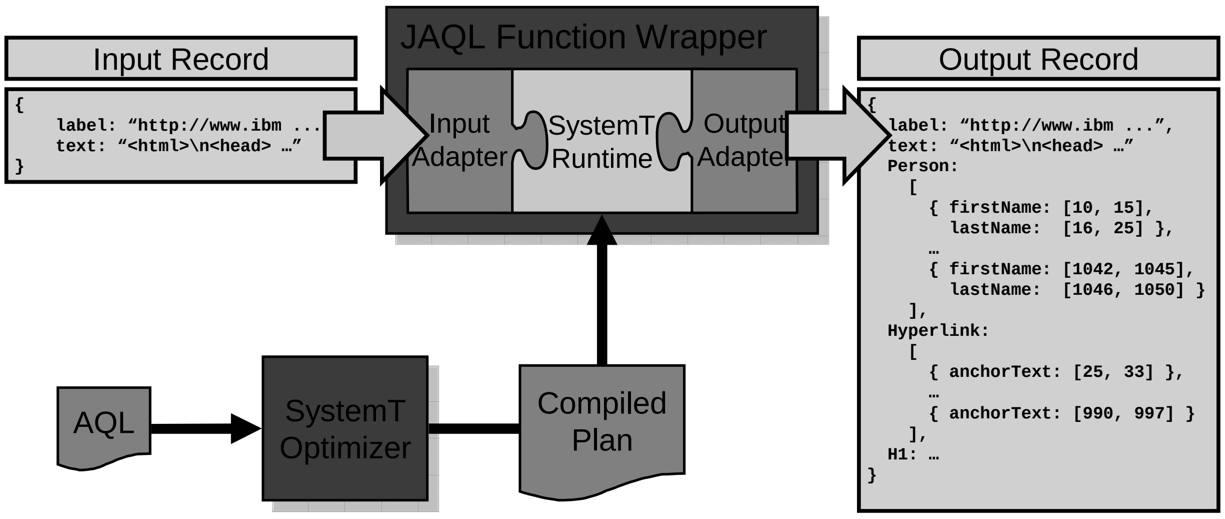
In addition to SystemT and Jaql, the content analytics platform includes the additional tools shown in Figure 1. Prior to analysis, content is ingested and written to distributed storage services. For example, the semantic search application described in Section 3 uses Nutch [1], a distributed crawler based on Hadoop, to ingest data. In addition, we plan to ingest data in enterprise content management systems like FileNet, Documentum, and OpenText into the platform using special load APIs.

For scalable storage and processing, we have bootstrapped the content analytics platform using the Apache Hadoop [6] project's HDFS, a distributed file system, and map-reduce, a parallel programming framework popularized by Google [4]. Files provide the platform with scan-based access and map-reduce is used to implement parallel aggregations and joins by re-partitioning data across a cluster. In Section 4, we describe our active research and development efforts to extend the available storage services with a distributed database and index as well as extend map-reduce to incorporate query processing techniques from the database literature.

Following analysis, the results are transformed and exported according to application requirements. For example, a search application requires inverted indexes to be built. For those search applications that further expose structured information discovered during extraction, exporting to a relational database is appropriate. For such cases, we plan to support customizable export tools to accommodate varying application requirements.

While SystemT and Jaql are the core tools used for analysis tasks, we envision higher-level abstractions and tooling to assist users in developing analytic workflows. For content analytics, a *document* is the unit of analysis and therefore we are designing many of the operators (e.g., SystemT) to be document-centric. With regard to tooling, we plan to build a set of services and GUI to assist with analytics development, evaluation, and administration.

The current focus at the Almaden Research Center is on SystemT and Jaql. We now describe these in more detail.



```
{
    label: “http://www.ibm ...
    text: “<html>\n<head> …”
}
```

```
{
  label: “http://www.ibm ...”,
  text: “<html>\n<head> …”
  Person:
    [
      { firstName: [10, 15],
        lastName:  [16, 25] },
      …
      { firstName: [1042, 1045],
        lastName:  [1046, 1050] }
    ],
  Hyperlink:
    [
      { anchorText: [25, 33] },
      …
      { anchorText: [990, 997] }
    ],
  H1: …
}
```

Figure 2: SystemT invoked from Jaql

2.1 Information Extraction using SystemT

SystemT is a system for rule-based information extraction that has been under development at IBM Almaden Research Center since 2006. SystemT is used to extract structured information from unstructured text, finding, for example, project home pages in corporate intranet web pages or person-phone number relationships in email messages. At the core of SystemT is a declarative rule language, AQL, for building extractors, and an optimizer that compiles these extractors for an algebra-based execution engine.

SystemT scales to massive document corpora by extracting information from multiple documents in parallel. The current version of the system supports two approaches to parallel scaleout: Direct embedding in a map-reduce job, and parallel execution as part of a Jaql query.

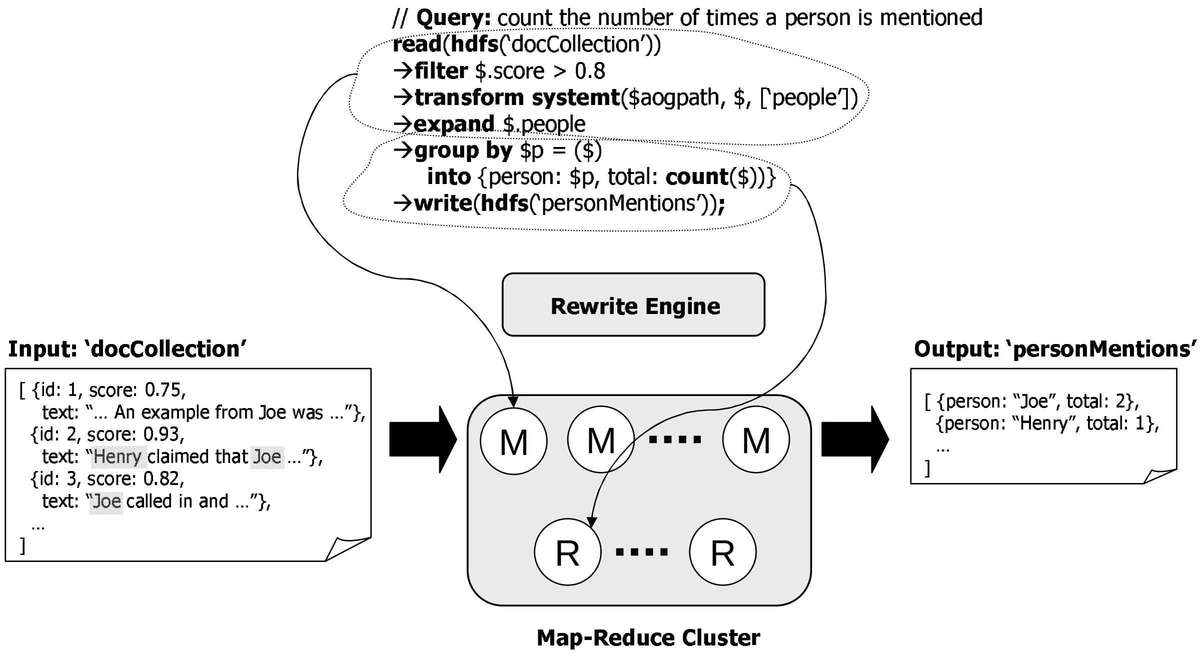
The direct embedding scaleout approach encapsulates the SystemT engine in a single “map” stage of a Hadoop map-reduce flow. The input to each mapper is a stream of documents, and the output of the mapper augments these documents with the structured information the system extracts from them. SystemT provides a layer of input/output adapters that support variety of input and output formats.

SystemT can also use Jaql’s automatic parallelization to enable scalable information extraction (see Figure 2). SystemT’s Jaql integration code encapsulates the information extraction runtime as a Jaql function. This function maps a record containing a document to an augmented record containing the document plus extracted structure. Jaql handles the mapping of the SystemT function call into a map-reduce framework, possibly executing a SystemT annotator and several other operations in a single map job.

2.2 Data Processing using Jaql

For flexibility during the ingestion, analysis, transformation, and exporting of data, we require a lightweight description language that supports semi-structured data and is easy to extend with new operators as well as sources and sinks of data. For this purpose, we are designing Jaql, a general purpose data-flow language that manipulates semi-structured information in the form of abstract JSON values.

JSON consists of atomic values like numbers and strings and two container types: arrays and records of name-value pairs (sometimes called maps or hashes). The values in a container are arbitrary JSON values; for example, arrays can have different types in each element and an element could itself be another array. The JSON



```
// Query: count the number of times a person is mentioned
read(hdfs('docCollection'))
→filter $.score > 0.8
→transform systemt($aogpath, $, ['people'])
→expand $.people
→group by $p = ($)
    into {person: $p, total: count($)}
→write(hdfs('personMentions'));
```

```
[ {id: 1, score: 0.75,
    text: "… An example from Joe was …"},
  {id: 2, score: 0.93,
    text: "Henry claimed that Joe …"},
  {id: 3, score: 0.82,
    text: "Joe called in and …"},
  …
]
```

```
[ {person: "Joe", total: 2},
  {person: "Henry", total: 1},
  …
]
```

Figure 3: Jaql query compiled to map-reduce

model provides easy migration of data to and from most popular scripting languages like Javascript, Python, Perl, and Ruby because these languages all directly support dynamic arrays and records; other programming languages also have support for these constructs in their standard libraries. Therefore, Jaql is easily extended with operators written in most programming languages because JSON has a much lower impedance mismatch than say XML, yet much richer than relational tables.

Jaql provides a framework for reading and writing data in custom formats that is used while ingesting and exporting. We do not expect large volumes of JSON data to be stored on disk, but most data, like comma-separated files (CSVs), relational tables, or XML have a natural interpretation as JSON values. Unlike traditional database systems, Jaql processes data in its original format; the user is not required to load it into some internal form before manipulating it.

Jaql provides support for common input/output formats like CSVs, as well as many common operators including: filtering, transforming, sorting, grouping, aggregating, joining, merging, distincting, expanding nested data, and top-k. By composing these simple operators plus calls to custom operators into a rich data flow, the user expresses complex analyses and transforms the data to produce the exported data. Jaql compiles an entire flow of these operators into a graph of Map/Reduce jobs for Hadoop to process.

Figure 3 shows an example Jaql query which computes the number of times each person is mentioned in a given document collection. Using SystemT to detect mentions of a person's name, Jaql computes this count using the standard operator palette and produces the output in a JSON array.

3 Example Application: Semantic Search

One of the driving applications of the analytics platform at IBM is ES2, a semantic search engine for the enterprise. ES2 leverages the content analytics platform and uses sophisticated analytics along with an intelligent index-building strategy to provide high-precision results for navigational queries [10]. ES2 uses Nutch [1] as its ingest mechanism to crawl the pages on the IBM intranet. The pages crawled by Nutch are stored in HDFS and are available for processing using System T and Jaql. In this section, we briefly describe how System T and Jaql form critical building blocks for the analytics in a system like ES2.

Analytics The analysis in ES2 consists of two distinct types: local analysis which processes a single document at a time and global analysis which operates over the data extracted from the entire collection. The information obtained from these analyses is used as an input in the indexing phase. Together, local and global analyses allow us to reason about the document collection better and bring back significantly more precise results [10] than would be possible using traditional IR strategies such as *tf-idf* and *PageRank* [3].

In Local analysis each page is individually analyzed to extract clues that help decide whether that page is a candidate navigational page. In ES2, four different local analysis algorithms namely *TitleHomePage*, *PersonalHomePage*, *URLHomePage*, *AnchorHome* and *NavLink* are used to extract certain features from the pages. These algorithms use rules based on regular expression patterns, dictionaries, and information extraction tools [9] to identify candidate navigational pages. For instance, using a regular expression like “\ A\ W*(.+)\s<Home>” (Java regular expression syntax), the *PersonalHomePage* algorithm can detect that a page with a title “G. J. Chaitin’s Home” indicates that this is the home page of G. J. Chaitin. The algorithm outputs the name of a feature (“Personal Home Page”) and associates a value with this feature (“G. J. Chaitin”). Readers interested in more details about local analysis algorithms may refer to [10]. These analysis tasks are expressed using AQL and are optimized and executed using SystemT.

Note that multiple pages in the collection may produce the same extracted feature during local analysis. Consider the case where homepage authors use the same title for many of their webpages. Continuing the example from the previous paragraph, “GJ Chaitin home page” is the title for many of the pages on GJ Chaitin’s website. Local analysis for personal homepages considers all such pages to be candidates. ES2 uses global analysis to determine an appropriate subset of pages as navigational and indexes them appropriately. [10] describes two algorithms: *site root analysis* and *anchor text analysis*. In ES2, we express these algorithms using Jaql which automatically compiles into map-reduce jobs that execute over the entire cluster.

Smart Indexing ES2 employs a collection of algorithms to intelligently generate variants of terms extracted during local analysis and global analysis before inserting the document in the index. Consider an example where ES2 identifies a page as the home page of a person named “G J Chaitin” using local and global analysis. During variant generation, a special set of rules is applied to such person names to enumerate other syntactic variants (e.g., skipping middle initials, merging multiple initials, only listing the last name, etc.). By inserting such variants into the index, ES2 can match the search term “GJ Chaitin” with the given page, despite the lack of space between “G” and “J”. Note that generating variants by skipping white space, when applied to arbitrary pieces of text, is likely to yield noisy search results. We only apply this approach to pages produced through a specific analysis workflow – in this case one that uses results from *PersonalHomePage* local analysis and *site root* global analysis. Indexing is done in a distributed fashion by leveraging map-reduce. Jaql is used to transform the outputs of different analytic tasks and produce appropriate search indexes over the extracted values. The output from this workflow is typically a set of navigational indexes that are then copied over to a separate set of machines to serve queries.

4 Future Challenges

While SystemT and Jaql constitute a majority of our current effort, we are also investigating other components of the platform including the distributed runtime, distributed storage layer, and the user-interaction hub. We broadly outline the challenges in these areas.

4.1 Enhancing Map-reduce Runtime

The map-reduce paradigm was popularized by the distributed system community. Compared with a database system, map-reduce based data processing platforms have better support for fault-tolerance, elasticity, and load balancing. However, many computations in the two systems are quite similar. We are investigating techniques that bridge the gap between the two systems, by applying what the database community has learned over the last three decades to map-reduce. One of our areas of focus is to improve join processing in map-reduce.

Although map-reduce was originally designed to process a single collection of data, many analytic applications require joining multiple data sources together. Here is a straightforward way of mapping a join operation into map-reduce: the map function iterates over the records from both input sources. For each record, it extracts the join key as the output key, tags each record with the originating source, and saves it as the output value. Records from both sources for a given join key are eventually merged together and fed to a reduce task. The reduce function first separates the input records into two sets according to the tag, and then performs a cross-product between records in those sets. This implementation is similar to a repartitioned sort-merge join in a database system. This algorithm incurs a significant overhead since all input records have to be sorted and most of them have to be sent across the network.

For a long time, the database community has been exploiting hash-based joins to avoid sorting and broadcasting joins to avoid the communication overhead. Leveraging those ideas, an alternative approach is to do the join in a map-only job. At the beginning of each map task, we load the smaller input source into a hash table. The map function then iterates through records from the larger input source, and for each record, probes the hash table to do the join. This approach avoids sorting and moving data from the larger input source. Our experimental results show that it can reduce the time taken by the straightforward approach by up to 70%. As the smaller input source gets larger, the map-only job becomes less efficient since more data has to be broadcasted to every node. For certain applications, we find that semi-join techniques can be used to improve the performance further. We are preparing a research paper to summarize those results in detail. We are also investigating techniques to extend Jaql compiler to automatically select the best join strategy among the many available.

4.2 Distributed Content Database

Different phases of an analytics workflow tend to have different database requirements. For example, during ingestion, new documents are incrementally inserted into the database. Then during analysis, documents are often processed in batch mode as part of one or more map-reduce jobs. Finally, in applications like search, users often want to interactively look at documents returned by search queries. A key question is whether one content-oriented database is sufficient to satisfy all these requirements. With relational databases, it is common to maintain at least two databases, one for warehouse applications and another for interactive applications. We suspect that much the same will happen here. This is because content analytics tend to be very resource heavy, making it difficult to support interactive applications on the same database without running into performance problems.

Assuming separate content databases are maintained for analytics and interactive applications, then we think it will be interesting to explore different database architectures. One architecture would be tuned for batch analytics, while the other would be tuned for interactive applications. Generally speaking, interactive applications present a bigger design challenge in a distributed environment, especially if support for transactions are included. This is because of the well known tension between availability and consistency [2]. In contrast, analytics applications tend to work on a stable collection of documents, where consistency and availability tradeoffs are not an issue. Content-oriented databases without support for high-performance transactions like HBase [7] leverage the underlying distributed filesystem.

4.3 Development and Administrative Hub

We plan to build a development and administrative hub that facilitates the formation of user communities and management of resources on the platform. The hub would provide services for managing users (user services); for launching, monitoring, and diagnosing content analytics jobs (job services); for uploading and cataloging assets such as content analytic flows, data sources, data sinks, sandboxes, annotators, and user functions (directory services); and for performing miscellaneous tasks such as sandbox creation, load/unload of data to/from the cluster, and collection of data distribution statistics used to optimize analytic flows (utility services).

Providing a design interface that allows workflows to be created iteratively, and interactively, in the context of a ”sandbox” is one of the challenges being tackled. A sandbox includes representative cluster configuration information, as well as representative samples from relevant data sources. We are also examining tools that would assist in the collaborative development of analytic components.

5 Summary

With increasing interest from enterprises to harness the value in their structured and semi-structured content, we believe that there is a rapidly emerging need for an enterprise content analytics platform. We identify two key features that are needed from such a platform: 1) the ability to perform compute intensive information extraction, and 2) build and maintain evolving workflows that process large amounts of data. We describe the efforts underway at IBM’s Almaden Research Center to address these requirements by way of SystemT and Jaql. In addition, we also laid out the broad challenges that lay ahead in building a dynamic, scalable, high-performance, and usable content analytics platform for enterprises.

References

[1] Apache Foundation. Nutch. http://lucene.apache.org/nutch/.

[2] Eric Brewer. Keynote speech: Towards robust distributed systems. In *PODC*, 2000.

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107 – 117, 1998. Proceedings of the Seventh International World Wide Web Conference.

[4] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[5] Electronic Discovery. Electronic discovery. http://www.discoveryresources.org/.

[6] Apache Foundation. Hadoop. http://hadoop.apache.org/core/.

[7] Apache Foundation. Hbase. hadoop.apache.org/hbase.

[8] JAQL. Jaql. http://code.google.com/p/jaql/.

[9] Frederick Reiss, Sriram Raghavan, Rajasekar Krishnamurthy, Huaiyu Zhu, and Shivakumar Vaithyanathan. An algebraic approach to rule-based information extraction. In *ICDE*, pages 933–942, 2008.

[10] Huaiyu Zhu, Sriram Raghavan, Shivakumar Vaithyanathan, and Alexander Löser. Navigating the intranet with high precision. In *WWW*, pages 491–500, 2007.