Terms Mining in Document-Based NoSQL: Response to Unstructured Data

Richard K. Lomotey and Ralph Deters
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada
richard.lomotey@usask.ca, deters@cs.usask.ca

***Abstract*—Unstructured data mining has become topical recently due to the availability of high-dimensional and voluminous digital content (known as "Big Data") across the enterprise spectrum. The Relational Database Management Systems (RDBMS) have been employed over the past decades for content storage and management, but, the ever-growing heterogeneity in today's data calls for a new storage approach. Thus, the NoSQL database has emerged as the preferred storage facility nowadays since the facility supports unstructured data storage. This creates the need to explore efficient data mining techniques from such NoSQL systems since the available tools and frameworks which are designed for RDBMS are often not directly applicable. In this paper, we focused on topics and terms mining, based on clustering, in document-based NoSQL. This is achieved by adapting the architectural design of an analytics-as-a-service framework and the proposal of the Viterbi algorithm to enhance the accuracy of the terms classification in the system. The results from the pilot testing of our work show higher accuracy in comparison to some previously proposed techniques such as the parallel search.**

Keywords- Unstructured Data Mining; Big Bata; Viterbi algorithm; Terms; NoSQL; Association Rules; classification; clustering.

I. *INTRODUCTION*

The enormous data at our disposal in the form of digital assets is greatly transforming the enterprise landscape. The high-dimensional data is termed as "Big Data" [1, 29] and it is defined as the confluence of: *Big transaction data* (i.e., exponential increase and diversity in the volume of transaction data), *Big interaction data* (i.e., increase in open data such as social media and device data), and *Big data processing* (i.e., increasing processing demand on high-dimensional data).

There are a lot of opportunities that can be derived from big data especially at the corporate and enterprise levels. New ideas are being formulated and new knowledge being discovered from the highly interconnected data while, availability and accessibility of data is also enhanced. However, at the enterprise, the issue is with the storage of the data. Most of the recently generated content is unstructured which means the data: 1) is heterogeneous (e.g., file documents, video, image, etc.), 2) has no standard schema, and 3) is from multiple sources.

Furthermore, while the past decades have witnessed the dominance of data storage following the Relational Database Management System (RDBMS) standard, this is no longer the case in today's data economy since only about 20% of the data is structured [1]. Rather, the NoSQL storage is seen as a practical way to store data in the "Big Data" era. NoSQL offers the enterprise players the flexibility to accommodate any form of data which can be structured, semi-structured, and unstructured.

The problem however is that, existing data mining techniques which are designed to work for RDBMS systems are not directly applicable in NoSQL environments [2].

This creates the need for research on data mining in the area of NoSQL storages, especially in the areas of document-based storages. Unfortunately, the area is understudied. But, at the general textual mining front, there has been active works which has led to the proposal of methodologies such as: information retrieval algorithms based on templates [3], association rules [4-5], topic tracking and topic maps [6], term crawling (terms) [7, 28], document clustering [8], document summarization [9], Helmholtz Search Principle [10], re-usable dictionaries [11], and so on. On the whole, most of the methodologies are based on Natural Language Processing (NLP) which is inherited from Artificial Intelligence and Linguistics [12-14].

Some of these methodologies are the underlying algorithms on which some of the current tools on unstructured data mining are built. Examples are the Apache Mahout [30], SAS text Miner [31], and R [32]. However, these tools are limited in terms of their direct applicability to NoSQL systems.

A previous work has proposed the architectural design of an analytics-as-a-service framework that aims at data mining from heterogeneous data sources such as RDBMS, Web content, and NoSQL [36]. The limitation however is that, the proposed system did not have any concrete rules for the determination of association between terms for the purpose of benchmarking against other tools. Also, the proposed algorithms such as parallel and linear search may lead to high terms extraction but not necessarily accurate when it comes to clustering and classification of terms.

Hence, this work adapts the previously proposed analytics-as-a-service framework by enhancing the component for NoSQL data mining. We put forward the Viterbi algorithm ahead of the parallel search technique in order to enhance the classification and clustering of the terms. Using an available dataset from the health domain on Psychiatry, the proposed system is tested and the results are compared to existing framework called RSenter [33]. Our findings are summarized as follows:

978-1-4799-5057-7/14 $31.00 © 2014 IEEE
DOI 10.1109/BigData.Congress.2014.99

- The *parallel search* technique which was proposed in the RSenter framework performs the topics extraction faster than the *Viterbi algorithm*.
- The True Positive and True Negative indices of the extracted topics for both algorithms are almost identical.
- The accuracy of the Viterbi algorithm regarding topics extraction is better than the parallel search due to the presence of high False Positive in the latter algorithm.
- The Viterbi algorithm performs better at terms organization which in this work is topics ranking based on relevance.
- The Viterbi algorithm performs much better clustering than the parallel search.
- The Viterbi algorithm is a better option for classification among the two methodologies.

The remaining sections of the paper are structured as follows. The next section discusses NoSQL storages and unstructured data analytics. Section III discusses the architectural design of the analytics-as-a-service framework and the underlying algorithms. And the evaluation of the framework is carried out in Section IV. The paper concludes in Section V with our contribution, lessons learned, and future direction.

II. *NoSQL and Unstructured Data*

Big data is driven by wide availability of voluminous data, highly heterogeneous, and content generation at a very fast pace (data streams). The data is mostly in heterogeneous formats and follows no particular structure. Since this situation has rendered RDBMS systems ineffective, the NoSQL [20] storage is preferred.

The latter storage style can accommodate schemaless, semi-structured, and unstructured data. While some of the NoSQL databases such as MongoDB, Google BigData, etc. support actual data formats (e.g., in JSON, XML, or other textual formats), others are purely file storages (e.g., Dropbox, Amazon S3, MEGA, etc.). Further, there are some that support both textual data and files as attachments (e.g., CouchDB [25]).

Also, in view of the fact that graphing has achieved significant success, emerging NoSQL databases have adapted the idea. Graph NoSQL databases (e.g., Neo4J) can be employed to build data storages in a way that can aid in simple queries as well as establishing relationship between the data without writing relational database queries [16-19]. Relational databases rely on keys and join operations which makes searching tedious in data silos. Further, most of the previously deployed NoSQL databases rely on MapReduce functions which cannot be shared but are specific to each NoSQL database. However, graph databases allow direct access to the data source of interest without any join or MapReduce operation. Moreover, graph databases are transactional and are optimized for cross-record connections.

The problem however is that, these NoSQL databases are products of different vendors so their query styles vary. Moreover, in order to aggregate the data from all of these sources, a mashup service has to be deployed which presents the researcher with two problems. Firstly, multiple APIs has to be studied and secondly, the returned data from the individual data silos are in different structures [15].

Also, most of the ongoing works on big data are seen on the Hadoop [34] framework. This may be due to the fact that the framework supports distribution of storages, which is a natural alignment to the RDBMS system. However, the other forms of NoSQL storages have received little attention from the research community even though they have heavy enterprise adoption.

The proposal of the Analytics-as-a-Service platform in [36-37] is aimed at performing data mining from heterogeneous storage layers including SQL, NoSQL, and Web content. The proposed algorithms however follow the parallel search technique which are inherent in the RSenter framework [33], [35]. In this wok, we seek to narrow the storage scope of the analytics-as-a-service framework to only the document-based NoSQL. However, an enhanced search algorithm will be proposed to enhance the terms mining.

III. *The Analytics-as-a-Service Architecture*

A. The Architectural design Flow

The architectural design of the analytics-as-a-service framework was originally proposed in [36] and further extended in [37]. This section reproduces the architectural design while the proposed algorithm is described in a latter section. The architectural design is shown in Figure 1.

On the input layer, the user is facilitated to perform two initial tasks: ***Search Criteria*** selection and *Data Source* specification. The search criteria tell the system whether the user wants to perform the topics and terms mining based on certain specifications. A typical specification can be the extraction of topics which are related to some particular keywords; where related can mean keywords which are antonyms or synonyms to the specified topic. Users can also define their own specifications, which will be explained later. Further, the user has to specify the data sources from where the data can be mined. A URL or relative path to the database(s) can be provided in this case.

Once the user's initial tasks (i.e., inputs through selection and specification) are completed, the information is parsed to the ***Request Parser***, which is an interface of the AaaS. As suggested in earlier sections, the AaaS framework is considered as a back-end system and can be deployed on an organization's internal server or on a cloud platform. However, we strongly recommend a powerful platform for hosting because the AaaS is responsible for the high computation and processing of the data mining tasks. The entire implementation of the AaaS is in Erlang [27] which is a programming environment that supports concurrency and inter-process communication. Since the AaaS framework acts as an application server, the *Request Parser* is the network component that receives the information from the input layer (using WebSocket or HTTP) and routes the request to the other components.

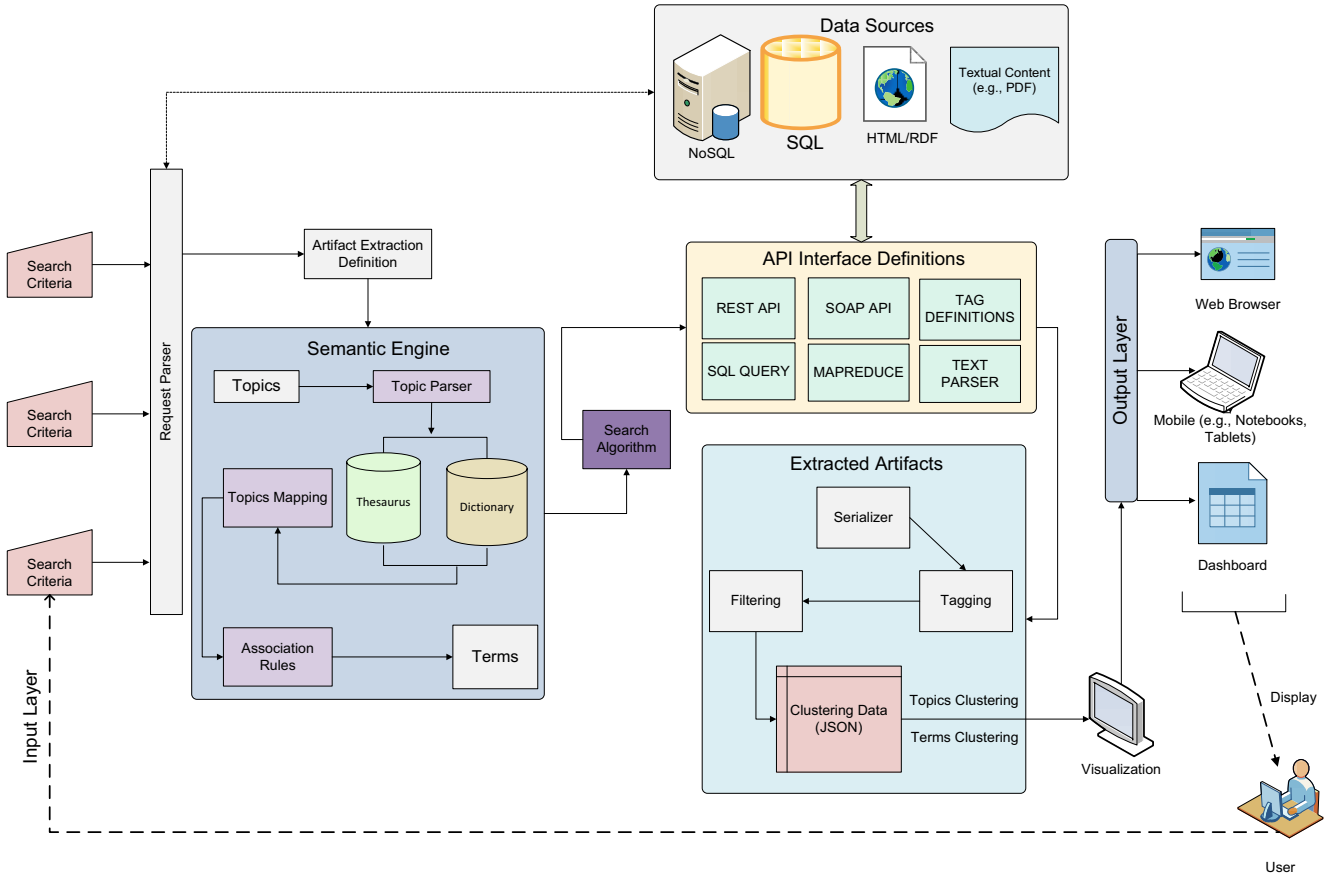

Figure 1. The architectural design of the Analytics-as-a-Service (AaaS) tool [36]

The requests are first routed to the ***Artifact Extraction Definition*** component. The responsibilities of this component include the validation of the input information from the user, interpretation of the requests, and the request categorization. The user input can be invalidated for instance if the specified data source does not exist or file format is currently not supported or the specified path is wrong.

Though currently not implemented, we want to perform other tasks in the future such as document clustering, file structure organization, etc. In that case, the artifact extraction definition component will be responsible for the categorization of the input request based on the data or document extraction needs of the user. When the request is validated, the artifact extraction definition component then interprets the request as topics extraction or terms extraction. The interpreted request is then sent to the ***Semantic Engine*** layer.

The goal of the semantic engine layer is to pass the request through series of reviews that will improve the quality of the data mining result. In the current version of the AaaS, we treat *Topics* as keywords that the user wants to extract from the data source. Typical cases of topics extraction can be looking for artifacts such as "Contract", "Economy", "Sports", etc. from a data source. The semantic engine will not have much to do in this case but to pass the topics for extraction at the next stage. However, *Terms* extraction involves a lot more because terms consider a specified keyword plus possible dependency keywords for interpretation. For example, when the artifact "Contract" is specified, it can also mean "Agreement" therefore the mining process will have to consider the latter keywords as well. This latter requirement convinced us to design the terms extraction as an extension on the topics. The consideration of the latter keyword can be specified by the user as part of the specification process on the input layer or left for the AaaS layer to decide and later the user can refine the result. In order to clearly explain the workings of the semantic engine to the reader we provide Table I as an example that we shall use to discuss the remaining sub-components.

TABLE I. SAMPLE ARTIFACT

Artifact	Synonyms	Antonyms
Contract	Arrangement	Disagreement
	Agreement	Misunderstanding
Contract	Acquire	Give
	Afflict	

Assuming the keyword "Contract" is the specified term, we necessarily need to understand what the word mean from the user's perspective. In the English Language, "Contract" can be synonymous to "Agreement" and the other words in Table I or can be synonymous to "Afflict" and the other words in similar line. The first problem that arises here is the presence of semantic issues where the mining process can potentially return terms involving "Acquire" when the user actually wants "Agreement". Even if we assume that the extracted terms should just return everything on "Contract", the outcome will be a result with a lot of False Negatives. This will eventually affect the *accuracy* and *specificity* of the result as we shall see later in the evaluation section.

So, when an artifact is validated and sent to the semantic engine, the latter considers that artifact as a ***Topic***. This topic is then forwarded to the ***Topic Parser*** where the actual meaning of the artifact is defined in order to generate an enhanced result. The topic parser is also important for the minimization of the other tasks ahead such as data cleansing. The topic parser has two components which are the *Dictionary* and the *Thesaurus*. These two components are storages which are built in DETS (a disk storage facility in Erlang) to keep a tabular structure as Table II. The dictionary contains predefined artifact in English, their corresponding synonyms, and antonyms. However, another concern at this point is how to make the dictionary adaptive. To achieve the adaptability of the dictionary, we build a separate lexical component which is the thesaurus. Adaptability in our case means that the topics and terms process should be extended or expanded to other domains outside the English Language. For instance, in the medical domain there are jargons that are not in the English dictionary or there are medical specific lexicons. To make our AaaS framework as generic as possible, the thesaurus only need to be populated with the domain specific jargons such as the case of the medical domain.

Since the dictionary and thesaurus are both tabular, we need to establish relationship between the artifacts. This requires a network (or graph) of inter-connected words because, for each word, we need to know all other artifacts that are dependencies either as synonyms or antonyms. Here is where we define a component called ***Topics Mapping*** where words are linked to each other based on the relationship that exists between the words. The topics mapping component is a script running on a *Bloom Filtering* methodology. Considering the vast amount of data that can exist in the lexicon (such as the thesaurus), it will be time consuming to go through the entire table in search for the synonyms and antonyms. Another way to imagine this scenario is: an artifact will have a synonym and the same synonym can later be an artifact which means the previous artifact becomes a synonym.

After the determination of all dependencies through topics mapping, ***Association Rule*** is established where the user request is matched against the topics mapping.

These rules are contained in a JSON script internally; for example, in the case of the third rule, the JSON will be as shown below:

{ "Keyword": "Contract",
"Synonym": "Agreement"
}

The establishment of the Association Rule prepares the semantic engine for the ***Terms*** extraction. The rule-based JSON file is then passed to the ***Search Algorithm*** layer. The search algorithm determines which methodology to apply when searching through the data sources. Currently, the proposed algorithms are: *Linear/Random Search algorithm, Parallel/Concurrency Search algorithm, and Pessimistic and Optimistic Search algorithms.*

The search algorithm specification activates the ***API Interface Definition*** layer. Here is where the queries are generated, constructed, and issued for the specific data sources. For instance, SQL query cannot be issued for a NoSQL specified database nor a Map/Reduce query for SQL database. This means for every supported data store, the query style that is specific to that data source must be issued. In the current implementation, the following Application Programming Interfaces (APIs) are defined and fully functional. For brevity, the details of the actual API implementation and calls are omitted from this paper; however, the documentation of the AaaS tool, which will be made open source, will contain the details. This is the part that is crucial to the developers because the understanding of the APIs is what will allow the customization of the queries for the specific mining task. For detail explanation of the APIs, the reader is referred to [36] and [37].

The completion of the query construction opens the connection to the specified database on the ***Data Sources*** layer. This layer is not necessarily on the same environment as the AaaS framework. The data source can be on remote servers within the same organization, outside repositories, or on the same server as the AaaS. For every request that is issued by the specific API interface to the data source, a response is expected which is passed to the ***Extracted Artifacts*** layer.

The extracted artifact layer is where the Knowledge Discovery in Database (KDD) process starts. The goal here is to determine existential evidence of topic and terms relationship through the KDD process. The data which is returned from the data source through the API interface is passed to the ***Serializer*** component. There are cases where the returned data is from multiple API interfaces especially when the data mining process involves multiple data sources. Thus, the Serializer is the first component that receives the data and performs the following macro activities:

Normalization: reducing the redundancy of the data and establishing relationship between the artifacts.

Grouping: Organizing similar artifacts into categories, and

Integration: Merging the artifacts into a format that gives single dimensional overview of the search result.

After the Serializer is done, ***Tagging*** is performed the data. Tagging involves determining the sources of each data, occurrences of specific artifacts, frequency of artifact occurrences etc. for statistical purposes. Once the tagging process is complete, the data is passed to the ***Filtering*** component. Here is where the topics and terms are organized based on relevance. For instance, we can determine

relevance based on the frequency of occurrences of terms from the tagging process; and only report results based on certain thresholds such as top 100 terms and so on. The filtered data is then organized based on ***Clustering*** where the data is modelled in JSON format for easy passing. Depending on the data mining need as specified by the user, the JSON data is parsed as *Topic Clustering* or *Terms Clustering* to the ***Visualization*** layer.

Results from the visualization layer are sent to the Output layer which is primarily the platform of choice that the user prefers to view the result. Currently, results can be viewed as a browser content, on tablet and notebooks, and as dashboard. An example of an output is Appendix I (Fig. 3) which shows the clustering result for "Psychiatry" related data.

B. The Viterbi algorithm

In order to improve on the clustering and classification of terms, the Viterbi algorithm with a well-defined expectation maximization methodology is proposed. Previous researchers such as Shinghal and Toussaint [39] and Wang et al. [38] employed the Viterbi algorithm for statistical analysis of sequences. Since topics and terms mining require the extraction of similar (related) artifacts, we expect that the modification of the algorithm within the context of document-based NoSQL will improve on the classification of terms. Besides, we will be able to model the detection of keywords in documents within the NoSQL as sequences and the search from one document to the next can be treated as a state transition. The document-based NoSQL (e.g., CouchDB [25]) comprises of several layered documents, identical to tables in RDBMS though the documents contain text of n-strings in mostly JSON format. This requires that we define a relationship between the specified keyword to be extracted, the term formed, the existence of the keyword in the dictionary, and the existence of the term in the NoSQL database. From the premise from [40] we can formally write:

$$K \subset Dic$$

$$T \subset Db$$

This means 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 database (Db). In a situation where there are no similar vocabularies (i.e., absence of related topics) 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.

When a keyword K is specified and there exist dependency/related keywords to K, we refer to each of the individual keywords as topics. We then need to determine the relationship between the topic with regard to K, and this is what we define as term formation. So, if we define topics as $t_1, t_2, \ldots, t_n$, then T in the NoSQL (i.e., Db) can be defined as follows:

$$T = \{t_1, t_2, t_3, t_4, \ldots, t_n\}$$

to mean that T can contain a list of several artefacts that are related to the specified term (described earlier as topics) and in the least, T = {t1} to mean that the specified term has no related keywords. Thus, T = ∅ (or null) 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 represented as 1 and the non-existence of T in the data source is represented as 0. Since the Viterbi algorithm is an extension of the Hidden Markov Model (HMM), we shall advance the HMM algorithm that is proposed in [41].

First, we define the entire *Db* containing all artefacts as the observation space *O*, such that $O = \{o_1, o_2, 0_3, o_4, \ldots, o_n\}$. Then we have to define a state space *S*, such that $S = \{s_1, s_2, s_3, s_4, \ldots, s_n\}$ where the state space represents the actual documents within the NoSQL. In this case *S* is a subset of *O* (i.e., $S \subset O$). We then define the extracted topics and their relationships as the sequence of observations *Y*, such that $Y = \{y_1, y_2, y_3, \ldots, y_n\}$.

From the thesaurus/dictionary, the system is able to determine the likelihood of related terms based on an initially assigned probability index, say Γ_i in state *i* and a transition from one document, *i*, to the next, *j*, with a probability of $\alpha_{i,j}$. Considering the most likely extracted term as a sequence *X*, such that $X = \{x_1, x_2, x_3, x_4, \ldots, x_n\}$, then

$$\beta_{1,k} = P(y_1 \mid k) \,.\, \Gamma_k$$

$$\beta_{n,k} = P(y_n \mid k) \,.\, max_{x \epsilon S} \left(\alpha_{x,k} \,.\, \beta_{n-1,x}\right)$$

Where $\beta_{n,k}$ represents the state sequence of n observed terms and k as the last in the sequence.

Thus, the function say *Func(k, n)* that returns the likely extracted term *x* for the determination of $\beta_{n,k}$ if $n > 1$, or *k* if $n = 1$, can be extended to:

$$x_N = argmax_{x \epsilon S} \left(\beta_{N,x}\right)$$

$$x_{n-1} = Func\,(x_n, n)$$

Using the Kalman filter algorithm, we are able to determine the maximum likelihood of the term formation, classification, and clustering through noise reduction based on the variance, *Var*:

$$Var = \frac{1}{N} \sum_{k=1}^{N} (z_k - x_k)^2$$

Where x_k is a scalar value from the measurement.

IV. EVALUATION

This section discusses the evaluations of the proposed system. In order to validate the results, we considered the same Psychiatry related medical data that is used in the evaluation of RSenter [35]. This data is stored in 30 separate CouchDB databases. The analytics-as-a-service part is deployed on an Amazon EC2 instance which has the following specifications: *Processor: Intel Xeon, CPU E5410 @ 2.33GHz, 2.41GHz, RAM: 1.70GB, System 32-bit operating system.* The view layer is deployed on a personal computer with the following specifications: *Windows 7 System 32, 4.12 3.98 GHz, 16GB RAM, 1TB HDD*. In the

experimentation, we focus on four major stages of the data mining process as represented in Fig. 2, starting with the basic requirement.

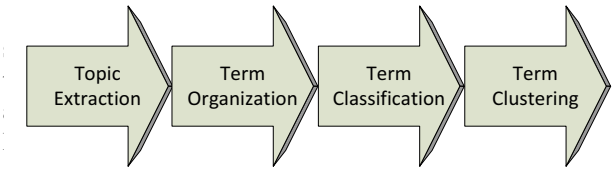

Figure 2. The Proposed Data Mining Stages from the NoSQL

- **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 stage of all the mining 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.
- **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.

So, by focusing on these four stages, we experimented with the proposed Viterbi algorithm in this work and the parallelization methodology that is proposed in RSenter [33, 35]. The experimented values are gathered and analysed using four base factors. True Positive (TP) refers to the extraction of expected results. False Positive (FP) refers to the extraction of perceived to be desired results but those results are not needed. True Negative (TN) is when the term is not found because it does not exist. False Negative (FN) is when the term could not be extracted but it exists in the NoSQL. These four base factors are extended to analyse for each algorithm, the Precision, Recall (Sensitivity), Specificity, Accuracy, and F1 Score. The results are reported in Tables II.

First, we analysed the topic extraction task. It is observed that both algorithms (the Viterbi and Parallelization) have almost identical TP value, i.e., 98.47% and 98.02% respectively. However, the accuracy of the extracted topics differs much since the Viterbi algorithm is about 94.58% accurate and the parallelization is about 89.99%. There is therefore the need to examine the source for the variation in the accuracy of the extracted topics. From the results, the true negative values for both algorithms are also almost identical. The false positive (FP) values however are a bit apart. The FP of the Viterbi algorithm is 3.95% and that of the parallelization is 5.02%. But, the main source of problem is from the false negative. The parallelization methodology returns higher false negative percentage value than the Viterbi algorithm.

The second analysis is the term organization. This is where we tried to rank the topics based on relevance. In the extraction of the keyword "Psychiatry", there are several words that are returned (over 4200 topics) but these words need to be ranked based on relevance and some may not even be of interest since they can be false positive data. Since term organization is a step from the topic extraction, errors in the latter can be extended to the former. This further sees a decline in the TP value in comparison to topics extraction. Overall, the Viterbi algorithm performs better term organization than the parallel algorithm. We saw that the true positive percentage of the parallelization methodology is 86.11% far less than 95.31% of the Viterbi algorithm. The high FN value is a concern too.

TABLE II. THE ANALYSIS OF THE EXTRACTED TERM FOR PSYCHIATRY

	Viterbi Algorithm				**Parallelization**			
	Topic Extraction	**Term Organization (Topics Ranking)**	**Term Classification**	**Term Clustering**	**Extracted Topics**	**Organized Terms (Topics ranking)**	**Term Classification**	**Term Clustering**
True Positive	98.47%	95.31%	93.77%	91.09%	98.02%	86.11%	80.33%	71.17%
False Positive	3.95%	4.02%	6.55%	9.33%	5.02%	8.87%	14.72%	20.33%
True Negative	98.87%	96.22%	95.00%	92.66%	97.95%	88.61%	81.21%	80.17%
False Negative	7.35%	10.13%	16.34%	22.41%	16.77%	24.68%	28.81%	35.41%
Precision	96.14%	95.95%	93.47%	90.71%	95.13%	90.66%	84.51%	77.78%
Recall (Sensitivity)	93.05%	90.39%	85.16%	80.26%	85.39%	77.72%	73.60%	66.78%
Specificity	96.16%	95.99%	93.55%	90.85%	95.12%	90.90%	84.66%	79.77%
Accuracy	94.58%	93.12%	89.19%	85.27%	89.99%	83.89%	78.77%	73.08%
F1 Score	94.57%	93.09%	89.12%	85.16%	90.00%	83.70%	78.68%	71.86%

The next stage is the terms classification. We analyzed how the framework groups new topics to existing groups. Based on previous results in [35], we already have some predefined group of terms for validation. Based on the classification, we observed the Viterbi algorithm shows 89.19% accuracy while the parallelization methodology shows 78.77%. The FN results for both algorithms are significant, but, the parallelization methodology has significantly much value around 28.81.

Finally, we analyzed the clustering result. We analyzed the grouped data within the context of relatedness. The Viterbi algorithm has 85.27% accuracy and the parallelization methodology has 73.08% accuracy. The defining factors for the clustering are the FN and the FP results.

Appendix I (Fig. 3) shows the clustering result of the Viterbi algorithm.

V. CONCLUSION

This work proposes the Viterbi algorithm as a methodology to improve the analytics of terms in document-based NoSQL systems. While the parallelization methodology was initially proposed as a means for speedy terms extraction, the methodology is not efficient when it comes to topics organization. Thus, in the era of "Big Data" where a lot is required regarding analytics, there is the need for improvement.

The testing of the proposed Viterbi algorithm in comparison to the parallelization methodology shows that the Viterbi algorithm is better at terms organization, terms classification, and terms clustering. Future works will focus on multimedia data analytics.

REFERENCES

[1] K. RUPANAGUNTA, D. ZAKKAM, AND H. RAO, "How to Mine Unstructured Data," Article in Information Management, June 29 2012, http://www.information-management.com/newsletters/data-mining-unstructured-big-data-youtube--10022781-1.html

[2] D. KUONEN, "Challenges in Bioinformatics for Statistical Data Miners," Bulletin of the Swiss Statistical Society, Vol. 46 (October 2003), pp. 10-17.

[3] J. Y. HSU, AND W. YIH, "Template-Based Information Mining from HTML Documents," American Association for Artificial Intelligence, July 1997.

[4] M. DELGADO, M. MARTÍN-BAUTISTA, D. SÁNCHEZ, AND M. VILA, "Mining Text Data: Special Features and Patterns," Pattern Detection and Discovery, Lecture Notes in Computer Science, 2002, Volume 2447/2002, 175-186, DOI: 10.1007/3-540-45728-3_11

[5] Q. ZHAO AND S. S. BHOWMICK, "Association Rule Mining: A Survey," Technical Report, CAIS, Nanyang Technological University, Singapore, No. 2003116 , 2003.

[6] W. ABRAMOWICZ, T. KACZMAREK, AND M. KOWALKIEWICZ, "Supporting Topic Map Creation Using Data Mining Techniques," *Australasian Journal of Information Systems*, Special Issue 2003/2004, Vol 11, No 1.

[7] B. JANET, AND A. V. REDDY, "Cube index for unstructured text analysis and mining," *In Proceedings of the 2011 International Conference on Communication, Computing & Security (ICCCS '11)*. ACM, New York, NY, USA, 397-402.

[8] L. HAN, T. O. SUZEK, Y. WANG, AND S. H. BRYANT, "The Text-mining based PubChem Bioassay neighboring analysis," BMC Bioinformatics 2010, 11:549 doi:10.1186/1471-2105-11-549

[9] L. DEY, AND S. K. M. HAQUE, "Studying the effects of noisy text on text mining applications," In *Proceedings of the Third Workshop on Analytics for Noisy Unstructured Text Data (AND '09)*. ACM, New York, NY, USA, 107-114. DOI=10.1145/1568296.1568314

[10] A. BALINSKY, H. BALINSKY, AND S. SIMSKE, "On the Helmholtz Principle for Data Mining," Hewlett-Packard Development Company, L.P.

[11] S. GODBOLE, I. BHATTACHARYA, A. GUPTA, AND A. VERMA, "Building re-usable dictionary repositories for real-world text mining," *In Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM '10)*. ACM, New York, NY, USA, 1189-1198.

[12] F. S. GHAREHCHOPOGH, AND Z. A. KHALIFELU, "Analysis and evaluation of unstructured data: text mining versus natural language processing," *Application of Information and Communication Technologies (AICT)*, 2011 5th International Conference on , vol., no., pp.1-4, 12-14 Oct. 2011, doi: 10.1109/ICAICT.2011.6111017

[13] V. TUNALI, AND T. T. BILGIN, "PRETO: A High-performance Text Mining Tool for Preprocessing Turkish Texts," 2012 *International Conference on Computer Systems and Technologies*.

[14] S. V. VINCHURKAR, AND S. M. NIRKHI, "Feature Extraction of Product from Customer Feedback through Blog," *International Journal of Emerging Technology and Advanced Engineering,* (ISSN 2250-2459, Volume 2, Issue 1, January 2012)

[15] J. SEQUEDA AND DANIEL P. MIRANKER, "Linked Data," Linked Data tutorial at Semtech 2012, Jun 07, 2012. Available: http://www.slideshare.net/juansequeda/linked-data-tutorial-at-semtech-2012

[16] Facebook Open Graph, Availlable: https://developers.facebook.com/docs/concepts/opengraph/overview/

[17] Google Knowledge Graph, Available: http://www.google.ca/insidesearch/features/search/knowledge.html

[18] Bing one-ups Knowledge Graph, The Next Web, http://thenextweb.com/microsoft/2012/06/07/bing-challenges-googles-knowledge-graph-with-new-britannica-encyclopedia-partnership/

[19] Twitter Interest Graph, http://blogs.ischool.berkeley.edu/i290-abdt-s12/2012/11/25/analyzing-the-twitter-social-graph/

[20] NoSQL, http://nosql-database.org/

[21] R. FELDMAN, M. FRESKO, H. HIRSH, Y. AUMANN, O. LIPHSTAT, Y. SCHLER, AND M. RAJMAN, "Knowledge Management: A Text Mining Approach," Proc. of the 2nd Int. Conf. on Practical Aspects of Knowledge Management (PAKM98), Basel, Switzerland, 29-30 Oct. 1998.

[22] R. FELDMAN, M. FRESKO, Y. KINAR, Y. LINDELL, O. LIPHSTAT, M. RAJMAN, Y. SCHLER, AND O. ZAMIR, "Text mining at the term level," *Proc. of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98)*

[23] J. C. SCHOLTES, "Text-Mining: The next step in search technology," DESI-III Workshop Barcelona, Monday June 8, 2009.

[24] J. LEE, D. GROSSMAN, O. FRIEDER, AND M. C. MCCABE, "Integrating structured data and text: a multi-dimensional approach," Proceedings of Information Technology: Coding and Computing, 2000. International Conference on , vol., no., pp.264-269, 2000, doi: 10.1109/ITCC.2000.844234

[25] CouchDB, http://couchdb.apache.org/

[26] SCHEFFER, T., DECOMAIN, C., AND WROBEL, S. 2001. Mining the Web with active hidden Markov models. Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on , vol., no., pp.645-646, 2001, doi: 10.1109/ICDM.2001.989591

[27] Erlang Programming Language, http://www.erlang.org/

[28] Y. GU, C. KALLAS, J. ZHANG, J. MARX AND J. TJOE. 2013. Automatic Patient Search Using Bernoulli Model. In Proc. of 2013 IEEE International Conference on Healthcare Informatics (ICHI 2013), page 517-522, September 9-11 2013, Philadelphia, PA, USA.

[29] R. AKERKAR, C. BADICA, AND D. B. BURDESCU, “Desiderata for research in web intelligence, mining and semantics”, In Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics (WIMS '12). ACM, New York, NY, USA, , Article 0 , 5 pages. DOI=10.1145/2254129.2254131 http://doi.acm.org/10.1145/2254129.2254131

[30] Apache Mahout, http://mahout.apache.org/

[31] SAS Text Miner, http://www.sas.com/en_us/software/analytics/text-miner.html

[32] R, http://www.r-project.org/

[33] R. K. LOMOTEY, AND R. DETERS, "RSenter: Tool for Topics and Terms Extraction from Unstructured Data Debris," 2013 IEEE International Congress on Big Data (BigData Congress 2013), pp.395,402, June 27 2013-July 2 2013, doi: 10.1109/BigData.Congress.2013.59

[34] Apache Hadoop, http://hadoop.apache.org/

[35] LOMOTEY, R. K., AND DETERS, R. 2013. RSenter: Terms Mining Tool from Unstructured Data Sources. International Journal of Business Process Integration and Management (IJBPIM), 2013 Vol.6, No.4, pp.298 - 311, DOI: 10.1504/IJBPIM.2013.059136.

[36] R. K. LOMOTEY AND R. DETERS, “Analytics-as-a-Service (AaaS) Tool for Unstructured Data Mining”, Proc. of the 2014 IEEE International Conference on Cloud Engineering (IC2E 14). pages: 6, March 10-14, 2014. Boston, Massachusetts, USA.

[37] R. K. LOMOTEY AND R. DETERS, “Towards Knowledge Discovery in Big Data,” Proc. of the 8th IEEE International Symposium on Service-Oriented System Engineering (IEEE SOSE 2014), Pages 11, April 8-11, 2014 - Oxford, UK

[38] Q. WANG, L. WEI, AND R. A. KENNEDY, "Iterative Viterbi Decoding, Trellis Shaping,and Multilevel Structure for High-Rate Parity-Concatenated TCM". IEEE TRANSACTIONS ON COMMUNICATIONS 50: 48–55.

[39] R. SHINGHAL AND G. T. TOUSSAINT, "The sensitivity of the modified Viterbi algorithm to the source statistics," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-2, March 1980, pp. 181–185

[40] R. K. LOMOTEY AND R. DETERS, “Analytics-as-a-Service Framework for Terms Association Mining in Unstructured Data”, International Journal of Business Process Integration and Management (IJBPIM), 2014, Vol. 7, No. 1, 2014 , pp.49-61

[41] R. K. LOMOTEY AND R. DETERS, “Terms Extraction from Unstructured Data Silos,” Proc. of the 8th IEEE International Conference on System of Systems Engineering (SoSE), pp.19-24, 2-6 June 2013, Maui, Hawaii, USA.

APPENDIX I: TERMS CLUSTERING RESULT ON PSYCHIATRY

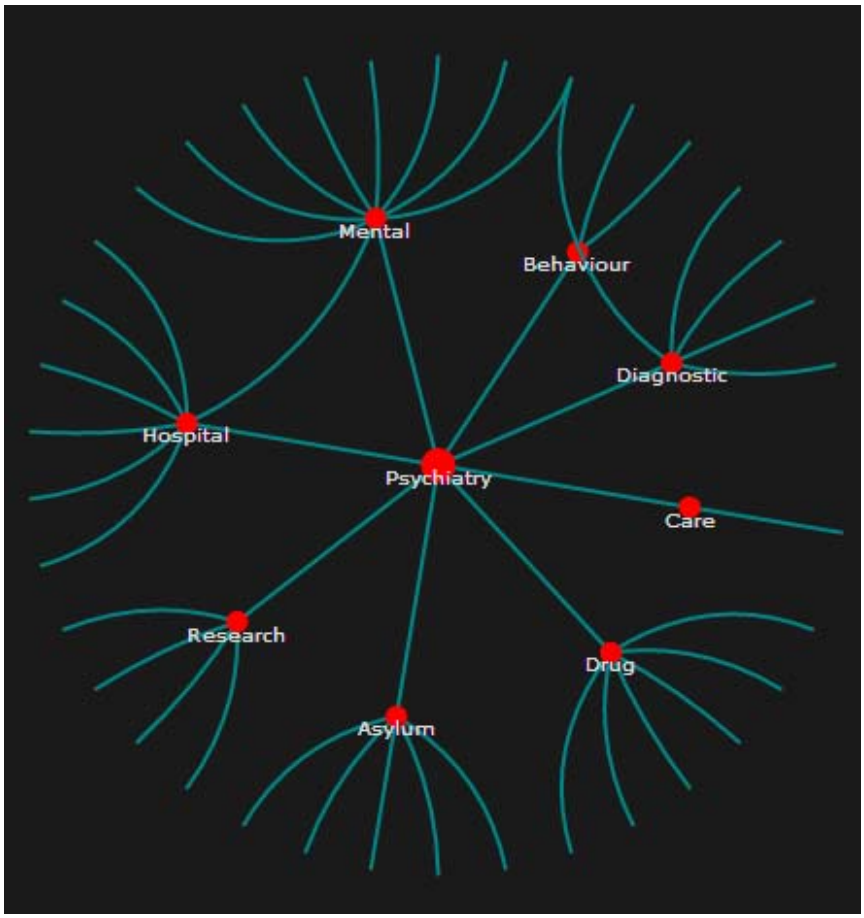

Figure 3. Clustering Visualization of Eight (8) most Ranked Terms using the Viterbi Algorithm