International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 14 (2022) pp. 347-362
© MIR Labs, www.mirlabs.net/ijcisim/index.html

Received: 27 August, 2021; Accepted: 10 May, 2022; Publish: 3 June, 2022

Big Data System of Research Data in The Informatics Department Based on Software Enhancement

Gunawan Budiprasetyo[1], Yoppy Yunhasnawa[2], Mustika Mentari[3] and Dito Cahya Pratama[4]

[1] State Polytechnic of Malang,
Jl. Soekarno Hatta No.9, Malang 65141, East Java, Indonesia
Telp: (+62341) 404424, Fax: (+62341) 404420
gunawan.budi@polinema.ac.id

[2] State Polytechnic of Malang,
Jl. Soekarno Hatta No.9, Malang 65141, East Java, Indonesia
yunhasnawa@polinema.ac.id

[3] State Polytechnic of Malang,
Jl. Soekarno Hatta No.9, Malang 65141, East Java, Indonesia
must.mentari@polinema.ac.id

[4] State Polytechnic of Malang,
Jl. Soekarno Hatta No.9, Malang 65141, East Java, Indonesia
ditocahyapratama717@gmail.com

***Abstract*: Research data between Indonesian institutions such as universities are mainly managed in an isolated way between research groups, thus creating difficulties in unveiling potentially rich insights within the groups. There would be considerably arduous tasks to explore research evolutions and formulate novelties using the current systems. Therefore, an urgent software enhancement in the form of a platform with various additional features is required to share data in a secure and controllable environment as well as to encourage the accountability and reproducibility of the research data. This paper presents a big data architecture, which is a scheme of big data system with four components for the development, which exhibits capabilities to store, query, download, and cite heterogeneous research data, at the same time giving data proprietors to get control of their assets. Review of designs are done from various implementations of web, mobile and institutional platforms where large data management are applied to develop big data system (BDS) functions and its constituents. BDS is a system that contains technologies that process and analyze a large amount of data. The subsequent design has demonstrated an effective front-end component and adequate back-end component by utilizing five core capabilities that exhibit BDS control and data management by using the IMDb and IMDb+ data sets to predict the popularity of films with several classifiers. The adoption of specific API has also given further advantage which gives more controls in the navigation system uses SPARQL query federation.**

***Keywords*:** Big Data, Research Data, Decoupled**.**

I. Introduction

Many informatics departments in universities manage for improving the ability and focus of academic staffs by grouping them into research groups. Also, the existence of group of studies assists students in getting relevant topics for their projects. Each research group regularly conducts academic works and contrives research data annually, and heterogeneous data are generated from these research activities. Not to mention, the external research data are produced by other universities, governments, and industries. This may lead to a rich set of high-volume, high-variety, high-velocity, high-veracity, and high-value of data of research activities [1]. Moreover, each research group usually restrains their own data. However, herein lays the drawbacks. It is considerably cumbersome to discover historic research data, to evaluate evolution of the research, and to formulate novelties of future research. This can potentially cause research recurrence and a lack of innovation. It is envisaged to remove these barriers by providing an adequate research data management to deal with archiving, compliance, security, privacy, sharing, and reuse [2]. This collaborative data sharing can reveal the potential benefits of the data and to allow more researchers to engage. The use of more data from different sources may derive additional knowledge and lead to improve the result of knowledge discovery processes [3]. Eventually, the data sharing may yield externalities and preserve innovation, while encouraging digital literacy and paving the ways for data-driven innovation and wealth for the common good of society.

Extensive works exist for storing, querying, and analyzing heterogeneous data in a timely manner. Big data technologies are maturing and easing researchers to produce valuable insights from large and un-uniform data. When it

comes to supporting the organization's goals, big data presents substantial opportunity for more effective decision-making [4]. According to Yang et al. [5] employed one of big data platforms, Apache Spark, to identify depression level of Facebook users. The selection of big data technology tools in research by Traina et al. [6] is suitable for overcoming the characteristics of big data techniques, namely variations and correctness of large and complex data needs to execute. It makes extracting quantitative data to acquire qualitative data information in research easier [7]. Additionally, it also helps researchers to extract various biological contexts and diseases [8], profiling technology as a pre-discovery opportunity and the development of clinical drugs to improvise and improve their performance like paper [9]. However, these advantages are overshadowed by concerns of data access, control, citation, and ownership. As the number of research data studies is increasingly growing, these concerns may lead to deterioration of the full potential of big data technologies in managing research data.

The Registry of Research Data Repositories (re3data) is one of the approaches to address challenges regarding the growing number of data repositories, especially in the following problems [10]: (i) to store data in long-term preservation and (ii) to find relevant archived data. The re3data provides information of more than 1,000 cross-domain data repositories [11]. Only four Indonesian data repositories are registered at the re3data. They are USU Institutional Repository/University of North Sumatra Institutional Repository, RIN Data Repository, Southeast Asian Climate Assessment & Dataset, and mycoCLAP. However, all of them do not present data access, control, citation, and ownership at the same time. In addition, they are solely focused to archive research papers and not research data. This is arguably not suitable for data integration and interconnection between one part and another as a form of research collaboration.

Following the challenges aforementioned, this study presents a big data architecture, which is a scheme of big data system (BDS) for managing research data. It allows students, academic staff, and external parties to exchange and share data with one other, and primarily, they can securely control their data and ability to view, access, query, and download by others. Also, it provides a citation feature to promote their data to cite for accountability and reproducibility.

There are numerous novelties to this study. First, the methodology can assist universities or industries in taking the essential steps to establish a big data system to manage their vast amounts of complicated data. Another notable feature of this research is the inclusion of Hadoop, HIPI, HBase, Phoenix, OAuth2, DCAT, and SPARQL Federated Query in the development process. The methodological novelty of this work is expected to increase as a result of the hybrid strategy. Moreover, with the help of using HBase and Phoenix approach, querying of multiple different files such as: csv, json, sql, txt, and xls using SQL language can be performed. Additionally, the main benefits of accomodating SPARQL endpoints is the ease of use and flexibility to extend and link internal datasets and external datasets. This situation provides opportunities to perform SPARQL federated query to add background knowledge into existing datasets.

II. Literature Review

The practice of data sharing among researchers is a common activity in science [12]. In the university environment, in which academic staffs demonstrate responsibilities to conduct researches and deliver learning and teaching processes, they require a special discussion about research data and how to discover these to support research publications. For instance, provided simulation resources in a digital repository for students of the nursing faculty to receive education and knowledge in the related subjects [13]. Chard et al. [14] term the simplest system to manage research data as the Legacy Research Data Portal (LRDP), which is an architecture of web application that performs uploading and downloading research data on its data repository based on user requests. LRDP spreads to various applications for various purposes under different names, such as the following: portals, science gateways, hubs, and Web Observatories. Portal is a system built using standard web technology that provides useful resources, involving several stakeholders in the web portal so that consistency of many researchers/ stakeholders can work together and access it more easily [15]. While Lawrence et al. [16] defines science gateways as digital platforms, either web or mobile applications, and based on sophisticated technologies to support communities to perform collaborative research. Wachs et al. [17] collaborating with support tools and hubs as a physical prototyping platform that proves the concept of collaboration and utilization of one centralized resource. According to Tiropanis et al. [18] defines Web Observatory (WO) to engage Web Science researchers meaningfully with research data.

Next, the basic requirements for research do not only deal with managing incoming and outgoing data like a data repository. Yet, it also requires additional access, control, and ownership at the same time. A set of these features can be found in the Web Observatory (WO). According to the previous studies, the WO is a web platform which demonstrates functionalities and capabilities as follows: (i) to share, collect, and analyze data on the web [19], (ii) to archive data on the web in a distributed fashion [18], (iii) to be middleware for generating complex data from different sources [18], [19], (iv) to harvest, query, and analyze various real-time and historic heterogeneous data as well as to allow data owner’s access control to their datasets [18], [20]. Aljohani et al. [21] reveals insight into Web Observatory in the past, recently exists, and also WO in the future. Table 1 summarizes existing WOs that are created or published by universities (Higher Education) as listed in the WO website on the Web Science Trust [22].

Hadoop is not only for storing large data but also comprises technology stacks to integrate data storage, data processing, and system management. Therefore, Hadoop is appropriate to select as a solution in the system level. Existing Relational Database Management System (RDMS’s) is very powerful in managing structured data; however, they do not confront to handle big data challenges. In the Hadoop technology stacks, NoSQL databases are growing into a standard to address the problems of big data [23]. Based on the data model, NoSQL can be classified into three primary types: k-value stores, column-oriented database, and document databases. HBase is categorized into the category of column-oriented database and

widely used by many IT companies, such as the following: Facebook, LinkedIn, and Twitter. It is built on the top of Hadoop Distributed File System (HDFS), providing capabilities of storing on multiple regions, data reapplication, and fault tolerance.

Next, two libraries to support the stacks implementation, Phoenix and HIPI, are reviewed and used in this work. Phoenix is a SQL layer to insert data into HBase and supports in parallelizing on multiple regions of a table leading to better performance of HBase compared to RDBMS. Phoenix enables performing parallelism over the regions in a region server and supports secondary indexes as well as targets low latency query over HBase tables using SQL queries [24], [25]. HIPI stands for Hadoop Image Processing Interface. It is developed using MapReduce technology to process large-scale images in a distributed environment [26], [27]. To improve the performance of MapReduce in handling images, HIPI creates a HIPI Image Bundle (HIB) to store multiple images in one large file [28]. HIPI administers a large set of images by providing two files: (i) a data file containing set of images put together and (ii) an index file containing information of images location in HIB.

Various work such as BDS research datasets have been implemented. However, it is necessary to adjust the BDS according to the needs of data availability and data processing in a system that is adapted to the research data of the informatics department. The proposed BDS architecture features a more comprehensive mix of several BDS literature studies that have been discussed. BDS research data from the informatics department can be accessed by many users with different preferences. Access data from one system, as well as process the data using available APIs.

III. METHODOLOGY

The general overview of BDS is shown in Figure 1. BDS contains two types of users, namely registered and anonymous. Registered users can upload a dataset, which in turn is called the owner. The owner can set the dataset access permissions into three types, namely public, displayed, and private. The displayed and private ones can only be utilized by registered users. To use these type of datasets, registered users must ask permission from the owner of the dataset, and if allowed, the registered user can take full advantage of the dataset. These advantages are being able to view, query, access, download, and cite datasets.

BDS is proposed to exhibit seven important features. Firstly, it allows data owners to upload their various heterogeny datasets and regulate their permissions whether public, displayed, or private. To access private datasets, users are required to ask for grants from the dataset owners. Secondly, it permits data owners to host their datasets not only in the local storage but also in remote storages, so that, BDS serves to be a middleware to access the data. Thirdly, it enables users to download, visualize, and query datasets. Fourth, BDS supports SPARQL query language. Users exhibit opportunities to make a SPARQL joint statement to integrate data from two or more triple data files stored either in the local storage or remote storage based on a common property between them. BDS also enables the use of SPARQL federated query to merge those files with public SPARQL endpoints. Fifth, BDS accommodates users to use the most popular query language, SQL, to incorporate tabular data from multiple files in the following formats: csv, json, sql, txt, and xls, based on a linking key between them. Sixth, BDS provides a feature for citing a dataset. Therefore, the information about the number of citations can be gathered either per dataset, user, or institution. This way may promote the dataset and improve its accountability and reproducibility. Seventh, BDS generates and publishes metadata of datasets in the machine-readable format. This way enables external web resources to access or harvest the description of the datasets.

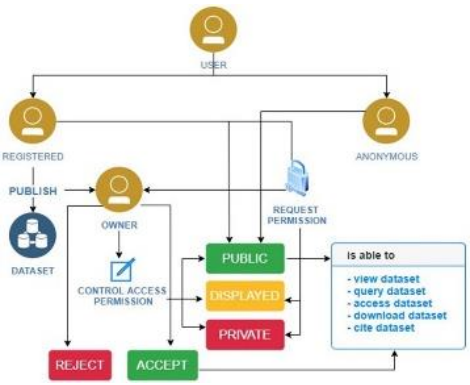

Figure 1. Overview of BDS

Next, all of the features proposed are intended to engage more students, academic staffs, and external researchers to foster new data-driven innovations by providing a secure environment for them to share and collaborate their research data. To make it work, the implementation of BDS is broken down into three main stages, i.e., design, development, and deployment, as shown in Figure 2. The first stage is to put a solid foundation for the further steps of the project. This paper is focused on the design step, and it contributed to result in a schema of the BDS. The scheme will lead the development process of the BDS and is easily replicable to other purposes. The last feature is directing BDS to be able to access publicly by users. To do so, performing a set of testing in pre-production environments is required.

In the design stage, a scheme of BDS is proposed, as shown in Figure 3. BDS is designed as a decoupled architecture to offer three advantages. Firstly, it benefits reconfigurability and scalability. Secondly, it can be a solution to the problems of performance. Thirdly, it simplifies operational complexity. The architecture of BDS is grouped into four different components (component that is usually used for front-end application development): User Application, API, Front-End, and Back-End, respectively. Each component communicates to each other using the API component. Below, each component in BDS will be explained.

Figure 2. Stages of the big data system implementation

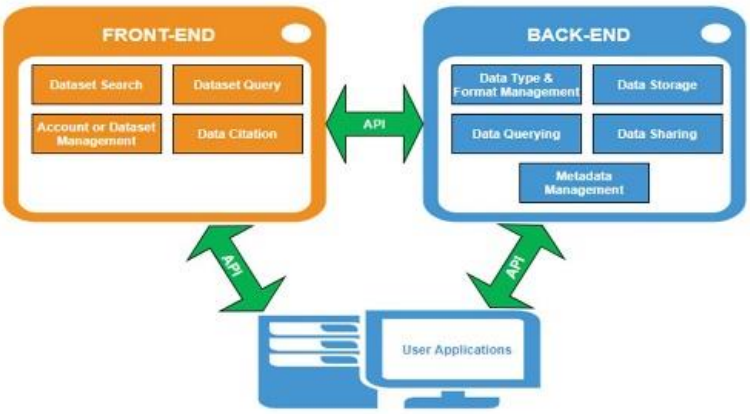

Figure 3. A Scheme of big data system

A. User-application component

This component contains applications, agents, services, etc. belonging to users that take advantage of BDS features. They do not need to download datasets and store them in their local storage. They can read datasets streams via the BDS APIs to do data analytics. To query datasets via the use of API, a user-application requires it to supply three parameters. The first is an access token. The access token exhibits an expiration time and the nature of it is reusable. The second is a dataset ID, and the last is a query statement. Two options are present of queries either of SQL or SPARQL. Once the user sends the query request, it is redirected to the dataset by BDS. The RESTful API will transfer back the result to the user. This approach streamlines the user's efforts in gathering readily data for performing data analytics. Users are advocated to cite datasets, thus respecting the ethical use of data as well as promoting the academic work of the owner.

B. API component

BDS proposes RESTful APIs for the two following goals: to connect among components as shown in Figure 3 and to provide a capability for end-users to query datasets in a secure environment, respectively. The first goal is to handle the operations and configurations of each component in a distributed manner, while the second goal is to allow users to access data whether via direct access on BDS UIs or via programmatic using user-applications.

BDS implements the RESTful API by following the principle of HATEOAS (Hypertext as the Engine of Application State). As argued by Garriga, et al. [29] and Neumann et al. [30], HATEOAS comprises embed hyperlinks and controls, so that it provides a navigation system throughout the service web and eases a client to know what a specific operation can be executed. As an illustration, Figure 4 indicates the comparison of REST and REST-HATEOS. Also, this figure shows that links to view, query, and download the dataset with id 12345 are included only if the user with id "user12345" demonstrates a status of authorized. BDS RESTful APIs allow consumers to locate the resource without the need to demonstrate an upfront understanding of the resource and its relationship.

C. Front-end component

This is the main interface in which users interact with BDS. The development of friendly web UIs is suggested using ReactJS, to provide non-technical or untrained users to visualize data to get understanding of the data. More importantly, the users can take advantages of four main functionalities available on the front-end: (i) to manage accounts or datasets, (ii) to search datasets, (iii) to query datasets, and (iv) to cite datasets. This component is possible to communicate with the back-end component and the user-application component using the API component.

D. Back-end component

This component resides on the server-side and is responsible for handling functionalities and servicing the components of front-end and user-application. In this proposal, the back-end is proposed using Java programming and exhibits functionalities as follows: (i) data types and formats management, (ii) data storage, (iii) data querying, (iv) data sharing, and (v) metadata management.

1) Data type and format management

BDS provides functionality for storing various data formats, such as the following: tabular data (structured and semi-structured) and non-tabular data (unstructured). The structured one allowed to store is SQL, NoSQL, and triple/RDF (Resource Description Framework). The semi-structured ones are tabular data in the formats of csv, text, and spreadsheet, while the unstructured ones are non-tabular data, images, audios, and videos. It follows an industrial standard from Freed et al. and Melnikov [31] to naming media types of various data formats supported in the portal. To access the available datasets in BDS, the naming standard and definitions of the media types are beneficial for informing protocols and procedures.

2) Data storage

To manage data storage, BDS proposes two (HDFS and HBase) of the Hadoop technology stacks. Figure 5 shows the distributed data store architecture of BDS to provide persistent storage for managing and accessing the data. These stacks are employed to ingest and store the various incoming data. Both HDFS and Hbase support distributed and parallel processing to store multiple stream data. This way can potentially minimize I/O or resource bottlenecks. To handle

tabular data in BDS, both structure (SQL & NoSQL) and semi-structure (csv, txt, & xls) data formats, a construction of transformation engine exists to transform these files into valid data formats. Hence, the engine can seamlessly load these inputs into Hbase storage using Data Manipulation Language provided by Phoenix library, such as the following: UPSERT VALUES for the purpose of row-by-row insertion, UPSERT SELECT for large data transfer across the same or different tables and DELETE to remove rows permanently. The engine leverages data definition language provided by Phoenix to perform operations of CREATE TABLE, DROP TABLE, and ALTER TABLE for adding/removing columns.

To deal with non-tabular data both semi-structured and unstructured (images, audio, and video) data formats, BDS manages to feed and insert these data into the HDFS storage. HDFS is set up consisting of three clusters. One is purposed for a name node, while the others are for data nodes. To store these files into HDFS, BDS utilizes the HIPI library for simplifying jobs. For scalability and reliability, BDS does a parallel job of storing these data in the distributed environment. This way keeps efficiency in processing resource-intensive tasks, such as the following: processing large-scale files.

3) Data querying

As a result of selecting HDFS to use, a requirement is present to prepare instrumentations of data representation to view and interact with the data storage. While an extensive demand for MapReduce knowledge was raised in previous methodologies to access and query the data, BDS excludes the need for broad MapReduce knowledge by proposing various data organization and representation approaches to allow quick and easy access to the data. Based on the data formats, data in BDS storage can be grouped into two types: tabular and non-tabular data.

The former is stored in HBase. Next, to provide access to the data via HBase query mechanisms, BDS suggests a SQL Skin on top of HBase, such as Phoenix. While previous approaches would require HBase scans to query the data in HBase storage, our approach is to perform distributed queries over HBase using SQL statements. This approach allows users to query tabular data in semi-structured formats using SQL queries either via BDS UIs or user-applications. Besides SQL, BDS also supports SPARQL queries to extract structured data, like triples.

The latter is media files such as images, audios, videos, etc. stored in HDFS. MapReduce is effective to handle accessing and querying data residing in HDFS. However, it is not a trivial task to represent images in a standard float image using MapReduce. To address this problem, BDS exploits HIPI to access these files from HDFS. HIPI creates HIB for a large set of images, which contains an index file. The unit functions of BDS works by running two important steps. Firstly, the function looks for the image location in the index file of HIB. This is considered an efficient process as it allows us to easily access images across the data file of HIB without being required to read in every image. Secondly, it performs a parallel job to encode and decode a specific image of the HIB in the MapReduce pipeline. This job results in standard float images to present.

Observatory	Hosted By	Features
Southampton Web Observatory (SUWO) https://wobs.soton.ac.uk/	Southampton University	Harvesting, querying, and analyzing multiple real-time and historic heterogeneous data and visualization, while providing data owners access control to their resources
Collaborative Online Social Media Observatory (COSMOS) http://www.cs.cf.ac.uk/cosmos/	Cardiff University	Contains social media research data especially Twitter, privacy protection and deception of data collection, visualization, and dissemination of social media data
OSoMe (awesome) http://truthy.indiana.edu/	Indiana University of Bloomington, USA	Focusing on media and technology in society, and curbing the spread of misinformation online and the manipulation of social media (tools, findings, publications, and resources)
National University of Singapore https://nextcenter.org/	National University of Singapore (NUS) and Tsinghua University of China	Analyzing heterogeneous data (social media) visualization, data analysis (especially unstructured data analysis for finance and marketing)
SONIC Northwestern Observatory http://sonic.northwestern.edu/	SONIC Lab Northwestern University	Provides several inventory datasets such as social media data; journal publication data; etc., external datasets, and also visualization

Table 1. Features of different web observatories

Figure 4. A comparison of REST and REST-HATEOAS

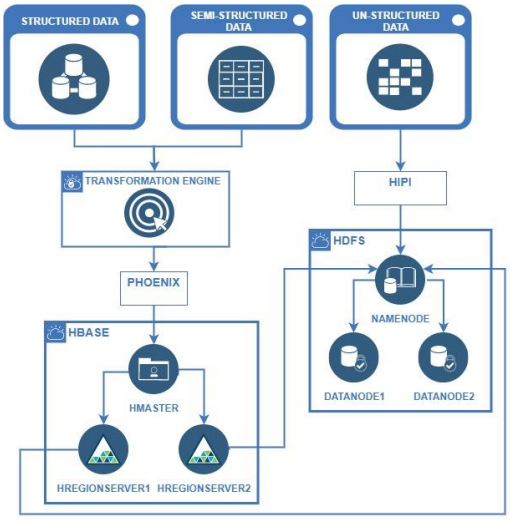

Figure 5. Data storage architecture of big data system

4) Data sharing

The designing of data sharing architecture is classified into two main steps entailing user access controls and API access controls. The former two roles are defined: users and owners. Registered and anonymous are two types of users. BDS only permits registered users to be able to publish datasets and grants them as the dataset owners. The owners demonstrate capabilities of controlling access permissions. They demonstrate full controls over the datasets. The dataset owners exhibit three options to determine the basic permission of their datasets. The first option is public, where the shared datasets allow for any users and applications to view and access. The second one is displayed, where the shared datasets are visible for any users; however, they only can be accessed by authorized users from their owners. The last one is private, where the shared datasets are only visible and accessible by the owners, unless they explicitly grant for a specific user to view and access it. The possibility exists for users to ask permission to access datasets, as long as they are already visible to them. They cannot request the datasets invisible for them. Dataset owners are either granting or denying the request. Each user exhibits a unique view of the catalog of shared datasets. BDS will render the corresponding view according to their permissions.

The latter prominent feature of data sharing architecture is to support accessing APIs for querying datasets in BDS. The query API is designed to ease end-users for performing analytics on top of data and provide better reusing of data. The query API includes two basic components, mechanisms of authentication and authorization and RESTful API. Authentication and authorization are achieved by adopting OAuth2, which was widely used to authorize an application on behalf of a user. OAuth2 defines four different roles [32], as follows: (i) Dataset owner, which demonstrates a full-control of access on a protected dataset, (ii) Dataset server, which hosts the protected dataset, (iii) Client, which is an application that acts on behalf the dataset owner and inherits its authorization, (iv) Authorization Server, which issues an access token to a client if the dataset owner provided the original credential.

The flow of OAuth2 involves five steps. First, a client requests an authorization from the dataset owner using a HTTP protocol. Secondly, if the owner of dataset responds positively, they will grant an authorization code to the client. Third, the client sends the authorization code to authorization server. Fourth, the authorization server receives the authorization code, exchanges it with an access token, and sends it back to the client. Finally, by providing an access token, the client accesses the dataset from resource server. The use of OAuth2 assures that both public and private datasets can be accessed in a secure way using RESTful API.

5) Metadata management

The intention of BDS to publish, share, and integrate research data is to augment values of transparency and self-empowerment to improve efficiency and effectiveness in conducting researches. To reach those full potentialities, BDS

publishes a set of rich metadata. These metadata enable datasets to discover, understand, and integrate by external web resources.

BDS applies a metadata model, namely DCAT (Data Catalogue Vocabulary). DCAT is a catalog for describing datasets and data services, which is built using a standard model and vocabulary, namely RDF [22]. Three considerations to use DCAT are present. Firstly, it eases the usage and aggregation of metadata from different catalogs. Secondly, it enhances the discoverability of datasets and data services as it is published on machine-readable data formats. Lastly, data catalogs can be published in a distributed way, and the possibility exists to make federated query across catalogs in multiple servers as they are built on a standard model and vocabulary.

BDS involves three kinds of important information, such as the following: access, license, and provenance in the metadata catalog. The access information provides users information on how to access data dumps or to query data using endpoints. The license information describes a clear statement related to the legality to use, share, and distribute datasets. For licensing, BDS implements Creative Commons Licenses, as they were used widely in industries. The licenses will be accepted around the world and will never expire as long as the copyright is also applicable (because they are built on copyright). For consideration of permissions, copyrights, and attributions, we accommodate six types of licenses (CC BY, CC BY-SA, CC BY-NC, CC BY-NC-SA, CC BY-ND, & CC BY-NC-ND) [33]. For the provenance information, we include the time when a dataset is generated for the first time. Figure 6 shows an excerpt of the metadata catalog of BDS.

```
@prefix dcat: <http://www.w3.org/ns/dcat#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix ex: <https://dcat.rdp.polinema.ac.id/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix v: <http://www.w3.org/2006/vcard/ns#> .
@prefix prov: <http://www.w3.org/ns/prov#>

ex:data-portal
  a owl:Ontology ;
.
ex:catalog
  a dcat:Catalog ;
  dct:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
  dct:publisher ex:jti-polinema ;
  dct:title "The Catalog of JTI - Research Data Portal"@en ;
  foaf:homepage <http://rdp.jti.polinema.ac.id/catalog> ;
  dcat:themeTaxonomy ex:themes ;
  dcat:dataset ex:dataset-001 ;
.
ex:jti-polinema
  a foaf:Organization ;
  rdfs:label "Information Technology Department of State Polytechnic of Malang
              (JTI - POLINEMA)"@en ;
.
ex:themes
  a skos:ConceptScheme ;
  skos:prefLabel "A set of domains to classify documents"@en ;
.
```

```
ex:dataset-001
  a dcat:Dataset ;
  dct:bibliographicCitation "Mentari, M; Budiprasetyo, G; Yunhasnawa, Y (2020):
      Using the Dempster-Shafer Theory in a Cat Skin Deseases Expert System" ;
  prov:generatedAtTime "2020-01-01T02:52:022"^^xsd:dateTime ;
  dct:issued "2020-05-05"^^xsd:date ;
  dct:license <https://creativecommons.org/licenses/by/4.0/> ;
  dct:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
  dct:modified "2020-05-05"^^xsd:date ;
  dct:publisher ex:jti-polinema ;
  dct:title "Skin Cat Deseases"@en ;
  dcat:contactPoint ex:MustikaMentari-vcard ;
  dcat:distribution ex:skin-cat-deseases-rdf ;
  dcat:keyword "ontology"@en ;
  dcat:keyword "expert system"@en ;
  dcat:theme ex:information_system ;
.
ex:MustikaMentari-vcard
  rdf:type v:Individual ;
  v:fn "Mustika Mentari" ;
  v:hasEmail <mailto:must.mentari@polinema.ac.id> ;
.
ex:skin-cat-deseases-rdf
  a dcat:Distribution ;
  dct:title "RDF distribution of Skin Cat Desease dataset"@en ;
  dcat:byteSize "5000000"^^xsd:decimal ;
  dcat:downloadURL <http://rdp.jti.polinema.ac.id/files/skin-cat-deseases.rdf> ;
  dcat:mediaType <https://www.iana.org/assignments/media-types/application/rdf+xml> ;
.
ex:information_system
  a skos:Concept ;
  skos:inScheme ex:themes ;
  skos:prefLabel "Information System"@en ;
.
```

Figure 6. An excerpt of semantic metadata catalog

IV. Result and Discussion

In this section, we discuss some BDS User Interfaces (UIs) indicating its features as well as two use cases. The first use case is to demonstrate the suitability of BDS to support data access and sharing in building movie classification models, while the last is to exhibit supporting BDS in performing SPARQL query federation.

A. User interface

As the security system is designed to implement OAuth2, BDS allows users to perform a social login using their social media account, i.e., Google, Facebook, and Github. This way does not require users to do a registration process. The registration is only mandatory for a user who does not want to log in using social media. Successful registration permits users to log in by filling the text boxes of username or email and password. To accommodate this scenario, Figure 7 indicates a login page.

BDS presents an UI for managing a dataset which contains two main sections: top (Figure 8) and bottom (Figure 9). As described in data sharing section, only a registered user can input a dataset, so that a user which is logging in will be displayed on top right corner of Figure 8. The top section involves ten inputs: name, description, research group, organization, website, tags, cited as, published, generated, and license. The cite input guides users how to cite a dataset they use. The input of the organization is to specify what organization the data owner is from. According to the cited as, the number of citations per dataset or user can be known. Moreover, it is combined with the information of organization and will produce the total number of citations per organizations. This may benefit for organizations when they are being assessed by an official higher education assessor. The licensing design, as shown in the license input in Figure 8, will explain the copyright grant from the owner of the dataset to the dataset user using a standard way, which is well known in the industry. The addition of the published date and generated date fields in Figure 8 is intended to an UI that supports the provenance of dataset. This way enables it to trace the originality of the dataset owner and to ensure its originality and value.

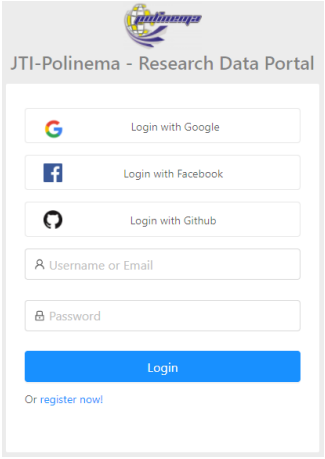

Figure 7. Login page

The bottom section contains three tabs related with supporting files of the dataset and the Query Browser tab. The first tab, local files, consists of a button and a table. The table is designed to display all supporting files of the dataset, which were stored in in HDFS or HBase. BDS can store various files in different formats and sizes, as illustrated in Figure 9. To add a new file is simply done by clicking the Add Files button, while the Edit and Delete buttons in the last column of the table are used to modify and delete an existing file, respectively. Following, pressing the Add Files or Edit button is next, a form as shown in Figure 10 appears. The Add/Edit File form exhibits six inputs: file name, description, file location, alias, permission, and shared to. The two first inputs are for naming and describing a file, while the third is for locating a file to store in the big data storage. The fourth is to determine whether the specific file (csv, json, txt, or xls) is queryable or not. If it is yes then the input is required to be filled with an alias name. For a sql file, the dataset owner does not require it to specify an alias. Either the queryable file or the sql file, BDS, will parse it. Subsequently, it creates tables and loads data into HBase storage. BDS provides the owner to arrange the file permission, i.e., public, displayed, or private. Successful operation will appear in the table on the Local Files tab, as shown in Figure 9. On the Remote Files tab, it provided inputs to locate a remote file and its description as well as notes to use it. This file may be stored in storage of the dataset owner or somewhere in third-party storage. The third tab is to keep favorite images so one of them can be selected to be an image profile of the user.

The last tab is dedicated to browsing the data by supplying queries with either SQL or SPARQL. This feature may be a unique design concept that is distinguished from other research portals especially the ones in Indonesia. The feature can query different file formats using a SQL query. BDS also provides another feature for querying semantic data using a SPARQL query. Figure 11 demonstrates a SQL query to join four existing different files: skripsi_salsa.sql, test_score.csv, interview_score.json, and employee.xlsx, respectively. In Figure 11, vacancy represents a table from the sql file, while employee symbolizes a from the xls file. While the other two test_score and interview_score tables denote the csv file and the json file. Figure 11 shows that our approach can combine data in different formats using a single SQL query, while Figure 12 shows the use of SPARQL to query the ontology file called skripsi_salsa.owl. Figure 11 and 12 exposes that the design of the portal equips users with two powerful query languages to explore datasets published in the portal. These features offer flexibility for users to make custom queries for their own purposes. Also, these allow users to download query results as csv, json, or xls files.

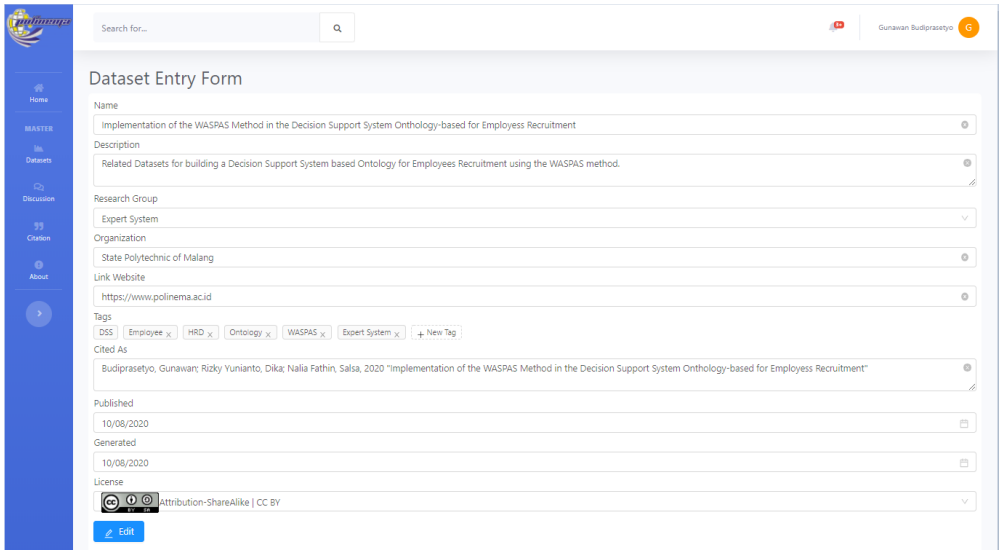

Figure 8. Dataset entry page (top section)

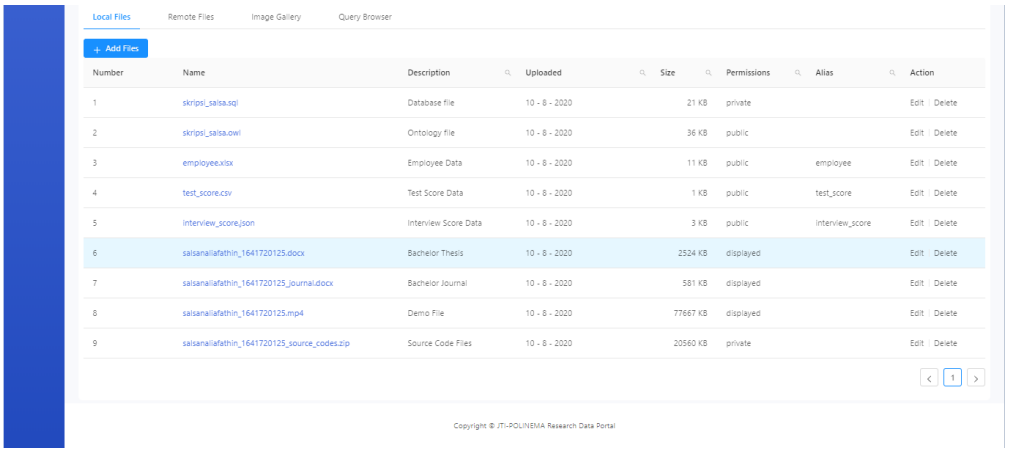

Figure 9. Dataset entry page (bottom section)

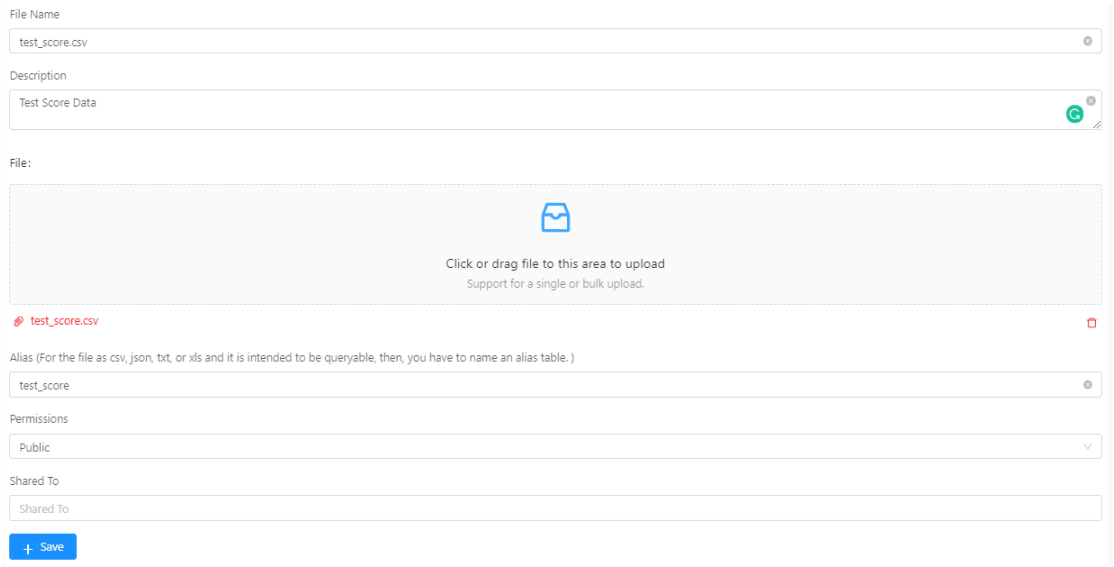

Figure 10. Add file page

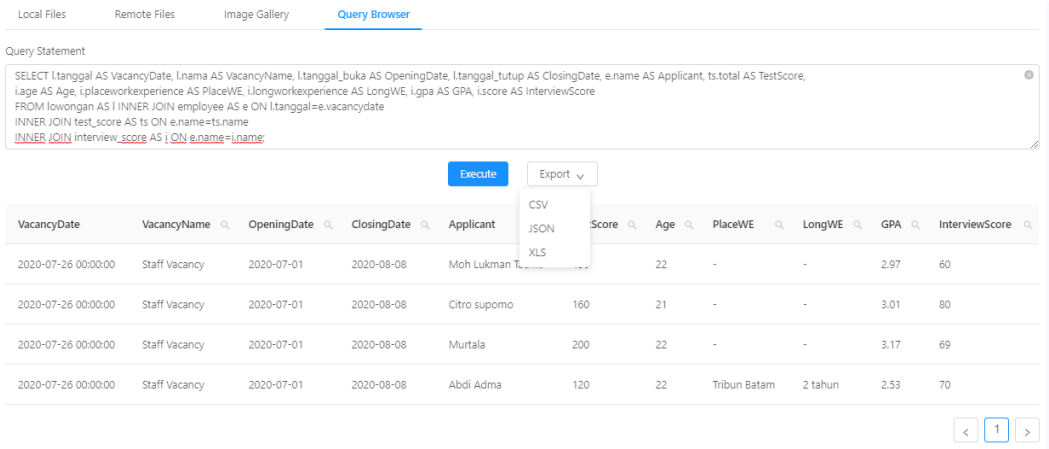

Figure 11. SQL query results

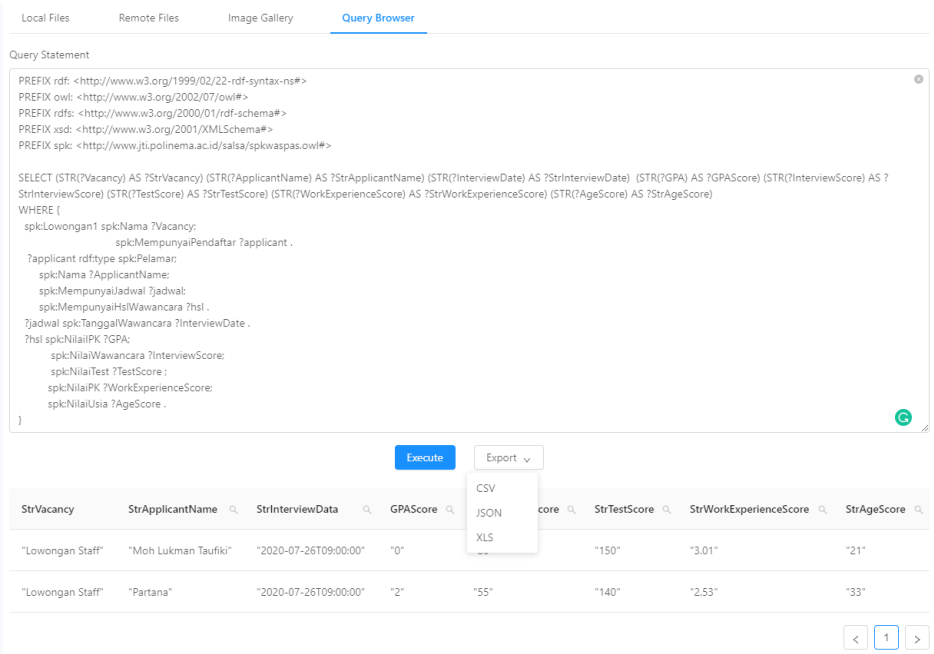

Figure 12. SPARQL query results

B. The use case of movie popularity classifications

To demonstrate this use case in building classification models for movie popularities, we involved two large files stored in BDS: moviesdb.sql and moviesdb.rdf, in which, the sizes of them are around 7 GB. Three steps of knowledge discovery, i.e., data selection, data transformation, and data mining, were performed to generate the classification models.

C. Data selection

The main data source used is from IMDb (ftp://ftp.fu-berlin.de/misc/movies/database/frozendata). There are two considerations to use IMDb dataset instead of datasets of Kaggle, data.world or Academic Torrents. Firstly, the IMDb is arguably the most complete movie data freely available on the internet. Secondly, we intend to integrate the IMDb data wither other related data sources. The high-level schematic of the integration of nine data sources (IMDb list files, DBpedia, Wikidata, and the following six websites: IMDb, BoxOfficeMojo, FilmAffinity, Metacritic, MovieMeter & RottenTomatoes) to create an integrated movie database is shown in Figure 13. The movie database is in the form of MySQL database and can be generated into a set of SQL scripts, namely moviesdb.sql. Subsequently, we write the script in Java, to convert the movie database into a file namely moviesdb.rdf, which contains a set of RDF data. Both moviesdb.sql and moviesdb.rdf are stored in BDS to be used to the purpose of knowledge discovery.

D. Data transformation

We generate two ready datasets for classification purposes: IMDb and IMDb+, which contain notable attributes in affecting the future of movies. All the datasets involved 6,287 movies for analysis and classification purposes. Those movies are gathered from the integrated movie database by applying five filters. The first filter is movies released between 1990 and 2017. As argued by Asad et al. [34] movies released after the year 2000 fell around the same time which delighted in the product of innovation [34]. However, finding movies in the low ratings in these periods is hard. Therefore, we made the range of movies released years wider. We believe that the wider the year range produces more possibilities to get movies with high, medium, and low ratings.

The second and third ones are English movies released in the USA. As suggested by Asad et al. [34], the IMDB list files potentially contain more movie data based on the English language and released in the USA. The fourth one is only movies receiving more than 1,000 user votes. As mentioned by Saraee et al. [35], the fourth filter was applied to eliminate the bias of an unknown movie with a few high votes. The last filter is only movies listed in DBpedia, IMDb list files (movies.lst), and IMDb website. The reason to apply this filter is that DBpedia provides links to Wikidata. Subsequently, Wikidata provides links to the other resources, such as the BoxOfficeMojo, IMDb, and Metacritic websites. Therefore, the consideration to use only movies provided in DBpedia as the data training is for the ease of data integration.

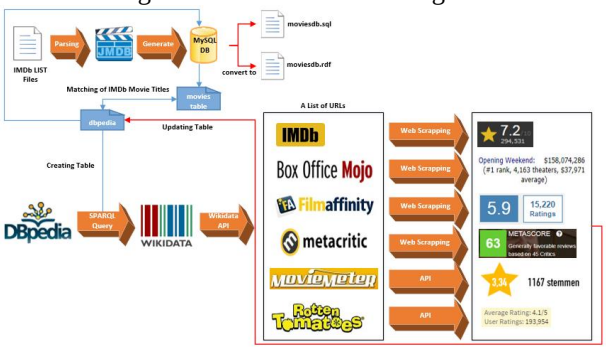

Figure 13. A high-level schema to create the Integrated DB

To build a dataset for the classification purpose, defining input attributes and an output attribute is required. The input attributes are factors possibly recognized as the success factor of a movie to get a high rating from users. Table 2 summarizes the input attributes of the IMDb dataset with a total number of 16 attributes. The IMDb dataset is generated from only one data source, namely the IMDb file list, while the IMDb+ dataset is an extension of the IMDb dataset with the addition of seven new input attributes, as shown in Table 3. Therefore, the total number of attributes of the IMDb + dataset is 23.

The main transformation in the generation of the IMDb and IMDb+ datasets was to calculate the numerical rankings for actors, actresses, and directors, as was performed by Saraee et al. [35] and Asad et al. [34]. Other inherent movie attributes from the IMDb list files that may be useful to predict the future popularity of a movie are cinematographers, composers, costume designers, distributors, editors, production companies, product designers, and writers. To the best of our knowledge, those attributes were not investigated by other researchers as the consideration factors to the future success of movies. We used the sum function only to get a distributor rating, while the other attributes used the average function to get the ratings. We believe more distributors involved in marketing a movie will increase that movie's likelihood of popularity.

Determining an output attribute is mandatory. We considered using user rating as provided by IMDb. IMDb provides a rating scale between 0 and 10, and every IMDb user can vote their favorite movie. Movie rating is a weighted average vote, and the value is continuous. To complete the aim of analysis and classification, we generalized the continuous numeric value of the average of voting into four categories, as shown in Table 3. This approach was used by previous researchers [34]-[36]. The SQL query to generate the IMDb dataset is shown in Figure 14, while the SPARQL query to extract data for composing the IMDb+ dataset is indicated in Figure 15. The popularity of the film has 4 classes as output with details of the rating range in Table 4, which consists of excellent, average, poor, and terrible.

Table 2. Summary of input attributes of the IMDb Dataset

Attribute Name	Math Operation	Values	References
Actor Rank	Sum	Positive Numbers	(Asad *et al.*, 2012; Saraee *et al.*, 2004)
Actress Rank	Sum	Positive Numbers	(Asad *et al.*, 2012; Saraee *et al.*, 2004)

Attribute Name	Math Operation	Values	References
Budget	Discretization	1,2,3, 4,5,6,7,8.9	(Asad *et al.*, 2012; Saraee *et al*., 2004; Afzal and Latif, 2016)
Cinematographer Rank	Average	Positive Numbers	
Competition		High, Medium, Low	(Simonoff and Sparrow, 2000)
Composer Rank	Average	Positive Numbers	
Costume Designer Rank	Average	Positive Numbers	
Director Rank	Average	Positive Numbers	(Asad *et al*., 2012; Saraee *et al*., 2004)
Distributor Rank	Sum	Positive Numbers	
Editor Rank	Average	Positive Numbers	
Genre		Action, adventure, thriller, biography, crime, drama, horror, comedy, fantasy, animation, mystery, music, war, documentary, romance, sci-fi, western, family, sport, short	(Asad *et al*., 2012; Sharda and Delen, 2006)
MPAA		R, PG, PG-13, G, NR	(Afzal and Latif, 2016; Simonoff and Sparrow, 2000; Sharda and Delen, 2006)
Production Company Rank	Average	Positive Numbers	
Production Designer Rank	Average	Positive Numbers	
Production Rank	Average	Positive Numbers	
Writer Rank	Average		

Table 2. Summary of input attributes of the IMDb Dataset

Table 3. Additional input attributes of the IMDb+ Dataset

Attribute Name	Math Operation	Values	Data Source	References
Golden Globe Nominee		Positive Integers	IMDb Website	(Afzal and Latif, 2016; Simonoff and Sparrow, 2000)
Golden Globe Win		Positive Integers	IMDb Website	(Afzal and Latif, 2016; Simonoff and Sparrow, 2000)
Metascore		Positive Integers	Metacritic Website	(Afzal and Latif, 2016; Simonoff and Sparrow, 2000)
Opening Gross	Discretization	1,2,3,4,5,6,7,8,9	BoxOfficeMojo Website	(Afzal and Latif, 2016; Simonoff and Sparrow, 2000)
Opening Theaters		Positive Integers	BoxOfficeMojo Website	(Afzal and Latif, 2016; Simonoff and Sparrow, 2000; Sharda and Delen, 2006)
Oscar Nominee		Positive Integers	IMDb Website	

Attribute Name	Math Operation	Values	Data Source	References
Oscar Win		Positive Integers	IMDb Website	(Afzal and Latif, 2016; Simonoff and Sparrow, 2000)

Table 3. Additional input attributes of the IMDb+ Dataset

Class	Rating
Excellent	7.5–10
Average	5–7.4
Poor	2.5–4.9
Terrible	1–2.4

Table 4. The output attribute of the IMDb and IMDb+ Datasets

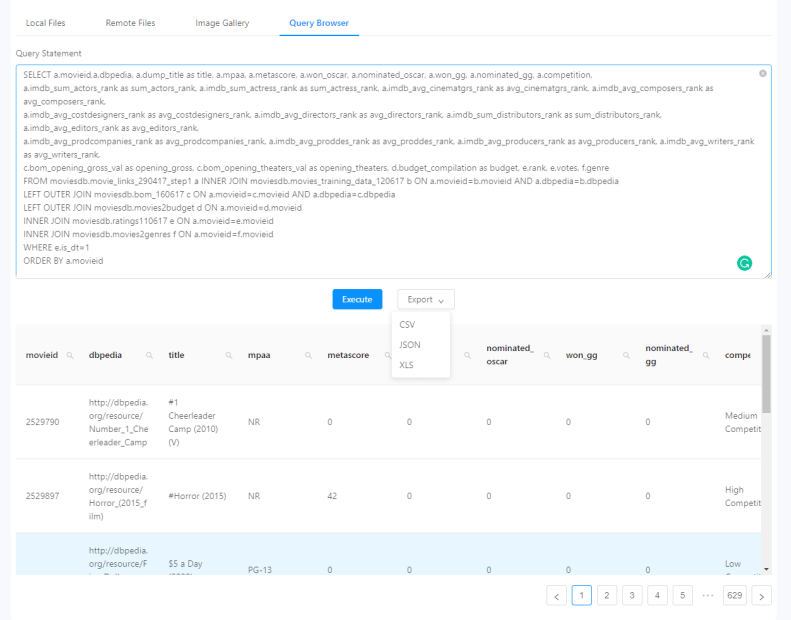

Figure 14. The SQL query to generate the IMDb Dataset

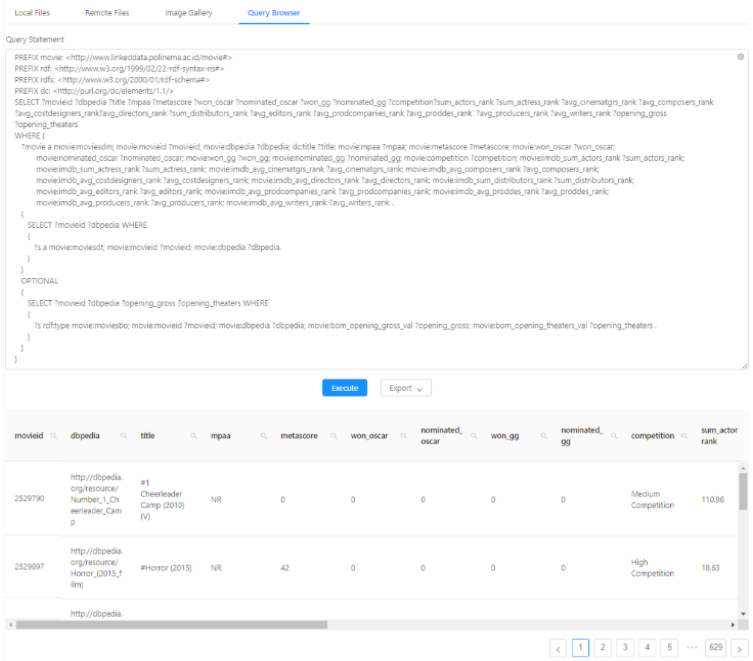

Figure 15. The SPARQL query to compose the IMDb+ Dataset

E. Data mining

In utilizing the IMDb and IMDb+ datasets to predict the future popularity of films, we employ five classifiers: Artificial Neural Network (ANN), Decision Tree (DT), k-NN, Rule Induction (RI), and Support Vector Machine (SVM). Advantages of neural networks include the following: to model complex dynamic systems on a wide variety of applications with low mathematical calculation requirements [37]. Decision Tree is robust in its imperviousness to noise, flexible in dealing with redundant attributes, and generates models at a low computational cost [38]. k-NN is a lazy learner that uses the training dataset as a lookup table with the aid of which they can match input variables to achieve the desired outcome [39]. k-NN is a simple supervised learning learning algorithm that is easy to implement [40]. RI is a classifier that demonstrates the ability to classify unknown data and to produce if-then rules in a simple form easily understood by non-expert users.

Predictive tasks turn future uncertainties into usable probabilities [41]. Therefore, selecting appropriate methods for testing the quality of a predictive model is imperative. The measurement of standard metrics (Accuracy, Precision, Recall) for measuring classification performances is often based on the confusion matrix. Besides these three metrics, we consider one additional measure, namely the F-Measure. As mentioned by Ference et al. [42], F-Measure can be beneficial to select the most well-suited hyperparameters.

Secondly, it balances the performance of the classes when a substantial class imbalance is found in a dataset. We selected a popular method to test and evaluate classification models, which are k-fold cross-validation. The setting of the k-value to get a broad research context or not, adapted to research needs [43]. In this work, we set k = 10. In the 10-fold cross-validation, the data set is divided into ten subsets and repeated in ten iterations. Over each iteration, one subset is leveraged as the testing data, while the nine subsets are utilized as training data. Across all ten trials, performance statistics are calculated.

The classification results yielded from each classifier using the IMDb and IMDb+ datasets are shown in Table 5 and Table 6, respectively. Applied to the two datasets, the two highest F-Measure value were achieved by k-NN and RI. In the IMDb dataset, the comparison is 83.47% and 82.02%, while the IMDb + dataset contributes to 86.33% and 82.75%. The results also suggest that the addition of input attributes from external data sources in the IMDb+ dataset can improve classification performances using the IMDb dataset. This is in-line with the advocate by Ristoski and Paulheim [3], in which the addition of background knowledge from different data sources exhibits the potential to improve data mining results. The use of background knowledge to enrich datasets is similar with the adding of new axioms, concepts, properties, and rules into an existing ontology in the process of ontology enrichment. Zaouga and Rabai [44] demonstrated the ontology enrichment was able to improve the performance of a decision support system to mitigate risks in project management.

Compared with previous works in classifying movie popularities, Asad et al. [34] achieved the best accuracy using the PART classifier for 77.72%, and the highest accuracy was gained by Afzal and Latif [36], with 84.34% using the classifier of Simple Logistic. Our results using the IMDb and IMDb+ datasets as indicated in Table 5 and Table 6 are much better, and the lowest accuracy was produced by SVM for 84.23% and 85.26%, respectively, while the best accuracy achieved was 89.62% using RI on the IMDb and 91.19% using k-NN on the IMDb+. As k-NN is a non-parametric classifier that supports non-linear solutions and tunes a few hyperparameters. Meanwhile, the IMDb+ having more attributes than the IMDb which may affect the IMDb+ to be more non-linear. This is one of the reasons why k-NN is better than other classifiers when working with the IMDb+. The best classifier is very likely to change when the hyperparameters tuning is applied.

Classifiers	Accuracy	Precision	Recall	F-Measure
ANN	86.59	81.58	74.86	78.20
Decision Tree	84.38	81.68	72.83	77.00
k-NN	89.19	85.36	81.66	83.47
Rule Induction	89.62	88.88	76.15	82.02
SVM	84.23	81.35	65.41	72.52

Table 5. Percentage of classification results of the IMDb dataset

Classifiers	Accuracy	Precision	Recall	F-Measure
ANN	88.32	83.57	78.26	78.20
Decision Tree	85.02	83.05	71.34	76.75
k-NN	91.19	88.44	84.32	86.33
Rule Induction	89.79	89.83	76.71	82.75
SVM	85.26	81.50	68.61	74.62

Table 6. Percentage of classification results of the IMDb+ dataset

F. The use case of SPARQL query federation

One of the advanced features of linked data is to interlink documents, data, people and organizations in various and often unexpected ways [45]. The moviesdb.rdf file contains information for a film entitled The Revenant (2015), with id 3591049 and an URL to DBpedia. For instance, providing information about what notable works were done by The Revenant's director is required. Leveraging the linked data principles, we address this question by interlinking the local moviesdb.rdf dataset to external DBpedia datasets to gain information enrichment. BDS supports SPARQL 1.1, which one of the features, namely SERVICE, is to be used to implement a federated SPARQL query. Federated SPARQL processing systems enable the execution of queries distributed over multiple SPARQL endpoints [45]. The fusion of information between the local dataset and the external dataset using the SPARQL federated query is shown in Figure 16. We queried to moviesdb.rdf to retrieve information such as the following: movie id, movie title, director, and DBpedia URL of the corresponding movie. The movielinks variable was utilized to join between these results and DBpedia SPARQL Endpoint. The use of movielinks as a subject and a director ontology of DBpedia as a predicate were applied to fetch the URI of The Revenant director. A triple consisting of the director URI as a subject and a notableWorks property of DBpedia as predicate were used to retrieve the notableWorks URI. Finally, the composition of the notableWorks URI as subject and a rdfs:label as predicate were leveraged to get a

list of notableWorks. The list consists three notable works of the director, as indicated in Figure 16.

V. Conclusion

This work accomplished one of the three stages of implementing BDS based on software enhancement (There are additional features customized to the research data). In the design stage a decouple architecture of BDS consisting of four components was produced, i.e., front-end, back-end, API, and user-application. This work also presented two use cases of BDS.

The front-end component is responsible to interact between either users or user-applications and BDS. The UIs of login and entry dataset pages are beneficial in three folds. Firstly, realistic, they are the images of the final product that even the development stage did not start yet. Secondly, they are easy to be changed by fixing in the design step is easier rather than later in the development one when coding processes are starting. Thirdly, they are satisfying because they are naturally acceptable and closely favor with the final BDS.

The back-end component was planned for exhibiting five core capabilities. The first one is to manage types and formats of data. The second one is data storage management. This way prepares storage mechanisms for large size files in different formats in a distributed manner to minimize I/O or resource bottlenecks. The third is data query management, which supports queries over different file formats using the most popular query language, SQL. It also assists users in working with triple data claimed as machine-understandable data using the SPARQL query. The fourth is data sharing management (public, displayed, and private) to limit users in accessing datasets. BDS places mechanisms of authentication and authorization using OAuth2 before user-applications can access data via APIs. The fifth is metadata management. BDS is intended to be able to discover, understand, and integrate by external web resources. To address those challenges, BDS published a metadata catalog based on a standard model and vocabulary and in a distributed way (leverages DCAT to manage the metadata system).

BDS also is equipped with components of API and user-application. The API component adapts RESTful API HATEOAS. It is more advantageous, compared with the standard API in two ways. The former is embedding hyperlinks and controls, while the latter is to provide a browsable navigation system. The user-application component is purposed to make BDS can be broadly accepted. More users and applications can connect, collaborate, and integrate into leveraging research data to improve existing researches.

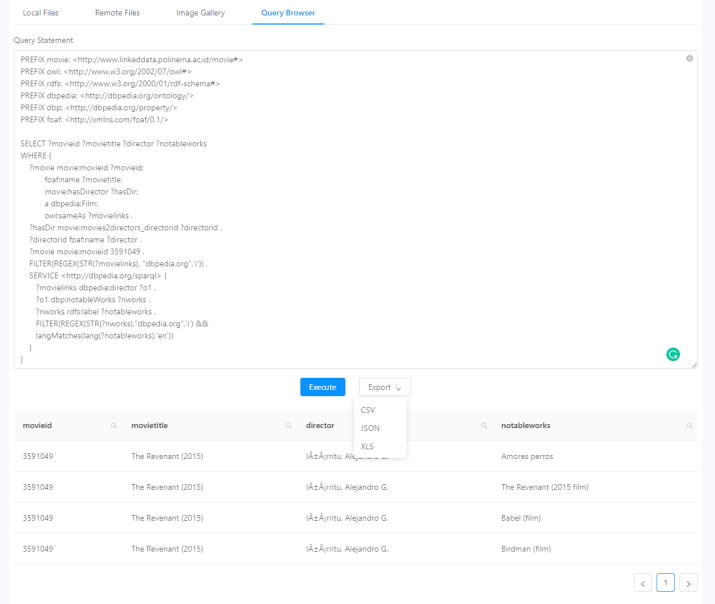

Figure 16. The federated SPARQL query to fuse information between two datasets

Two case studies are provided as a manifestation of the use of BDS, the first using the IMDb and IMDb+ data sets to predict the popularity of films in the future, using five classifiers (ANN, DT, k-NN, RI, and SVM). This first use to show that in BDS we can take advantage of available datasets available on BDS and then process them using several classifiers using the available APIs on BDS. Second is the SPARQL query federation use case. The second use to show BDS features in the form of advanced features of linked data that connect documents, data, people, and organizations. The next paper would be a discourse of developing BDS.

Acknowledgments

The authors would like to thank the State Polytechnic of Malang for the grant fund support through 2020 Applied Research Grant Contract.

References

[1] A. Gandomi, M. Haider. “Beyond the Hype: Big Data Concepts, Methods, and Analytics”, *International Journal of Information Management*, 35 (2), pp. 137–144, 2015.

[2] M.I. Bellgard. "ERDMAS: An Exemplar-Driven Institutional Research Data Management and Analysis Strategy", *International Journal of Information Management*, 50, pp. 337–340, 2020.

[3] P. Ristoski, H. Paulheim. "Semantic Web in Data Mining and Knowledge Discovery: A Comprehensive Survey", *Journal of Web Semantics*, 36, pp. 1–22. 2016.

[4] T. A. T. Izhar, T. Torabi, and M. Ishaq Bhatti, "Using Ontology to Incorporate Social Media Data and Organizational Data for Efficient Decision-Making," *International Journal of Computer Information Systems and Industrial Management Applications*, 8, pp. 372–385, 2016.

[5] X. Yang, R. McEwen, L.R. Ong, M. Zihayat. "A Big Data Analytics Framework for Detecting User-Level Depression from Social Networks", *International Journal of Information Management*, 54, pp. 102–141, 2020.

[6] A.J. Traina, S. Brinis, G.V. Pedrosa, L.P. Avalhais, C. Traina Jr, "Querying on Large and Complex Databases by Content: Challenges on Variety and Veracity Regarding Real Applications", *Information Systems*, 86, pp. 10–27, 2019.

[7] L. Abberley, N. Gould, K. Crockett, J. Cheng. "Modelling Road Congestion Using Ontologies for Big Data Analytics in Smart Cities". In *2017 International Smart Cities Conference (ISC2), IEEE*, pp. 1–6, 2017. 6

[8] J.F. Brothers II, M. Ung, R. Escalante-Chong, J. Ross, J. Zhang, Y. Cha, A. Lysaght, J. Funt, R. Kusko. "Integrity, Standards, and QC-Related Issues with Big Data in Pre-clinical Drug Discovery", *Biochemical Pharmacology*, 152, pp. 84–93, 2018.

[9] B. Gupta, A. Kumar, R.K. Dwivedi. "Big Data and Its Applications–A Review". In *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), IEEE*, pp. 146–149, 2018.

[10] W.K. Michener. "Ecological Data Sharing", *Ecological Informatics*, 29 (1), pp. 33–44, 2015.

[11] H. Pampel, P. Vierkant, F. Scholze, R. Bertelmann, M. Kindling, J. Klump, H.J. Goebelbecker, J. Gundlach, P. Schirmbacher, U. Dierolf. "Making Research Data Repositories Visible: The re3data.org Registry", *PloS One*, 8 (11), pp. e78080, 2013.

[12] C.L. Borgman. "The Conundrum of Sharing Research Data", *Journal of the American Society for Information Science and Technology*, 63 (6), pp. 1059–1078, 2012.

[13] B. O'Neill, C. Ryan, S. Roy, T. Simes. "Supporting Nursing Faculty with a Digital Repository of Simulation Resources", *Teaching and Learning in Nursing*, 15 (3), pp. 175–180, 2020.

[14] K. Chard, E. Dart, I. Foster, D. Shifflett, S. Tuecke, J. Williams. "The Modern Research Data Portal: A Design Pattern for Networked, Data-intensive Science", *PeerJ Computer Science*, 4 (6), pp. 1–30, 2018.

[15] Bringula, R. P. "Influence of faculty- and web portal design-related factors on web portal usability: A hierarchical regression analysis". *Computers & Education*, 68, 187–198, 2013.

[16] K.A. Lawrence, M. Zentner, N. Wilkins-Diehr, J.A. Wernert, M. Pierce, S. Marru, S. Michael. "Science Gateways Today and Tomorrow: Positive Perspectives of Nearly 5000 Members of the Research Community", *Concurrency and Computation: Practice and Experience*, 27 (16), pp. 4252–4268, 2015.

[17] Zhong, H., Wachs, J. P., & Nof, S. Y. "Telerobot-enabled HUB-CI model for collaborative lifecycle management of design and prototyping". *Computers in Industry*, 65(4), pp. 550–562, 2014.

[18] T. Tiropanis, W. Hall, J. Hendler, C. de Larrinaga. "The Web Observatory: A Middle Layer for Broad Data", *Big Data*, 2 (3), pp. 129–130, 2014.

[19] W. Hall, T. Tiropanis. "The Web Science Observatory - The Challenges of Analytics Over Distributed Linked Data Infrastructures", *Computer Networks*, 56 (18), pp. 3859–3865, 2012.

[20] R. Tinati, X. Wang, T. Tiropanis, W. Hall. "Building a Real-Time Web Observatory", *IEEE Internet Computing*, 19 (6), pp. 36–45, 2015.

[21] N.R. Aljohani, R.A. Abbasi, F.M. Bawakid, F. Saleem, Z. Ullah, A. Daud, M.A. Aslam, J.S. Alowibdi, S.U. Hassan. "Web Observatory Insights: Past, Current, and Future", *International Journal on Semantic Web and Information Systems (IJSWIS)*, 15 (4), pp. 52–68, 2019.

[22] Web Observatories - Web Science Trust. (2019), "The Web Science Trust (WST)", Web Sciences Trust: Untangling the Web of Humans and Technology, 2019. [Online]. Available at: https://www.webscience.org/web-observatory/ (accessed 10 December 2019).

[23] A.X.L. Han Hu, Y. Wen, T.S. Chua. "Toward Scalable Systems for Big Data Analytics: A Technology Tutorial", *Explore Research*, 2, pp. 2–28, 2011.

[24] H.J. Kim, E.J. Ko, Y.H. Jeon, K.H. Lee. "Techniques and Guidelines for Effective Migration from RDBMS to NoSQL", *The Journal of Supercomputing*, 76, pp. 7936–7950, 2020.

[25] K. Gupta, A. Sachdev, A. Sureka. "Empirical Analysis on Comparing the Performance of Alpha Miner Algorithm in SQL Query Language and NoSQL Column-oriented Databases Using Apache Phoenix", *arXiv* preprint arXiv:1703.05481v1, pp. 1–21, 2017. 24

[26] D.P. Augustine. "Leveraging Big Data Analytics and Hadoop in Developing India's Healthcare Services", *International Journal of Computer Applications*, 89 (16), pp. 44–50, 2014.

[27] P. Ushapreethi, B. Jeyakumar, P. BalaKrishnan. "Action Recongnition in Video Survillance Using Hipi and Map Reducing Model", *International Journal of Mechanical Engineering and Technology*, 8 (11), pp. 368–375, 2017.

[28] C. Sweeney, L. Liu, S. Arietta, J. Lawrence. "HIPI: A Hadoop Image Processing Interface for Image-Based Mapreduce Tasks", *Chris. University of Virginia*, 2 (1), pp. 1–5, 2011.

[29] M. Garriga, C. Mateos, A. Flores, A. Cechich, A. Zunino. "RESTful Service Composition at a Glance: A Survey", *Journal of Network and Computer Applications*, 60, pp. 32-53, 2016.

[30] A. Neumann, N. Laranjeiro, J. Bernardino. "An Analysis of Public REST Web Service APIs", *IEEE Transactions on Services Computing*, 14 (4), pp. 957–970, 2018.

[31] N. Freed, A. Melnikov, M. Kucherawy. "Media Types", 2020. [Online]. Available at: http://www.iana.org/assignments/media-types/media-types.xhtml (Accessed February 21, 2020).

[32] F. Maali, J. Erickson. "Data catalog vocabulary (DCAT)", *W3C Recommendation*, January 16, 2014. [Online].

Available at: http://www.w3.org/TR/vocab-dcat/ (Accessed December 23, 2019).
[33] G. Hagedorn, D. Mietchen, R.A. Morris, D. Agosti, L. Penev, W.G. Berendsohn, D. Hobern. "Creative Commons Licenses and the Non-commercial Condition: Implications for the re-use of biodiversity information", *ZooKeys*, 150, pp. 127–149, 2011.
[34] K.I. Asad, T. Ahmed, M.S. Rahman. "Movie popularity classification based on inherent movie attributes using C4. 5, PART and correlation coefficient". In *2012 International Conference on Informatics, Electronics & Vision (ICIEV), IEEE*, pp. 747–752, 2012.
[35] M. Saraee, S. White, J. Eccleston. "A Data Mining Approach to Analysis and Prediction of Movie Ratings", *Management Information Systems*, 10, pp. 343–352, 2004.
[36] H. Afzal, M.H. Latif. "Prediction of Movies Popularity Using Machine Learning Techniques", *International Journal of Computer Science and. Network Security*, 16 (8), pp. 127–131, 2016.
[37] Vergini, Eleni S. Groumpos, Peter P. "Advanced State Fuzzy Cognitive Maps applied on nearly Zero Energy", *IFAC PapersOnLine*, 54-13 pp 533–538, 2021.
[38] R.C. Barros, M.P. Basgalupp, A.C. de Carvalho, A.A. Freitas. "A Survey of Evolutionary Algorithms for Decision-Tree Induction", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42 (3), pp. 291–312, 2011.
[39] P. Viswanath, T.H. Sarma. "An Improvement to K-nearest Neighbor Classifier". In *2011 IEEE Recent Advances in Intelligent Computational Systems, IEEE*, pp. 227-231, 2011.
[40] Tharwat, A., Mahdi, H., Elhoseny, M., & Hassanien, A. E. "Recognizing human activity in mobile crowdsensing environment using optimized k -NN algorithm". *Expert Systems with Applications*, 107, pp. 32–44, 2018.
[41] J. Taylor. *Decision management systems: a practical guide to using business rules and predictive analytics*, Pearson Education, Boston, 2011.
[42] Ferenc, R., Bán, D., Grósz, T., & Gyimóthy, T. "Deep learning in static, metric-based bug prediction". *Array*, 6, 100021, 2020.
[43] Rafalo, Mariusz. "Cross validation methods: Analysis based on diagnostics of thyroid cancer metastasis". ICT Express, 2021.
[44] W. Zaouga and L. B. A. Rabai, "A Decision Support System for Project Risk Management based on Ontology Learning," International Journal of Computer Information Systems and Industrial Management Applications, 13, pp. 113–123, 2021.
[45] P. Peng, L. Zou, M.T. Özsu, L. Chen, D. Zhao. "Processing SPARQL Queries Over Distributed RDF Graphs", *The International Journal on Very Large Data Bases*, 25 (2), pp. 243–268, 2016.

Gunawan Budiprasetyo received his Bachelor of Industrial Engineering and Master of Management Technology – Management of Information Technology degrees from UPN Surabaya and ITS Surabaya in 2000 and 2007, respectively. In 2019, he finished Ph.D program in Computer Science at the University of Southampton. He is active as a lecturer and researcher in State Polytechnic of Malang. His research interest is big data, distributed information system, decision support system, machine learning, and semantic web.

Yoppy Yunhasnawa received his Bachelor of Applied Science degree from Electronic Engineering Polytechnic Institute of Surabaya in 2013 and Master of Science degree from Chang Gung University in 2016, now he is an active lecturer and researcher at State Polytechnic of Malang. His researches focus on software engineering, artificial intelligence, information systems, and database programming.

Mustika Mentari received her Bachelor of Computer from Brawijaya University in 2011 and Master of Computer from Institute Technology of Sepuluh Nopember in 2014. Now she is an active lecturer and researcher which focuses on computer vision, image processing, and machine learning.

Dito Cahya Pratama is an active student of bachelor program in State Polytechnic of Malang. He experienced frontend web developer with a demonstrated history of working in the computer software industry.