ISBN: 978-981-94-1156-6
2024 the 14th International Workshop on Computer Science and Engineering (WCSE 2024)
doi: 10.18178/wcse.2024.06.018

Semantic-based Big Data Integration with Apache Spark

Nang Kham Soe [1, +] and Myat Pwint Phyu[1]

[1] University of Information Technology, Yangon, Myanmar

Abstract. Providing a consistent and unifying view of all data is a challenging task in the big data context because each big data store has its own data model and permits flexible schema. Moreover, advanced technologies (such as Hadoop MapReduce, Apache Spark) are needed to be able to leverage the integration process of big data. Although differences in data models can be solved by using Apache Spark, it cannot be used for integrating data with different data schemas. For these reasons, a semantic-based data integration approach is proposed to provide a unified view of data in different big data stores. The approach generates schemas by means of local ontologies for data in different big data stores and merges extracted ontologies by using the proposed alignment algorithm. Then, the global (integrated) ontology is converted to the schema using the Spark Dataset API. The schema is used in the data integration step. The main steps of the proposed system are to align local ontologies for global ontology construction, convert global (integrated) ontology to schema in the form of Apache Spark Dataset, and integrate data by applying the schema. The proposed approach is implemented on top of the Apache Spark framework, and the study uses Apache Cassandra and MongoDB as big data stores. Experimental evaluation is conducted to verify the accuracy of the proposed approach.

Keywords: Big Data Integration, Apache Spark, NoSQL databases, Ontology

1. Introduction

With the proliferation of digital technologies, businesses and organizations are generating vast amounts of data every day. These data may be stored in different big data stores based on their needs. Data in different big data stores needs to be used and provide a unified view to get precise insight in some domains (i.e., academic, transport, energy). The NoSQL data systems [1] are innovative database technologies for storing big data. NoSQL data systems allow adding new data records with different schemas to the same table dynamically. There are four main types (document-oriented, columnar, key-value, and graph) of data models in NoSQL databases [2]. Document-oriented data model provides data entities as documents. The documents may be JSON, BSON, or XML format. Examples of document-oriented NoSQL data stores are MongoDB and DynamoDB. The columnar model provides for storing data in columns instead of rows. Apache Cassandra implements such type of data model. Key-value data model allows the simplest data structure, where the data (values) are accessed by strings called keys. The data stores, such as BerkeleyDB and Riak, are implemented on this model. Graph data model allows data entities to be nodes in a connected graph, and relationships between each entity are expressed by properties and labels. The popular graph-based NoSQL databases are Neo4J and InfoGrid.

Providing a unified view of data is a challenging task in the big data context because of the heterogeneity of data models and data semantics (schema) in big data stores [1]. Heterogeneity problems can be addressed by using ontology. Ontologies are formal representation, have been used in data integration systems because they provide an explicit and machine-understandable conceptualization of a domain [3]. With the volume and variety challenges in the big data context, data integration task requires a combination of advanced technologies (such as Hadoop MapReduce, Apache Spark). Apache Spark [4] provides an interface that allows to program large clusters with implicit data parallelism. Data from different data sources can be translated into a built-in data structure (immutable lists, data frames) with Spark.

The proposed system is implemented using Apache Spark to homogenize data in different data stores and improve the performance of the system. Apache Spark native data integration operators allow data with the

[+] Corresponding author.
E-mail address: nangkhamsoe@uit.edu.mm.

same schema to be integrated. Therefore, we apply ontology in the common schema extraction phase for big data contexts. In this study, MongoDB (document-oriented NoSQL database) and Apache Cassandra (columnar NoSQL database) are used as data sources. In order to provide a unified view of data in NoSQL data systems, the proposed system follows four steps. These are homogenizing data from different data stores, common schema extraction using ontology, transforming formal representation of schema to spark dataset schema, and integrating data using spark dataset schema.

The remainder of the paper is organized as follows: Section 2 explains Apache Spark, and the ontology-based big data integration is described in Section 3. The literature survey is presented in Section 4, and the proposed system is explained in Section 5. Section 6 describes the testing environment and experimental results. Finally, the conclusion is in Section 7.

2. Apache Spark

Apache Spark is an open-source cluster computing platform for batch and real-time data processing. Apache Spark's key feature is in-memory cluster computing, which boosts an application's processing speed. It's built to handle a variety of workloads, including batch applications, iterative algorithms, interactive queries, and streaming [5]. Spark effectively hides the complexities of distributed processing behind a convenient API. Data sources can be translated into immutable lists (data frames, dataset) and then transformed with a declarative API based on functional programming primitives (such as map, fold, and groupBy). Spark will split data in the background, distribute the partitions to a cluster of machines, optimize user-provided computations to reduce data movement, and apply them in parallel [4]. It can be used as a standalone framework on a single machine, with one executor for each CPU core. A cluster manager, such as Spark's standalone cluster manager, YARN, or Mesos, will manage the cluster of servers that Spark will use to execute tasks. In the Apache Spark framework, the data is read into a dataset (Spark's main data structure) by the Spark Context object. Spark can read input files, automatically deduce the schema, and load it as a native dataset using Spark SQL.

3. Ontology-based Data Integration

An ontology specifies concepts as well as the relationships that exist between them in the specified domain. Ontologies are used in data integration for five applications [3]:

- Local ontology construction for each data source's metadata (source schemas),
- Global ontology construction for providing a conceptual view of the schematically heterogeneous source schemas,
- Supporting high-level queries (building a query without specialized knowledge of the different data sources),
- Declarative mediation (global ontology is used as a declarative mediator for query rewriting between peers),
- Mapping support by providing a thesaurus, formalized in terms of an ontology.

Our system applied ontology for local ontologies construction and global conceptualization in common schema extraction.

4. Related Works

Some authors presented ontology-based approaches to solve heterogeneity problems (data models, semantics (schema)) in big data integration. Cure et al. proposed a data integration system [6] to retrieve information effectively. The proposed system has been implemented using Apache Cassandra and MongoDB to overcome different data models and schemas of these data stores. They create local ontologies and build a global ontology based on the local ontologies' alignment results. To harmonize the two local ontologies, they used several alignment methods. They enriched local ontologies for alignment using the IDDL reasoner. Then, for simple correspondences, they applied three alignment methods (OWL-lite Alignment, AROMA [7], and JWNL Alignment). They adopted graph formalism for complex correspondences. Abbes et. al presented MongoDB-based big data integration approach using modular ontologies [8] to address structural (data schema) heterogeneity. In matching local ontologies step for generating global ontology, the Levenshtein

distance dissimilarity function is used for string matching and Wu and Palmer similarity measure are adopted for discovering structural similarity. Their approach used MongoDB as data sources and generated OWL as target data format. Mountasser et al. proposed a semantic-based big data integration framework with the aim of addressing big data challenges [9] and the variety of data schema problems. Their approach parses the entities in local ontology and stores them in HBase. In the matching steps, a clustering approach with several alignment methods (language-based, string-based, and graph-based) is used. This framework was implemented on top of Hadoop MapReduce and allowed structured, semi-structured, and unstructured data as input data. Finally, the complex correspondences are discovered by graph formalism. The proposed framework extracts OWL as the target data format. Stripelis et al. proposed a virtual mediation layer for data integration on top of Apache Spark [10] while considering the different data format. Data in PostgreSQL DBMS, XML data in the eXtensible Neuroimaging Archiving Toolkit (XNAT), and data in MySQL DBMS are used in the system.

Among the related works, big data processing tools are used for improving the performance of the system in the two systems [9-10]. And all above mentioned works applied ontologies in common schema extraction and data integration phases. Our proposed system is implemented on top of Apache Spark to overcome data models heterogeneity problem and only applied ontologies for common schema extraction because of limitation for the size of the memory in which data (triple format) level integration. Moreover, the study presents semantic-based approach to solve data semantics(schema) heterogeneities problem.

5. The Proposed System

To provide a unified view of data in different NoSQL databases, the proposed system follows three tasks: These are homogenizing data sources, sematic-based common schema extraction, and data integration by applying common schema. It uses MongoDB and Cassandra as the data sources. The proposed system is implemented on top of Apache Spark to overcome the heterogeneity challenge in the data model of NoSQL databases (MongoDB, Cassandra). The overall system architecture is depicted in Fig 1.

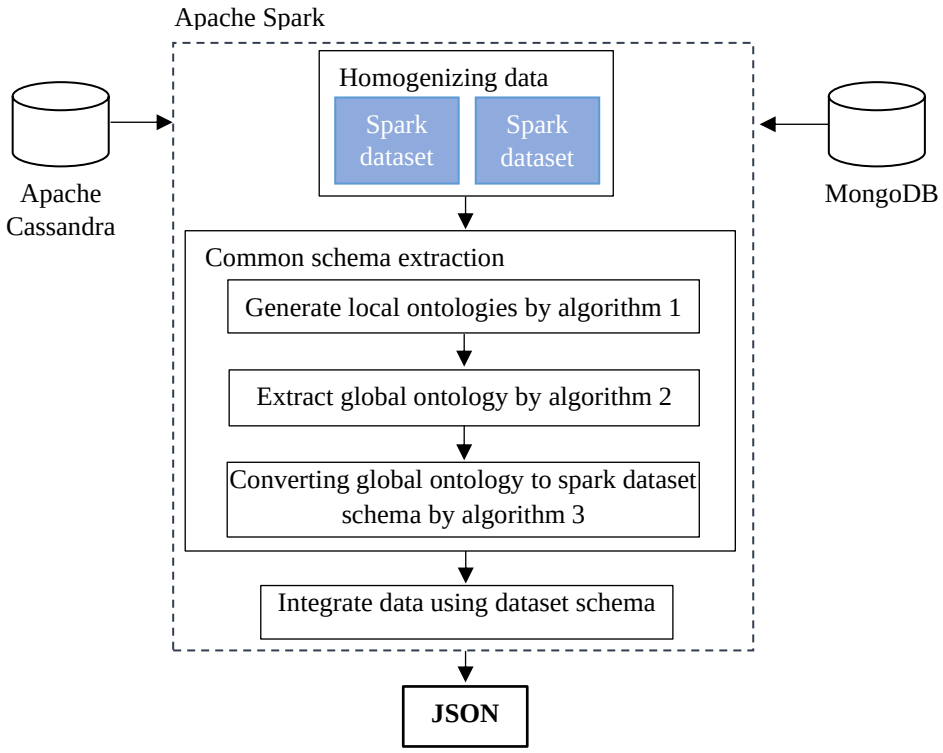

Fig. 1: The overall system architecture.

5.1. Homogenizing Data

Data are loaded from MongoDB and Apache Cassandra onto the Apache Spark framework as native datasets using Spark SQL. The Spark native dataset is formed as a table in the RDBMS. Common schema extraction and data integration processes are performed on these native datasets. The number of datasets

depends on the number of loaded tables from Cassandra and the number of loaded documents from MongoDB to be integrated.

5.2. Semantic-based Common Schema Extraction

Each data source may contain data with different schemas in the same domain based on application needs. Moreover, NoSQL databases allow different schemas in the same table. Apache Spark native data integration UNION operator cannot integrate if it is not in the same schema. The JOIN operator integrates all columns from both datasets, but it does not consider column duplication. Therefore, a semantic-based common schema extraction algorithm is proposed in this system. There are three main steps in this task: generating local ontologies from each source, extracting global ontology by merging local ontologies, and converting global ontology to the Spark dataset schema.

5.2.1. Generating Local Ontologies from each source

In this step, schemas from each dataset are mapped to local ontologies using the mapping algorithm. Spark dataset schema forms the same structure as schema of table in Relational Database. To generate local ontology, the proposed system creates a class for each data set. Then, columns that have primitive types are mapped to data properties of the class and columns that have complex types (array, struct, map) are transformed into object properties of the class. Then, annotation for the object property is added based on their type. After that, the created class is inserted into the ontology. After this step, the two local ontologies for loaded data from Cassandra and data from MongoDB are extracted.

Algorithm 1: Mapping algorithm

Input: spark datasets
Output: Local Ontology
BEGIN
FOR each dataset
Create a class.
Transform columns that have primitive types into dataProperties of the class.
Map columns that have complex types(array,struct,map) to objectProperties of the class and annotate objectProperties based on its type .
Add created class in the ontology.
END FOR
END

5.2.2. Extracting Global Ontology by Merging Local Ontologies

Before merging the two local ontologies, the proposed system aligns each class of ontology with every class of other ontology. To do so, the study firstly compares two strings for class names, data properties names, and object properties names of two different local ontologies by using the Jaccard similarity measure. Then, the two different strings are semantically compared by using WordNet [11].

The ontology class may contain data properties and object properties. The proposed system considers the names and types of the data property for the similarity of two data properties calculation of the two different classes δ (c.dPro , c'.dPro) as presented in equation 1.

$$\delta(c.dPro, c'.dPro) = 0.5 * \delta(c.dProName, c'.dProName) + 0.5 * \delta(\, c.dType\, , c'.dType) \quad (1)$$

The object properties of the two classes in different local ontologies are compared by using names and ranges of objectProperties as expressed in equation 2.

$$\delta\,(c.oPro, c'.oPro) = 0.5 * \delta(c.oProName, c'.oProName) + 0.5 * \delta(\, c.Range\, , c'.Range) \quad (2)$$

The ontologies alignment procedure takes the two local ontologies as input and returns the alignment result as expressed in algorithm 2. The alignment result is an array of the most similar class pairs for each class. After calculating the similarity of each class of the two ontologies, the classes pair which has maximum similarity in the alignment result is selected.

The study applies the cosine similarity measure for comparing two different classes calculation. Therefore, the two classes are needed to be transformed into the two vectors based on the alignment result (similarity scores of classes names, similarity scores of each data property, and similarity scores of each

object property). After transforming vectors for the two selected classes, the similarity score of the two different classes δ (c, c') is calculated by using equation 3. The measure finds the dot product of the vectors divided by the product of their lengths to get similarity of two classes.

$$\delta(c, c') = \frac{\vec{c}.\vec{c'}}{\|\vec{c}\| \cdot \left\|\vec{c'}\right\|} \quad (3)$$

Algorithm 2: Ontologies alignment algorithm

```
Input: Two ontologies o and o′
Output: AlignResult
BEGIN
    FOREACH class c of o DO
      FOREACH class c′ of o′ DO
        Calculate δ(c.dPro,c'.dPro)  by equation 1
        Calculate δ(c.oPro,c'.oPro)  by equation 2
        Calculate δ (c, c′) by equation 3
        Set c, c′ and δ (c, c′) to simpairs
      END FOR.
        Set max (simpairs) to ALignResult
    END FOR
END
```

The classes in the two ontologies are merged based on the similarity score of the two different classes to generate global ontology. The classes merging procedure has two conditions to merge the two classes. The two classes c and c′ are merged into the common class in the global ontology if the similarity level of the pair is 1.0. This means that the two schemas are completely the same. The two classes c and c′ are merged after doing some update c if the similarity level of the pair is less than 1.0 and greater than or equal to 0.5. The classes pair which similarity levels below 0.5 are not merged. Each class of these pair is created in global ontology.

5.2.3. Converting Global Ontology to Spark Dataset Schema

In this step, the system converts global ontology to spark data schema because data integration process implemented by Spark Dataset API. The transforming ontology to spark dataset schema procedure is described in algorithm 3. The schema will be used in the data integration phase as the common schema from two different sources.

Algorithm 3: Transforming ontology to spark dataset schema algorithm.

```
Input: Global ontology
Output: spark dataset
BEGIN
  FOR each class of the global ontology
    Create a dataset.
    Transform data properties of the class into columns of the dataset and the types of columns is based on type of
data properties.
    Map object properties of the class to columns of the dataset and the types of columns is based on annotation of
object properties.
   END FOR
END
```

5.3. Data Integration by Applying Spark Dataset Schema

By using spark dataset schema, data from different sources are integrated into JSON file. The file is ready to use for later tasks in data management.

6. Experiments

For our experiments, Apache Spark version 3.0.1 on Windows 10 Pro machine equipped with an Intel Core i5-950 processor at 2.8 GHz and 8GB of RAM are used. The proposed system is deployed by standalone cluster mode using 3 Spark executors allocating 2 cores and 2GB of RAM for each one. As big data stores, Apache Cassandra 3.10 and MongoDB 3.6 are used. OWL API 5.0 version for ontologies

construction is used. As a testing dataset, this study uses MAG data (papers, authors, venue) and AMier data (papers, authors, venue) that are available in [12]. To verify accuracy of the proposed semantic-based common schema extraction approach in terms of precision, recall and F-measure as shown in Fig 2. We observe that the accuracy result is the same when setting threshold (0.7,0.8, 0.9) in the matching stage. The lower and higher threshold may affect the accuracy result. Therefore, the proposed system will be used threshold (0.8) as optimal threshold value for matching ontologies.

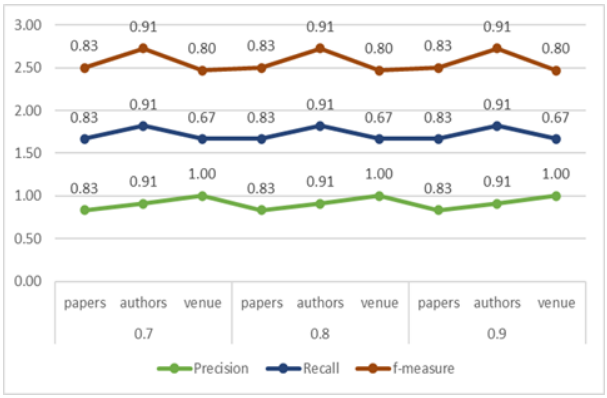

Fig. 2: The accuracy of proposed common schema extraction approach.

7. Conclusion

Providing a unified view of data integration in different NoSQL databases is a challenging task because of the heterogeneity of data models and data schemas. Moreover, data integration for big data context is required with the aid of big data processing framework. Therefore, a semantic-based big data integration approach is proposed. The proposed system applies ontology in the common schema extraction phase and implements on top of apache spark. Evaluation is conducted to verify the accuracy of the common schema extraction by means of Precision, Recall, and F-measure. The result shows that accuracy is steady when setting threshold (0.7, 0.8, 0.9). Therefore, the study choses threshold value (0.8) for our proposed system. As a future work, the accuracy of the proposed approach will be verified by other datasets and the performance of the system will be observed.

8. References

[1] Amghar, Souad, Safae Cherdal, and Salma Mouline. "Data integration and nosql systems: A state of the art." In Proceedings of the 4th International Conference on Big Data and Internet of Things, pp. 1 -6. 2019.

[2] Amghar, Souad, Safae Cherdal, and Salma Mouline. "Which NoSQL database for IoT applications?." In 2018 international conference on selected topics in mobile and wireless networking (mownet), pp. 131-137. IEEE, 2018.

[3] Cruz, Isabel F., and Huiyong Xiao. "The role of ontologies in data integration." Engineering intelligent systems for electrical engineering and communications 13, no. 4 (2005): 245.

[4] Gousios, Georgios. "Big data software analytics with Apache Spark." In Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings, pp. 542-543. 2018.

[5] Apache sparkTM - unified engine for large-scale data analytics. Available at: https://spark.apache.org/. (Accessed: 09 May 2024).

[6] Curé, Olivier, Myriam Lamolle, and Chan Le Duc. "Ontology based data integration over document and column family oriented NOSQL." arXiv preprint arXiv:1307.2603 (2013).

[7] David, Jérôme. "Association rule ontology matching approach." International Journal on Semantic Web and Information Systems (IJSWIS) 3, no. 2 (2007): 27-49.

[8] Abbes, Hanen, and Faiez Gargouri. "MongoDB-based modular ontology building for big data integration." Journal on Data Semantics 7 (2018): 1-27.

[9] Mountasser, I., Ouhbi, B., Hdioud, F. and Frikh, B. "Semantic-based Big Data integration framework using scalable distributed ontology matching strategy." Distributed and Parallel Databases 39 (2021): 891-937.

[10] Stripelis, Dimitris, Chrysovalantis Anastasiou, and José Luis Ambite. "Extending apache spark with a mediation layer." In Proceedings of the International Workshop on Semantic Big Data, pp. 1-6. 2018.

[11] https://wordnet.princeton.edu/.

[12] https://www.openacademic.ai/oag/.