DATA MINING FROM DOCUMENT-APPEND NoSQL

[Richard K. Lomotey, Ralph Deters]
[University of Saskatchewan, Saskatoon, Canada]
[richard.lomotey@usask.ca, deters@cs.usask.ca]

Abstract
Due to the unstructured nature of modern digital data, NoSQL storages have been adopted by some enterprises as the preferred storage facility. NoSQL storages can store schema-oriented, semi-structured, schema-less data. A type of NoSQL storage is the document-append storage (e.g., CouchDB and Mongo) which has received high adoption due to its flexibility to store JSON-based data and files as attachment. However, the ability to perform data mining tasks from such storages remains a challenge and the required tools are generally lacking. Even though there is growing interest in textual data mining, there is huge gap in the engineering solutions that can be applied to document-append storage sources. In this work, we propose a data mining tool for term association detection. The flexibility of our proposed tool is the ability to perform data mining tasks from the document-source directly via HTTP without any copying or formatting of the existing JSON data. We adapt the Kalman filter algorithm to accomplish macro tasks such as topic extraction, term organization, term classification and term clustering. The work is evaluated in comparison with existing textual mining tools such as Apache Mahout and R with promissory result on term extraction accuracy.

Keywords: Data mining; NoSQL; Kalman filter; Unstructured data; Big data; Association rule

1. INTRODUCTION
There is huge amount of digital data today that can be used to achieve several goals. This data economy is referred to as “Big data” (Zikopoulos et al., 2012). The high-dimensional data comes in heterogeneous formats and at high streaming velocity. This situation has rendered existing Relational Database Management Systems (RDBMS) inefficient especially at the storage level. While RDBMS systems are designed to store schema-based data in certain predefined formats, the data being generated nowadays have no schema. The greater portion of the generated digital content has no particular structure and the data comes in several formats such as multimedia, files, data, and so on. Thus, the NoSQL (NoSQL, http://nosql-database.org) storages have been proposed as a measure to accommodate the growing style of data in unstructured formats. Different types of NoSQL storages exist and some examples are multimodal databases, graph databases, object databases, and so on.

In this work, we shall focus on the document-append style NoSQL storage. This style of storage supports structured, semi-structured, and unstructured data. The testing environment will be CouchDB which supports upload of files as attachments and actual data in JSON format. Identical to Web services, CouchDB can be interacted with using the HTTP verbs following the REpresentational State Transfer (REST) style such as GET – to retrieve data, POST – to create new records, PUT – to update an existing data, DELETE – to remove an existing record, and HEAD - to retrieve the meta-info of a stored data. Further, the data stored is treated as a resource and have unique identifiers per document.

The problem however is that, we are faced with the challenge of performing data mining tasks from such storage facilities. Even though the interaction with such data storage is facilitated as Web service (and can be accessed via the browser), we are not able to employ tools such as Web crawlers to mine data from them. This is because, crawlers rely on the HTML tags within the DOM of a webpage to retrieve information. The presence of anchor and href tags (i.e., <a href=””></a>) on a page enables the crawler to traverse all hyperlinks that are found within the search space. In the case of document-append storages, there are no such tags; rather, the data is JSON or XML per document. Also, while RDBMS storages have a structure and support cross-database queries based on keys, the document-append storage has no keys and relies on map reduce. This situation has rendered data mining tools that have been built for RDBMS and web crawlers inefficient in the world of document-append storages (Fig. 1 compares the two).

In this paper, we propose a data mining tool that can be employed to perform analytics of terms and topics from the document-append storage using CouchDB as case study. Users are enabled to perform data mining tasks directly from CouchDB via HTTP requests without any copying or formatting of the existing JSON data. We propose the Kalman filter algorithm to achieve requirements such as topic extraction, term organization, term classification and term clustering. The work is evaluated in comparison with existing textual mining tools such as Apache Mahout and R with promissory result on term extraction accuracy.
Here are the highlights of this work:

- Proposed a data mining framework for Document-Style NoSQL
- Proposed the Kalman filter algorithm to determine the association between topics in the data source
- The framework supports topic extraction, term organization, term classification, and term clustering
- In comparison to Apache Mahout and R, the proposed framework is:
  - more accurate
  - more flexible to adapt to other data sources
  - more scalable to accommodate several data sources
  - less scalable than Apache Mahout in terms of the processing job per time.

The remaining sections of this work are structured as follows. Section II reviews some works on unstructured data mining. Section III describes our proposed search tool, Section IV details the adapted Kalman filter algorithm and the preliminary evaluation of the proposed tool is carried out in Section V. The paper concludes in Section VI with our findings and future work.

2. BACKGROUND WORKS

2.1 NoSQL Databases

Today, digital content generation is influenced by multiple stakeholders such as: end-users, enterprise consumers, services providers, and prosumers. This has also fueled the nature of the data which has shifted from schema-based storages to semi-structured and unstructured data (Rupanagunta et al., 2012). In summary, the concept of unstructured data is described as:

- **Data Heterogeneity**: The data is represented in varying formats (documents, multimedia, emails, blogs, websites, textual contents, etc.)
- **Schema-less**: The data has no specific structure because there is no standardization or constraints placed on content generation.
- **Multiple Sources**: The data is from diverse sources (e.g., social media, data providers such as Salesforce.com, mobile apps, sensors, etc.)
- **Varying API Types**: The data requires multiple APIs to aggregate data from the multiple sources, and the APIs are not standardized nor have SLAs.

While the Web has been at the forefront of content accessibility and delivery, it is not a platform for storage. Thus, a revolution in storage has been proposed that shift focus from RDBMS storages to NoSQL databases. There are varying forms of NoSQL storages such as: document-appoint NoSQL that supports textual and file attachments (e.g., CouchDB), File-based storages (e.g., Dropbox, MEGA, Amazon S3), Wide-Column Storages (e.g., Hadoop, HBase), Graph-based storages (Neo4J, InfoGrid), and so on. While many of the ongoing works on big data and unstructured data mining has targeted Hadoop and its family line of storage, not many works is seen on the document storage front. As of the time of writing, we could not find any tool that is aimed at the document storage style. Our work will therefore explore this option where CouchDB will be our use case environment. The storage allows data to be stored in a JSON format and does not use keys as in the case of RDBMS.

Further, each document is uniquely identified by an identifier which can be used to access the document over a hyperlink. In order to query multiple documents, the MapReduce methodology can be adopted.

Table I. Text-Based Data Mining Requirements and Proposed Techniques
While the underlying technology on which document-style database are built follows the Web standard, performing data mining will take different dimensions. Web data mining is done using the Web crawler which relies on the HTML tags to interpret data blocks (i.e., content) on a web page. The anchor tag and href attributes also aids the crawler to go to other interconnected pages. The document-style storage however does not have tags; rather, its text-based that follows the JSON. Moreover, RDBMS specific data mining tools are not designed for textual mining and storage facility such as CouchDB that allows file attachments further complicates the issue. This creates the need to research on how to perform data mining and knowledge discovery from such storage. Thus, our major goal in this paper is to research and design a data analytics tool that targets the document-style NoSQL. However, to achieve this goal, we need to explore the second major aspect of the work, which is textual and unstructured data mining.

### 2.2 Unstructured Data Mining

Without data mining, we cannot deduce any meaning out of the vast data at our disposal. As already posited, existing data mining techniques that are well-advanced have been designed to work either with schema-oriented databases which are structured (Kuonen, 2003) or Web crawlers. To enhance the data mining process, scientist in both academia and industry are beginning to explore the best methodologies that can aid in the unstructured data mining process; especially in the context of textual data mining. As a result, we have witnessed some requirements such as: Information Retrieval (IR) algorithms based on templates (Hsu and Yih, 1999), Association Rules (Delgado et al., 2002, Zhao and Bhumick, 2003), Topic tracking and topic maps (Abramowicz et al., 2003), Term crawling (terms) (Janet and Reddy, 2011, Feldman et al., 1998), Document Clustering (Han et al., 2010), Document Summarization (Dey and Haque, 2009), Helmholtz Search Principle (Balinskiy et al., 2010), Re-usable Dictionaries (Godbole et al., 2010), and so on. For brevity, the reader is referred to Table I to see the overview of the area including unexplored areas.
There is an ever growing demand for the consumption of high-dimensional data across different enterprises such as Biomedical Engineering, Telecommunications, Geospatial Data, Climate Data and the Earth’s Ecosystems, and Capital Research and Risk Management (Groth and Muntermann, 2010). This has resulted into the employment of some or the combination of the various requirements described in Table I to deploy unstructured data mining tools. Though the application of unstructured data mining tools is broad and some previous works including Scharber (2007) focus on the available tools in general textual mining, we can identify with the fact that the deployment of these tools require the active participation of software engineers as well as software engineering principles.

Gonzalez et al., 2012 saw the need to implement a tool that enforces employee data accuracy and completeness. The tool is based on Information Extraction (IE), Adaptive Learning, and Resource Classification. The main advantages of the tool are: the transformation of the extracted data into a planning decision data, the support for precision and recall metrics, and resource utilization and demand economization.

While all efforts are being made to extract data, it is still challenging to identify tools that can perform storage and classification of extracted unstructured data into structured databases. What is even more challenging is the extraction of data from unstructured data sources that contain text and other data formats such as multimedia. A good way to deal with such challenging instances is the adoption of a flat-file data storage approach. This is evidenced in the C# oriented framework presented by Abidin et al., 2010 which employs layered architecture to deal with the unstructured data extraction regarding multimedia. The layers consists of the: Interface layer (the view that will facilitate the interaction between the user and the system), source layer (the data sources which can be in a semi-structured or unstructured format), XML layer (the extracted data is transformed into a flat-file XML document format), and multimedia database (the final storage of the structured data). The final storage location in this case must be a data source that supports multimedia data (e.g., Oracle 11g standards).

But one of the limitations that we perceive with this type of framework is that, it requires the initial data source to be in a webpage. Technically, webpages are in a form of a structure which is defined within DOM elements. So, passing the DOM structure into XML is straightforward. What is difficult is when the source elements are not in any defined tag and are just in documents. Most tools are yet to address this challenge. Another challenge when faced with data extraction especially in software engineering tasks is the fact that mostly, the developers use jargons which are not part of standard dictionaries. When one observes the CTSS tool (Christy and Thambidurai, 2008) for instance, the tool offers a lot of guarantees in terms of short processing time and the fact that different underlying modeling techniques are employed in its deployment. Some of the techniques are passing and soft matching rules. But since the input source is a URL, the wide array of benefits that the tool offers is limited to only web contents.

It is therefore necessary to look for enhanced ways to deploy applications that can accept inputs from non-webpages as well. Moving away from Web contents, some researchers focus on Request for Comments (RFC) documents (Gurusamy et al., 2002). While RFC is syntactically different from HTML, the authors’ definition of unstructured RFC requires some level of structure or organization of text in the document. The tool proposed takes a document and reads certain semantic tags in the document such as Content, Abstract and so on before generating the required structured document. This also means that some semi-structure initialization is required to commence the mining process.

In an attempt to understand email and bug report communication between developers (and knowing that the communication takes unstructured textual format) and not a web content, Panichella et al., 2012 propose the following workflow: (1) Downloading emails and categorizing them into classes, (2) Extracting paragraphs from the reports, (3) Tracing Paragraphs onto Methods, (4) Filtering the Paragraphs, and (5) Computing Textual Similarities Between Paragraphs and Methods. Of course in this approach, the dependency on a semi-structured HTML as data source is alleviated but the challenge is the interpretation of paragraphs in a general textual document. What the authors propose is the extraction of methods as paragraphs which limits the scope of the application in terms of general textual data. But, the uniqueness of this approach is the facilitation of re-documentation of source code artifacts which are extracted from the unstructured data sources to be presented in a summarized format. This approach deviates a little from the overly used language generation techniques to write the summarization through automated techniques. Similarly, Mailpeek (Bacchelli et al., 2012) focuses on text extraction by clustering email contents which are shared by developers.

Realizing the fact that developers are saddled with the responsibility of interacting with unstructured text (especially source code), Bettenburg et al., 2011 outlined some technical plans to establish a meaningful communication between developers. The authors propose a semantic detection module that checks and rectifies spelling and grammar of the text. Clearly, the authors have seen that the software engineering landscape is crowded with violation of technical words which are not found in the standard language dictionaries. A good way to enforce spellchecking is to employ morphological language analysis to identify and describe the morphemes (i.e., smallest linguistic units that are semantically meaningful). The use of linguistic can further aid developers to adopt rationale detection. Rationale-containing texts have matching linguistic relationships which means, the extraction of useful information can be done through the identification of rationale features (Rogers et al., 2012).
From the background works, we seek to adapt some of the ideas to design terms mining tool for document-based NoSQL. The next section details the proposed architectural design.

3. The Architectural Design of the Data Mining Tool

3.1 The Process Flow Execution

The entire system (as depicted in Fig. 2) is made of three layers namely: the view layer, the data mining layer, and the storage layer. The view layer represents a dashboard which allows the user to enter the topics and terms to be mined. This is the output/input layer where the user is enabled to interact with the system. This layer is designed to be deployed on heterogeneous devices such as mobile devices, and desktop (including browser).

When the search term is specified from the input layer, the request is sent as to the entry point on the analytics layer which is designated as the Request Parser. At this point, the specified term is treated as the search artefact. Further, the user has to specify the link to the data source (i.e., the NoSQL storage) in order to facilitate the request parser to guarantee that the search can be carried out.

Without the specification of the link (which must be a URL over an HTTP protocol), the system will not perform the data extraction task. Once the initial requirements are met, the search artefact is sent to the analytics layer for Artefact Extraction Definition. The importance of this component is to differentiate topics from terms, and to further determine the exact requirement of the user. Specifically to this work, we are focusing on extracting single word terms as well as two word terms.

For example, single word terms may be heart, disease, psychiatry, etc.; while two word terms may include heart disease, psychiatry medication, etc. The successful decoding of the search term activates the Semantic Engine component where the analytics framework tries to understand the terms to be extracted.

Initially, we consider the artefact specified by the user as Topics. This aids us to do the traditional information retrieval task which focuses on extracting the exact artefacts specified by the user. This approach also suggests that when a user specifies an artefact, we look for the exact keyword.

However, to perform intelligent terms analytics, there is the need to accommodate the concepts of synonyms, antonyms, parts of speech, and lemmatization as proposed by Zhao and Bhowmick, 2003. Thus, the topics are sent to

http://hipore.com/ijsc
the Topics Parser to check the meaning of the specified artefact. There are two layers that are designed to check the meanings of the artefacts, the Dictionary and the Thesaurus. The dictionary contains standard keywords, their meanings, and their antonyms. The thesaurus contains jargons that otherwise are not found in the standard dictionary. This is important for situations where the analytics is done in certain technical domains such as the medical field where most of the keywords are not in the standard dictionary. The thesaurus can be updated by the specific domain that is adopting the framework for the analytics task. This further makes the proposed tool highly adaptable and agile. When the artefact is reviewed by the topic parser by looking through the thesaurus and dictionary, the Topics Mapping task is enabled. The mapping tasks include the organization of the specified artefact and possible synonyms as may be found in the semantic engine. The expected artefact and the newly found keywords from the dictionary and the thesaurus are categorized as Terms. For instance, when searching for the artefact haemophilia, the terms formation can be formulated as:

{“Hemophilia”: “Bleeding Disorder”}
{“Hemophilia”: “Blood Clotting”}
{“Hemophilia”: “Pseudohemophilia”}
{“Hemophilia”: “Hemogenia”}

The successful formation of the terms leads to the activation of the Search Algorithm component. The algorithm we proposed is based on the inference-based Apriori which, will be discussed in great depth in the next section. The search algorithm component is also linked to the main data source where the data is stored. Using the CouchDB framework for instance, the data source can be accessible over REST API. Primarily, the search algorithm interfaces the data source through the HTTP GET method. The data source which is document style stores the data following the JSON format. Also, the Map/Reduce feature is employed to query multiple documents within a single database. In our framework, the documents are extracted and written in a flat text file. This aids us to easily read through the entire storage repository regardless of the number of documents.

The extracted terms are then reviewed to determine the association between the terms. We apply Association Rules based on the similarities and diversity of the extracted terms. Then, we employ the Filtering methodology to remove the pollutants (or noise) from the data. In most instances, the extracted artifacts are bound to contain keywords that appear valuable but are not actually relevant to the term mining tasks (e.g., antonyms, or similar written words but mean different things). The next task is tagging, which is applied to the filtered data. Tagging is employed to determine the existing relationship between the filtered data and how they are connected to the user specified artifact. At this stage, the Serializer component checks the format of the categorized data for visual presentation. The data must be modelled in JSON format before it can be parsed to the visual display. Once the data clustering is done in JSON, the output is sent to the visualization layer as evidenced in Appendix I. The user can then see the output of the result on the dashboard.

4. The Kalman Filter Algorithm

While the Kalman filter algorithm has been used extensively in areas such as navigation control systems, the adaptation of the methodology in data mining is limited (Pietruszkiewicz, 2008). For further mathematical details and formalisms on the Kalman filter, the intelligent reader is referred to van der Merwe and Wan, 2003. In this section however, we shall discuss how we employ a variance of the Kalman filter which we model as a Bayesian estimate.

First of all, we consider the process of storing data in the NoSQL as time dependent due to the fact that most storage accepts streaming data. The Kalman filter can use recursive estimates to determine states based on incoming data. This is identical to the recursive Bayesian estimation that is an unobserved Markov process, and the data states (i.e., measurements) can follow the hidden Markov model (HMM).

The algorithm relies on the search history of the system users to filter the search result. From the specified terms to be extracted, we can establish the following base relationships:

\[ K \subset Dic \]
\[ T \subset Db \]

To mean that the specified keyword \((K)\) is a subset of the vocabularies in the dictionary/thesaurus \((Dic)\), and the constructed terms \((T)\) from the semantic engine is a subset of the dataset in the NoSQL \((Db)\). In a situation where there are no vocabularies similar (i.e., absence of synonyms or antonyms) in the \(Dic\), then \(K = T\). Also, it is practical to get into situations where \(K\) is not found at all in the dictionary and we treat this case also as \(K = T\). The characteristics of \(T\) can be defined as follows:

\[ T = \{n_1, n_2, n_3, n_4, ..., n_m\} \]

to mean that \(T\) can contain a list of several artifacts, \(n\), that are related to the specified term (described earlier as topics) and in the least, \(T = \emptyset\) to mean that the specified term has no related keywords. Thus, \(T = \emptyset\) is not supported because this is an indirect way of saying that there is nothing specified to be searched. The existence of \(T\) in the data source is assigned a probability of \(I\) and the non-existence of \(T\) in the data source is represented as \(0\).

The proposed system has a Search Log that keeps the record of the frequency of the keywords being searched. This is a disk storage that can grow in size so we use ranking based on the frequency of hits to prioritize which
records to keep. When the storage is growing out of proportion, less frequently searched keywords (i.e., less prioritized keywords) are deleted.

Based on our goal, we expect the Kalman filter will be the best fit to achieve a reliable data mining tasks based on data source that is changing over time. The association rule is designed so that we can determine the presence of frequency of terms, \( T \) in the database \( Db \). Further, we model every pile of document as a row, identical to RDBMS systems.

Also, it best if we consider the interactions within the proposed system as a finite state machine where we can define the states of the data in the NoSQL per every time setup. Given a term \( A \) on a particular NoSQL document, the conditional probability that the term or its dependency keywords \( B \) exist(s) in another appended documents or in another NoSQL database can be expressed as:

\[
f_{ab}(a_i, b_j) = f_{a}(a_i) \times f_{b}(b_j)
\]

However, the condition of the existence of terms can be independent and random which follows that:

\[
f_{ab}(a_i, b_j) = f_{a}(a_i) \times f_{b}(b_j)
\]

Which means following the Markov assumption, the above expression can be re-written given the immediately previous state as:

\[
P(A_n | A_0, \ldots, A_{n-1}) = P(A_n | A_{n-1})
\]

Similarly the measurement of the terms in the NoSQL at the \( n \)-th timestamp is dependent only upon the current state of available list of terms and is conditionally independent of all other states given the current state.

\[
P(B_n | A_0, \ldots, A_n) = P(B_n | A_n)
\]

These assumptions can then be extended to design the hidden Markov model using the probability distribution:

\[
P(A_0, \ldots, A_n, B_1, \ldots, B_n) = P(A_0) \prod_{i=1}^{n} P(B_i | A_i)P(A_i | A_{i-1})
\]

But, the probability distribution of interest in the estimation of \( A \) is conditioned on the availability of data in the current timestamp if the Kalman filter methodology is employed. This requires that we predict and update the next steps. The predicted state can be derived as the sum (integral) of the products of the probability distribution associated with the transition from the \( (n-1) \)-th timestamp to the \( n \)-th and the probability distribution associated with the previous state, over all possible \( A_{n-1} \).

\[
P(A_n | B_{n-1}) = \int P(A_n | A_{n-1})P(A_{n-1} | B_{n-1})dA_{n-1}
\]

The recorded terms within a given timestamp can be measured in different time slots since the data is streaming as:

\[
B_t = \{B_t \ldots, B_t\}
\]

The product of the measurement likelihood and the next predicted state (of possible terms within the document) is a probability distribution:

\[
P(A_n | B_n) = \frac{P(B_n | A_n)P(A_n | B_{n-1})}{P(B_n | B_{n-1})}
\]

Where the normalization term is formalised as:

\[
P(B_n | B_{n-1}) = \int P(B_n | A_n)P(A_n | B_{k-1})dA_k
\]

The Kalman filter algorithm aided us to recursively go through the documents in the NoSQL at various timestamps and the result for the term miming is updated as new readings are recorded.

### 5. EVALUATION

Two datasets from our medical partners (the Saskatchewan Health Region) are analyzed based on the proposed algorithms. These data is on Hemophilia and Psychiatry. We deploy our framework on Amazon EC2 instance with the following specifications: Processor: Intel Xeon, CPU E5410 @ 2.14GHz, 2.37GHz, RAM: 1.70GB, System 32-bit operating system. The output layer is deployed on a personal computer with the following specifications: Windows 7 System 32, 4.12 3.98 GHz, 16GB RAM, 1TB HDD. The size of the training dataset containing the Hemophilia related record is 2 terabyte and the Psychiatry related data is 10 terabyte. The dataset is spread across multiple NoSQL databases that are hosted on separate EC2 instances so that there is not so much latency experienced during the testing. The dictionary and thesaurus are populated with almost 6000 medical jargons specific to Psychiatry and Hemophilia.
For the purpose of testing, we reviewed some of the existing tools on textual data mining and the prominent ones are highlighted in Table II.

Table II. Some Textual Mining Tools.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Major Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orange</td>
<td>Statistical, Machine Learning, Visualization</td>
</tr>
<tr>
<td>WEKA</td>
<td>Machine Learning, Knowledge Analysis</td>
</tr>
<tr>
<td>Apache Mahout</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>RapidMiner</td>
<td>Machine Learning Text Mining</td>
</tr>
<tr>
<td>R</td>
<td>Statistical Computing and Graphics</td>
</tr>
</tbody>
</table>

In this paper, we benchmark the performance of our framework against Apache Mahout (https://mahout.apache.org/) and R (http://www.r-project.org/). In the experimentation, we focus on four major stages of the data mining process as represented in Figure 3, starting with the basic requirement on topic extraction.

- **Topic Extraction**: This is the stage where the specified keyword (artefact) is searched for in the NoSQL and the result is returned with several other keywords. This is the initial step considered before proceeding with the other activities.
- **Term Organization**: This is topics ranking based on relevance. When the topics are extracted, there is no guarantee that all the keywords are actually (truly) relevant. Term organization is where the topics are ranked and non-relevant ones are either eliminated or rated low. The determination of relevance is based on co-occurrence of terms at a particular timestamp.
- **Term Classification**: Since the terms are organized in groups from initial stages, newly identified topics are added to existing groups if they belong to that group; otherwise, a new group has to be created to properly classify those topics.
- **Term Clustering**: At this stage, the terms are grouped and we try to understand the best-fit relationship between the topics in the group.

5.1 Accuracy Index

In Tables IV and V, we report the results of the experiments on the accuracy of the various mining tasks. To validate the result, we compared the reliability of the proposed framework to Apache Mahout and R as shown. There are four base factors that are measured which are: True Positive (TP) - refers to the extraction of expected terms (also referred to as success ratio), False Positive (FP) - refers to the extraction of the desired terms plus unwanted results, True Negative (TN) - is when the term is not found because it does not exist in the NoSQL database, and False Negative (FN) - is when the term could not be found but it exists in the data source. The four base factors are extended to calculate the Precision, Recall, Specificity, Accuracy, and F1 Score. The formalisms are provided in Table III.

Table III. Calculated Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Formula</th>
</tr>
</thead>
</table>
| Precision       | \[
|                 | \frac{TP}{TP + FP} \]                        |
| Recall (Sensitivity) | \[
|                  | \frac{TP}{TP + FN} \]                      |
| Specificity     | \[
|                 | \frac{TN}{TN + FP} \]                      |
| Accuracy        | \[
|                 | \frac{TP + TN}{TP + TN + FP + FN} \]        |
| F1 Score        | \[
|                 | \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \] |

For clarity, we shall refer to our proposed framework in this paper as NoSQL Miner. Considering the two data sources, we studied the performance of each of the tools under discussion. In both tables IV and V, the following activities are recorded: Topic Extraction (i.e., Topic Ext.), Term Organization (i.e., Term Org.), Term Classification (i.e., term Class.), and Term Clustering (i.e., Term Clust.). We observed that the results for both data sets are symmetrical so the nature of data is not a determining factor for performance. This creates the need to study the details of the output.

Table IV. 2 Terabyte JSON Text on Hemophilia Related Records in CouchD

http://hipore.com/ijsc
For brevity, the accuracy index of each of the tools is plotted in Figure 5. Clearly, our proposed NoSQL Miner shows better accuracy result than Apache Mahout and R regarding extraction, organization, classification, and clustering. But before we discuss the accuracy, there is the need to understand the level of falsehood and truthfulness in the result. These factors are the underlying principles for the measurement of accuracy. To measure the truthfulness, we found the sum of the true positive and true negative values. That is, truthfulness is formalized as:

\[
truthfulness = TP + TN
\]

Since we record the TP and TN values as percentages, the expected maximum value for truthfulness is 200. This is plotted in Figure 4.
It is interesting to observe that both Apache Mahout and R outperform our proposed NoSQL Miner in terms of truthfulness. This means for a given data set, the other tools are likely to retrieve more desired topics (i.e., success ratio) and also ignore unlikely topics. This behavior is good and desired in certain systems where the focus is not on the overall accuracy of the extracted terms but rather on the success ratio of extraction. However, if the truthfulness of Apache Mahout and R are better, how come they have least accuracy? This question leads us to analyze the level of falsehood. This is the case where the systems exhibit their levels of wrong perception about topics and terms. In other words, falsehood is the rate at which the tool is wrong. Formally, we express this as:

\[ \text{Falsehood} = FN + FP \]
The level of falsehood in Apache Mahout and R is high. Especially, the false positive results are high. This situation explains why though both tools have high truthfulness result, the final accuracy is poor. The result of both truthfulness and falsehood give us an idea about how Apache Mahout and R are probably doing the topic extraction.

When the data source is parsed to Apache Mahout and R, they try to extract as much topics as possible based on their underlying algorithms. This can be advantageous as more topics (i.e., keywords/artefacts) can be captured which tends to increase the true positive result. At the same time, this behavior can lower the chances of ignoring a correct topic in the data source. As already posited, the attempt to increase the true positive value can be good for some systems. However, the behavior that these tools exhibit shows that while extracting more topics, the false positive value is also increasing. This is because keywords in the data source that are not needed for a particular query result get extracted and added to the terms.

Since we evaluate the data mining activities in four steps, it is important to state that errors are carried over from one step to the next. So, errors in the topic extraction step are carried to the term organization task. And errors in term organization are carried to term classification, and finally to terms clustering. Thus, if we observe the results in Table IV and V, the false positive and false negative values increase from one step to next.

5.2 Search Duration and Data Scale

One of the key contributions of this paper is offering support for streaming data into document-based NoSQL systems. Currently, we are not able to see such support for Apache Mahout and R in this regard. Moreover, these tools require some specific data formats to process the data. For example, the R system uses a package called “rjson” which converts the JSON objects into R objects. Though some effort is required, it can be automated to retrieve data from incoming sources.

Considering the search duration, our proposed NoSQL Miner is faster. This is because we designed our system to interact with the data source directly and process the JSON data in its format without any further conversion. In the case of the other two frameworks, we have to observe the conversion period and when there are errors the procedure has to restart. The search duration is the overall time observed for each tool to perform topic extraction, term organization, term classification, and term clustering.

After repeating the experiments over six (6) times, we realized that to process data (of approximately 30 million topics) on average, the NoSQL Miner spends 2718 seconds while Apache Mahout and R spend 3958.8 and 4438.8 seconds respectively.

Another observation is the fact that Apache Mahout is the most scalable in terms of job processing per time. The framework processes lots of topics per time. In case we are considering batch data, then Apache Mahout will have been the fastest in terms of accomplishing the unstructured data mining tasks. However, when the data is streaming and the processing is done over time, the NoSQL Miner is the fastest. The reason is most of the bottlenecks with object conversion are not present in the NoSQL Miner.

5.3 Limitations

Though our proposed NoSQL Miner outperforms the Apache Mahout and R tools, we cannot conclude that the underlying algorithms of the various tools make the difference. For instance, it is not clear whether we can conclude that the Kalman filter algorithm in the case of the NoSQL Miner is better than the Fuzzy K-Means clustering, Naive Bayes classifier, and Random forest classifier in the case of Apache Mahout.

The study of the underlying algorithms requires independent research and it is best if these algorithms are tested in the same framework. However, the proposed NoSQL Miner can be considered as a preferred choice within the context of our research goal which focuses on mining unstructured streaming data at different times and re-adjusting the classifier based on incoming data.

This goal naturally fits into the Kalman filter algorithm and since we model this algorithm as a Bayesian estimate (Section 4), we are able to attain good results. It will be interesting to know how the result will be affected if the engineering challenges of automatically mining streaming data in Apache Mahout and R are overcome.

6. Conclusions

There is increasing amount of unstructured data and NoSQL databases have been proposed as preferred storage facilities. While several NoSQL databases exist, our work focuses on the document-append NoSQL database. This type of NoSQL supports actual data and files. Some examples are CouchDB (which is tested in this paper) and MongoDB. The challenge however is that, the support for mining data from this type of storage is lacking from both engineering and data mining perspectives. Most of the textual mining tools available are not easily applicable to document-based NoSQL. Even though the data from document-based NoSQL can be accessed via the browser, Web crawlers are not a good fit for mining data from this storage facility. This is because Web crawlers rely on html tags and attributes to understand contents on the Web while data retrieved from document-based NoSQL is just text (e.g., JSON).

This paper therefore propose and design a data mining tool for document-based NoSQL systems using the Kalman filter as the underlying algorithm. Two main aspects of big data, variety and velocity, are considered in the design of the tool. The performance of the proposed tool is compared with well-established frameworks for textual mining such as Apache Mahout and R.
In summary, the work accomplishes the following in the area of unstructured data mining in the big data era:

- Proposed a data mining framework for Document-Style NoSQL
- We proved that the Kalman filter algorithm though used extensively in areas such signal processing can be used in textual mining of streaming data in document-based NoSQL systems.
- The proposed framework supports topic extraction, term organization, term classification and term clustering
- In comparison to Apache Mahout and R, the proposed framework is:
  - More accurate when considering data inflow over some time slots.
  - More flexible to adapt to other data sources. Though not tested in this work, the framework can process any text or flat database (e.g., XML).
  - More scalable to accommodate several data sources. This is because the framework can access several data sources via HTTP connection.
  - Less scalable than Apache Mahout in terms of the processing job per time.

In the future, this work can be extended to other areas of analytics outside topics and terms. Works can also be done in the areas of re-usable dictionaries, and file organization.

7. ACKNOWLEDGMENT

Thanks to our industry partners from the Saskatchewan Health Region, Canada who are the enterprise consumers of this work.

Final big thanks to Prof. Patrick Huang (University of Ontario Institute of Technology), Prof. Jia Zhang (Carnegie Mellon University) and the entire editorial team of the International Journal of Services Computing (IJSC).

8. REFERENCES


http://hiipore.com/ijsc

28


Authors

Appendix I

Richard K. Lomotey is currently pursuing his Ph.D. in Computer Science at the University of Saskatchewan, Canada, under the supervision of Prof. Ralph Deters. He has been actively researching topics relating to mobile cloud computing and steps towards faster mobile query and search in today’s vast data economy (big data).

Prof. Ralph Deters obtained his Ph.D. in Computer Science (1998) from the Federal Armed Forces University (Munich). He is currently a professor in the department of Computer Science at the University of Saskatchewan (Canada). His research focusses on mobile cloud computing and data management in services ecosystems.

Visualization on Hemophilia using our Data Mining Tool