LinkWiper – A System For Data Quality in Linked Open Data

by

Srivalli Gade

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
(Computer and Information Science)
in the University of Michigan-Dearborn
2016

Master's Thesis Committee:

Associate Professor Brahim Medjahed, Chair
Associate Professor Marouane Kessentini
Associate Professor Qiang Zhu

Acknowledgements

It is a pleasure to thank profusely all people to make this thesis possible.

At the outset, I would like to express my sincere gratitude to my preceptor and advisor Dr. Brahim Medjahed for his continuous support in completing my research paper with patience.

His enthusiasm and immense knowledge and motivation helped me in guiding all the time during my research and writing of this thesis. I thank wholeheartedly for reposing confidence in me and giving me the opportunity to work with him and learn from him. Without his precious guidance I would not have completed this thesis. I express my deep sense of appreciation for sharing the pearls of wisdom with me during the course of this research work.

Besides my advisor, I would like to thank the rest of members of my thesis committee: Prof. Dr.Marouane Kessentini, Prof. Dr. Qiang Zhu for not only their insightful comments and encouragement, but also exposing me to very hard question which incented me to widen scope of my research from various perspectives time to time.

Last but not the least, I would like to thank my family, my parents, my brother and sister for supporting me and helping me morally throughout writing this thesis in particular and also my life in general

Table of Contents

List of Figures

List of Tables

Abstract

Linked Open Data (LOD) provides access to large amounts of data on Web. These data sets range from high quality curated data sets to low quality sets. LOD sources often need strategies to clean up data and provide methodology for quality assessment in linked data. They allow interlinking and integrating any kind of data on the web. Links between various data sources enable software applications to operate over the aggregated data space as if it is a unique local database. However, such links may be broken, leading to data quality problems. In this thesis we present LinkWiper, an automated system for cleaning data in LOD. While this thesis focuses on problems related to dereferenced links, LinkWiper can be used to tackle any other data quality problem such as duplication and consistency. The proposed system includes two major phases. The first phase uses information retrieval-like search techniques to recommend sets of alternative links. The second phase adopts crowdsourcing mechanisms to involve workers (or users) in improving the quality of the LOD sources. We provide an implementation of LinkWiper over DBPedia, a community effort to extract structured information from Wikipedia and make this information using LOD principles. We also conduct extensive experiments to illustrate the efficiency and high precision of the proposed approach

Chapter 1 Introduction

1.1 Motivation

Linked open Data (LOD) [1] comprises an unprecedented volume of structured data present in the Web. However, these data sets are of varying quality ranging from extensive curated data sets to low quality data. So there are numerous amounts of data quality problems in Web. First of all structured data is referenced through links as shown in Figure-1.1. These links can be dereferenced, or there is possibility that these links can be broken for any cause. There are numerous methodologies or frameworks to find out the reasons for the breakdown of links.

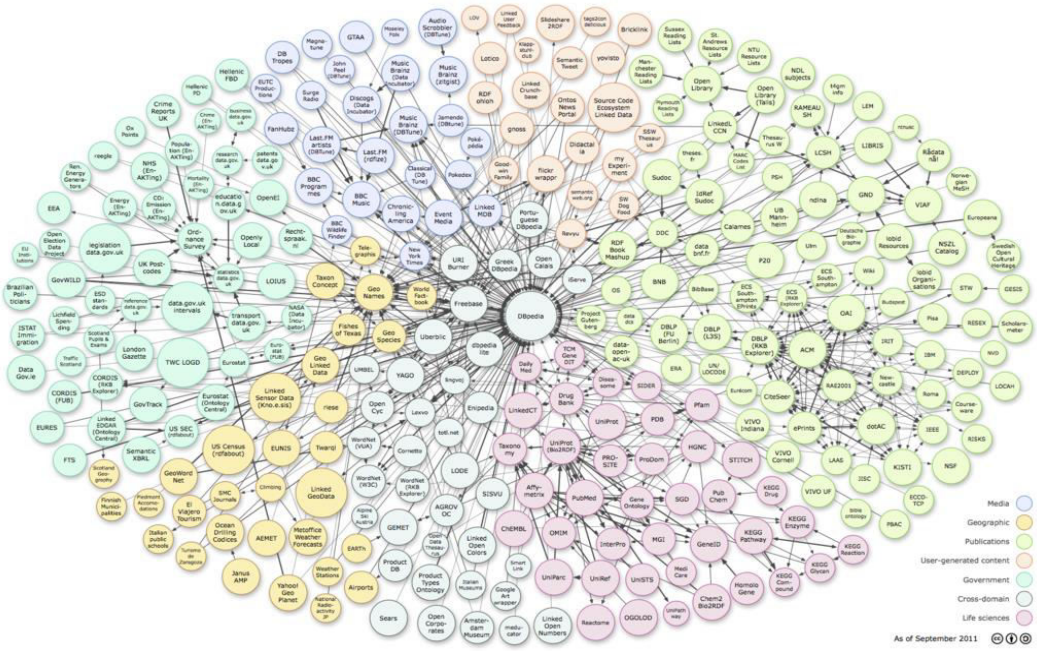


Figure 1.1 – Example of Linked Open Data Cloud diagram

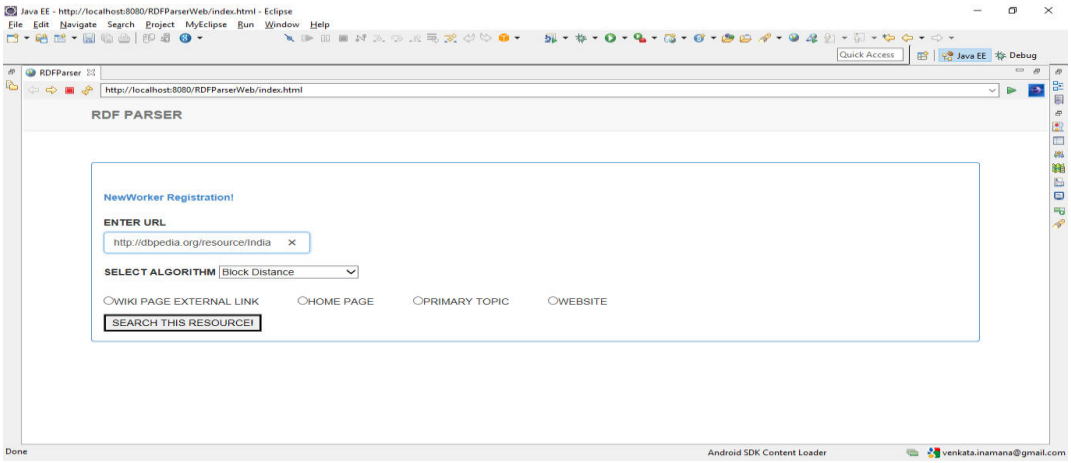


Figure 1.2 - RDF PARSER

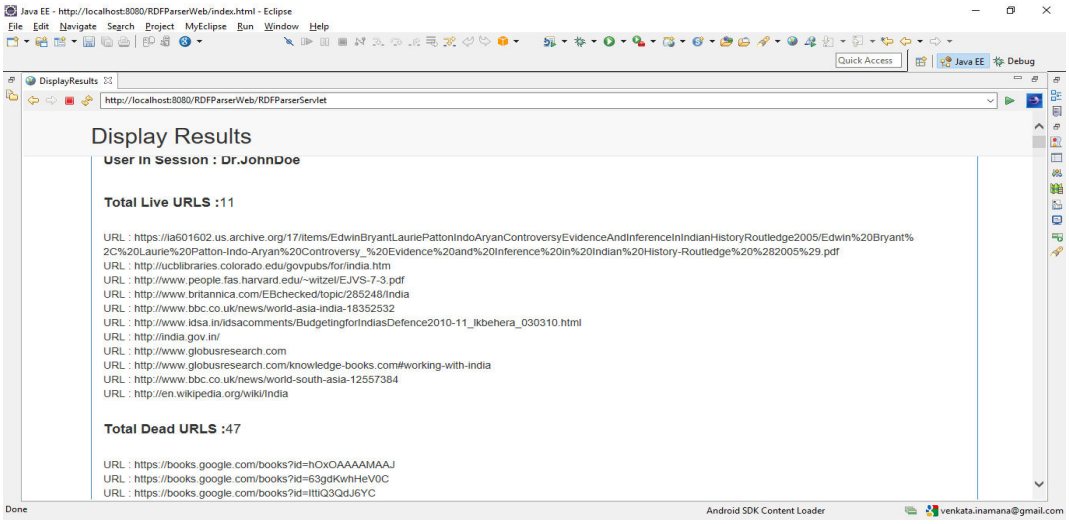


Figure 1.3 - RDF PARSER Results

There are related works in literature. Automatic Link Exploration System (ALEX) is one system that aims at improving the quality of links between RDF data sets by using feedback provided by users on the answers to the linked data queries. ROOMBA is a system that demonstrates the general state of various datasets and groups, including the LOD cloud group, need more attention as most of the datasets suffer from bad quality metadata and lack some informative metrics that are required to facilitate dataset search. In a lightening talk session at the First workshop on the Linked Data Quality, many challenges are discussed. The proposal to tackle different problems is also discussed. There is lot of research in literature regarding data quality assessment and meth-

odology for the Linked Data. We have gone a step ahead and provided solution of some data quality issues and define an automated tool/framework which can be used to any other data quality problem and can implement the strategy of solution.

In this thesis, we develop an automated tool/framework, called LinkWiper, for the specific solution of dereferencing the bad URLS or broken URLs. The automated tool/framework developed is so effective in bringing the most accurate domain specific URLs to that of bad URL. As part of implementing the solution we have made use of domain specific and semantic text matching algorithms. To get effective and more accurate answers for the tool generated questions we make use of crowd sourcing concepts and developed algorithms. This has helped in getting better results and filter out low quality links that got generated which are irrelevant to the search.

1.2 Quality Issues in Linked Open Data

After mentioning about the goal of linked data, we can look into data quality issues and decide what issues we consider to render solution using our automated tool/framework. The Semantic Web has gained popularity after structured data getting published as Resource Description Framework (RDF) [2]. The era of advanced development of data has thrown a lot of challenges to Web experts. The Linked Data paradigm has gained enormous momentum to create transition from document oriented Web to Web of data called ultimately Semantic Web [3]. The term Linked Data refers to as set of best practices for publishing structured data on the Web. Many data providers have adopted these best principles over the years leading to the creation of global data space that contains billions of assertions – the Web of Linked Data [1].

The main goal of Linked Data is to publish structured data by interlinking relative documents in Web of data. The huge amount of data has created enormous challenges with regards to data quality. Since data is extracted via crowd sourcing of semi-structured data sources there are many challenges with quality of data sets published. One of the better strategies is to assess data quality issues and avoid them. There are five classifications of principles for data quality assessment in semantic Web [5]. These principles are:

The principle **Quality of Data Source** is availability of data and credibility of the data Source. Whether an access method, protocols perform properly and also data sources credibility is verified. Incoming and outgoing links of URI s are de-referenceable. We faced this challenge to overcome solution in our system. URI dereferencing problem are taken care of this challenge and solution is proposed. The principle **Quality of Raw Data** is mainly related to the absence of duplicates, entry mistakes and noise in the data. Even part of this principle is covered in the first data quality issue in URI referencing. In the absence of availability of duplicate URI s and mistakes in referencing are taken care of. The principle **Quality of the semantic conversion** is related to the transformation of raw data into rich data by making use of vocabularies. The principle Quality of the linking process is related to the quality of links between two datasets. The principle **Global quality** is cross-cutting of the other principles and covers the source, raw data, semantic conversion, reasoning and links quality.

The principle (Quality of data Source) is taken and the metrics considered are detection of dead links, broken links, and non-dereferenced bad links, forward and backward links. As part of implementing the solution we make use of domain specific and semantic text matching algorithms. To get effective and more accurate answers for the tool generated questions we have made use of crowd sourcing concepts and developed algorithms. This has enabled to get better results and filter out low quality links that got generated which are irrelevant to the search. The high level diagram is shown in Figure- 1.4.

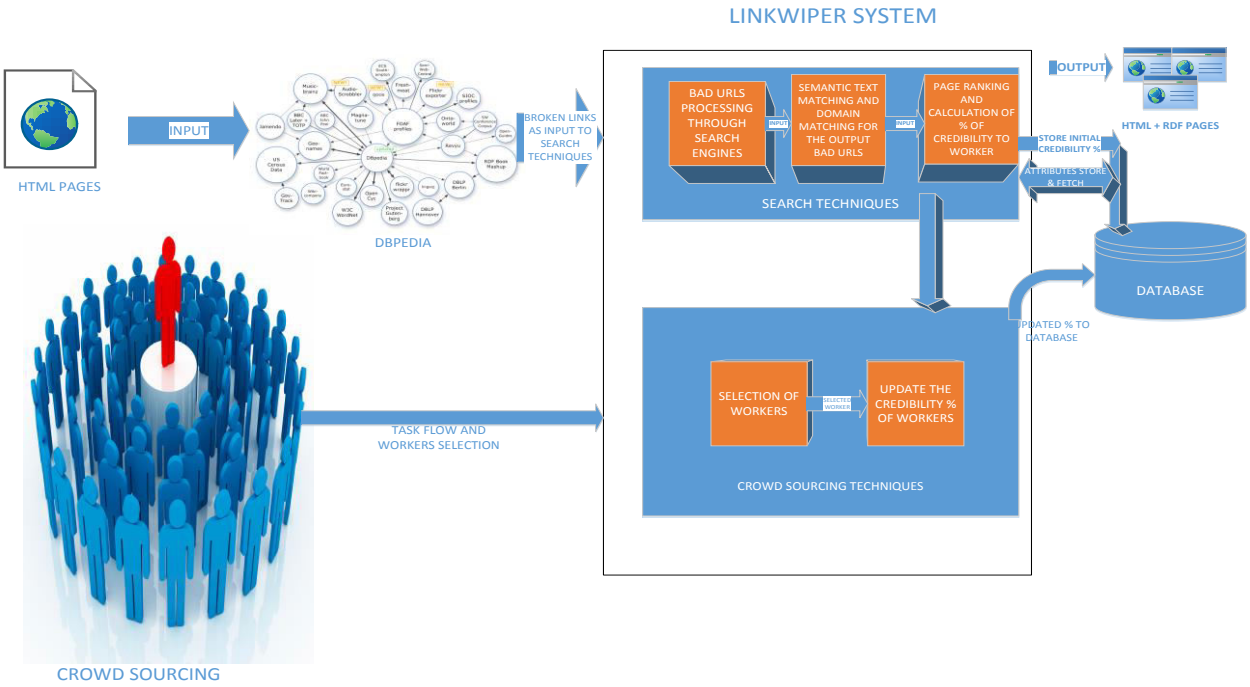


Figure 1.4 - High level diagram

1.3 Problem Statement

The aim is to develop system that will do data quality in Linked Open Data and primary focus is to take one data quality issue and provide solution in an automated tool/framework. Out of many data quality issues we focus mainly on quality of data source. In that principle primary metric is quality of links and if link is dereferenced, forward or backward links may be broken and our goal is to find the URLs recommended for those bad URLs. For this to achieve implemented domain specific and semantic text matching algorithms are to get the URLs matching close to the bad URL. And these algorithms and the approach also need crowd sourcing concepts to inculcate less cost and achieve more productivity and efficiency in System. For that those concepts are added and built in the system. The system is so unique and can be used as solution to other problems as well. We have given example system (see section 1.4 Contributions) that can be built for making use of this system and concepts.

1.4 Contributions

LinkWiper aims at solving the data quality issue "Quality of data Source". The dimensions in this principle are accessibility of Server, checking whether query responds to data, detection of dead links, detection of all forward links, de-referencability issues such as an URI returns an error 4x/5x, response code for broken links, detection of all local in or back links, misreported content types, Whether the content is suitable for consumption, or the content be accessed. In our automated tool/framework user searches the data in the Web of data of DBPedia, there is possibility of having dereferenced or inaccessible or bad URLS of any type to be part of set of result URLs. Our goal is to find the recommended URLs for bad URLs when user searches for resource or page in DBPedia. We have followed different concepts and algorithms and techniques such as Semantic text matching, domain matching techniques and Crowd Sourcing techniques to build this automated tool/framework. And also this project is unique in its own way as lot of research projects similar to this are there but no other project contribute like this project by recommending good and perfect URLs to the bad URLs.

1.3.1 System that can be built using this automated tool/framework

Data Quality issue that is addressed and solved using LinkWiper automated tool is **Availability** in Quality Data Source. This principle states that accessibility of SPARQL endpoint & server and checking whether query responds to the query, detection of dead links, de referenceable issues and misreported content types i.e. detection of whether the content is suitable for consumption, and the content be accessed.

Now let's take **Versatility** and apply the same principles of our automated tool/framework. This principle belongs to metric "Quality of Raw data". This states the absence of duplicates, entry mistakes, and noise in the data. Versatility means that the data provided be represented by using alternative representations. This can be achieved by conversion into various formats or by the data source enables content negotiation.

The aim of the system proposed is to remove duplicates in data store and provide different representation names for the same article. Solution proposed for this issue is for the different article names for the same representation and as step one we have to use search engines (Google Schol-

ar, ISI Web of Science, ACM digital library, IEEE Digital Explore Library etc.) are for getting different types of representations for the same article name. The data is stored in data store. Using semantic text matching algorithms article names are sorted and duplicate names are removed. After analysis and review shortlist of names are obtained. And using crowdsourcing concepts, workers in the pool will select the correct representation of article. And in the Figure shown below shows the steps involved in Figure-1.5 diagram.

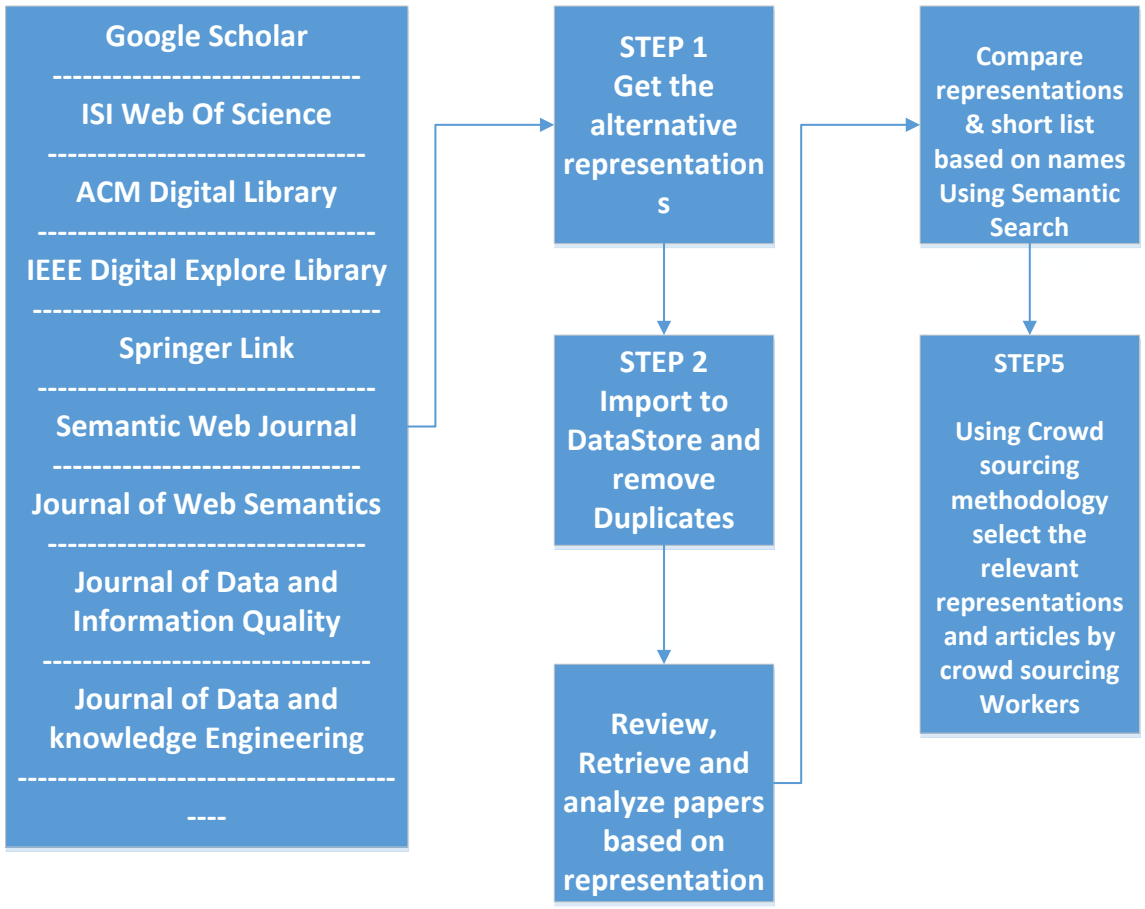


Figure 1.5 - High Level Diagram of System

1.5 Organization of Thesis

The Organization of Thesis is as follows:

Chapter 2 covers Background of thesis. Section 2.1 covers the Introduction of this chapter. Section 2.2 covers the different classifications of data quality issues of semantic Web and linked data. In that there are sub sections Sub-Section 2.2.1 covers the principle quality of data Sources. Sub-Section 2.2.2 covers the principle quality of raw data. Sub-section 2.2.3 covers the principle Quality of the Semantic conversion. Sub-Section 2.2.4 covers the quality of linkage principle. Section 2.3 covers the related work and methodologies.

Chapter 3 covers the Architecture of LinkWiper System. Section 3.1 covers the different architectural components in system such as Search Engines, Semantic text matching, Domain Algorithm, Page ranking Technique, Percentage of relevancy. Section 3.2 covers the Crowd Sourcing techniques. Section 3.3 covers the Security of System. Section 3.4 covers how the system should recover from inconsistent states. Section 3.5 covers the model system that can be built using this automated tool/framework concepts.

Chapter 4 Covers the Implementation and Validation of System. Section 4.1 covers the implementation details of system. This Section will cover subsections of System tiers, configuration, directory structure, error logging, validation of System, Experiments conducted for validating tool/framework algorithms performance.

Chapter 5 covers the conclusion and Future Work part.

Chapter 2 Background

This chapter presents the background of the concepts used in this thesis. Section 2.1 presents Linked data concepts. Section 2.2 presents data quality issues and principles. Section 2.3 presents related work and methodologies. Section 2.4 presents contributions of this thesis and last section 2.6 presents organization of the thesis.

2.1 Linked Data Concepts

World Wide Web (WWW) [6] has radically altered the way information is processed, retrieved and published. Hypertext links helps to achieve in browsing the documents and traversing all the way from one document to another document and analyze the query results of workers search capabilities. This functionality has enabled enormous growth of linking lot of documents in Web of data.

Despite the many benefits that Web of data offers, there is lot of challenges in data quality and principles that are not applied correctly created lot of adhoc in the Web of data causing lot of data quality issues. Traditionally data is stored in csv, raw dumps or HTML tables sacrificing much semantics and structure. However HTML is not enough to store all kinds of structured data in Web. Data era has improved a lot such that documents can be interconnected from one data space to another data space. The evolution of structured data best practices and published data principles has led to structured data interconnecting with one another and is known as Linked Data. These adoptions of principles led into different domains such as people, books, movies, music, genres, government sectors, health media etc. This Web of data enables linking one document from one another and from one space of sector to another. There are Linked Data Search Engines also enables to crawl the Web of data by following links and provides extensive capabil

ities of aggregating data as if we are working in local database now. There are numerous number of benefits from Linked Data and the remainder of the sections enables key features of Linked Data and next section following that will explain what the data quality issues in Linked Data are.

Linked Data [1] is about using the Web to connect related data that wasn't previously linked, or using the Web to lower the barriers to linking data currently linked using other methods. More specifically, Wikipedia defines Linked Data as "a term used to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using URIs and RDF".

According to Tim-Berners Lee [7], to understand the concept and value of linked data it is important to understand the mechanisms of sharing and reusing the data on Web. The principles are use URIs as names for things, use HTTP URIs so that people can look up those names, when someone looks up a URI, provide useful information, using the standards (Resource Description Framework [2] *, SPARQL [9] and include links to other URIs so that they can discover more things.

Semantic Web has evolved as a platform of increasing importance of data interchange and integration through the growing community implementing data sharing using international semantic Web standards called **Linked Data**. Linked Data is evolved to share valuable structured information in flexible and extensible manner across the Web. It is a publishing paradigm for making data and not just human readable documents fully accessible and inter linkable anywhere on the internet.

When publishing data on Web, information is represented using Resource Description Framework (RDF) [2]. In RDF data is represented as triplets. Three parts of triples are the Subject, the Predicate and the Object. Triplets are subject is the URI identifying the described resource, predicate is the relationship exists between the subject and the Object and Object is the URI of another resource that is related to the Subject.

Basically RDF triples are of two types. They are Literal Triples have an RDF literal used to describe the properties of resources. e.g.: name and date of birth of a person and RDF Links- represent typed links between RDF links and resources. The benefits of using RDF Model in Linked Data Context are clients can look up URI in an RDF graph over the Web to retrieve additional

information, information from different sources merge naturally, the data model enables you to set RDF links between data from different data sources, the data model allows you to represent information that is expressed using different schemata in a single model and combined with different Schema languages such as RDF-S and OWL, the data model allows you to use as much as little structure as you need, meaning that you can represent tightly structured data as well as semi structured data.

RDF Features best avoided in Linked Data Context are use of Blank Nodes. (Impossible to set RDF links to a blank node and merging data from different Sources would be more difficult.), use of RDF reification. (Semantics of Reification are unclear and as reified statements are cumbersome to query the SPARQL.) And use of RDF Collections or Containers. (They do not work together with SPARQL. Please check whether application needs Collections or Containers.).

In Choosing URI names the following principles observed are choose good names that other publishers can confidently link to your resources, you will have to put technical infrastructure in place to make them dereference able, use HTTP URIs for everything, do not define URIs in some ones namespace, try to keep URIs stable and persistent or else it will break established links. Vocabulary to represent information are do not define new vocabulary from scratch, provide for both humans and machines, make term URI de referenceable and make use of other people's terms. So RDF links allow users to navigate from one data source to other data sources and to discover additional data. In order to be part of Web data, data sources should set RDF links to related entities in other data sources. As data sources often provide information about large number of entities, it is common practice to use automated or semi-automated approaches to generate RDF links. In various domains there are generally accepted schemata. For instance in the publication domain ISBN and ISSN are common domains and in financial domains there are ISIN identifiers, EAN and EPC codes are widely used to identify products. If the link data source and data targets are already both support of these identification schema RDF links can be made easily. This approach has been used in LOD cloud in various domains.

If no shared naming schema exist, RDF links are often generated using similarity of entities in both data sets. Such similarity is computed based on record linkage and duplicate detection within the database community. Various RDF generation link algorithms are available. For example, SILK is the open source framework for integrating heterogeneous data sources. SILK is based on

the Linked data paradigm, which is built on two simple ideas. First, RDF provides an expressive data model for representing structured information. Second, RDF links are set between entities in different data sources.

2.1.1 Metadata Information

Linked data should be published along with the creation of Meta data in order for the clients to assess the quality of published data, data should be accompanied with creator date, creator time, as well as creator method. Meta information for example can use the standards of Semantic Web Publishing Vocabulary.

In order to support clients in choosing the most efficient way to access Web data for the specific task they have to perform, data publishers can provide additional technical metadata information about their data set and its inter-linkage relationships with other data sets. The Semantic Web Crawling Sitemap extension allows data publishers to state which alternative means of access are provided besides de referenceable URIs. The vocabulary of interlinked data sets defines terms and best practices to categorize and provide statistical Meta information about data sets as well as the linked data sets connecting them.

2.1.2 Linked Data Publishing Tools

A variety of Linked data publishing tools have been developed. Some of the examples are D2RServer is a tool for publishing relational data bases as linked data, Talis Platform is platform provides Linked data complaint hosting for content and RDF data, Pubby is a linked data frontend for SPARQL End points, Paget is a framework for building Linked Data applications, Linked Media Framework is a linked data server with updates and semantic search. Publish-MyData is a linked data publishing platform run by Swirrl. RDF data-hosting, linked data API, SPARQL end point and customizable visualizations.

2.1.3 Linked Data RDF editors

There are two basic type of editors available. They are (a) Hyena: RDF Editor (b) Vapor: Linked data Validator.

The most visible example of application of Linked Data is **Linked Open Data (LOD) [10]** project and the main aim of this project is to make use of principles of Linked Data and publish data on the Web.

2.2 Classification of Data Quality issues in terms of Semantic Web and Linked Data

Despite its importance, data quality has only recently being receiving attention from the Semantic Web community. Most of the related works in the data quality assessment of LOD investigate the quality problems of published data sets. In this section we describe the lot of data quality principles and different metrics related to that principle as shown in Table-2.1.

Table 2.1 - Data Quality Principles

Data Quality Principle	**Attribute**
Quality of Data Sources	Accessibility, Authority & Sustainability, License, Trustworthiness & Verifiability,
Quality of Raw Data	Accuracy, Referential Correspondence, Cleanness, Consistency, Comprehensibility, Completeness, Typing, Provenance, Versatility, Traceability
Quality of the Semantic Conversion	Correctness, granularity, Consistency
Quality of the linking Process	Connectedness, Isomorphism, Directionality

2.2.1 Quality of Data Sources

This principle is related to the availability of the data and the credibility of the data source. Accessibility metric checks whether access methods and protocols perform properly and is all the URIs de-referenceable and the in-going and out-going links operate correctly. Authority & Sustainability metric checks whether the data source provider a known credible source or is he sponsored by well-known associations and providers and are there credible basis for believing the

data source will be maintained and available in the future. License metric checks whether the data source license clearly defined. Trustworthiness & Verifiability metric checks whether the data consumer examine the correctness and accuracy of the data source. The consumer should also be sure that the data he receives is the same data he has vouched for and from the same resource. This can be ensured using digital signatures thus verifying all possible serialization of that data.
Performance metric checks whether the data source capable of coping with increasing requests in low latency response time and high throughput.

2.2.2 Quality of the raw data

This principle is mainly related to the absence of duplicates, entry mistakes, and noise in the data. Accuracy metric checks whether the nodes referring to factually and lexically correct information. Referential correspondence metric checks whether the data described using accurate labels without duplications and the goal is to have one-to-one references between data and real world. Cleanness metric checks whether the data clean and not polluted with irrelevant or outdated data and are there duplicates and the data formatted in a consistent way (i.e., are the dates all formatted yyyy/mm/dd) and tools such as Google Refine or Data Wrangler provide already a good answer to these issues by allowing the cleaning of complex data sets.
Consistency metric checks whether the data contradict it. For example, is the population of Europe the same as the sum of the population of the European countries? To achieve that we need to validate the underlying vocabulary and syntax of the document with other resources.
Comprehensibility metric checks whether the data concepts understandable to humans? Do they convey logical meaning of the described entity and allow easy consumption and utilization of the data? If a concept is described using multiple labels (a set of concepts in a owl: same AS relationship), which one should be consumed? How can we specify which label is canonical? Completeness metric checks whether we have all the data needed to represent all the information related to a real world entity? Moreover, is the data related or linked to this set complete as well, e.g., all European countries, all French cities, all street addresses, all postal codes…? Typing checks whether the data properly typed as a concept from a vocabulary or just as a string literal? Having the data properly typed allows users to go a step further in the business analysis and de-

cision process. Provenance checks whether provenance in the Semantic Web is considered as one of the most important indicators of "quality." And the data sets can be used or rejected depending on the availability of sufficient and/or relevant metadata attached. Versatility checks whether the data provided be presented using alternative representations. This can be achieved by conversion into various formats or if the data source enables content negotiation.
Traceability metric checks whether all the elements of my data traceable (including data itself but also queries, formulae) and Can I know from what data sources they come.

2.2.3 Quality of the Semantic conversion

Semantic conversion is the process of transforming “normal” raw data into “rich” data, i.e. input: [tabular data] output: [RDF using x Vocabulary]. The use of high quality vocabularies and the efficiency of data discovery process are major factors in increasing the quality of data. However, one of the most important aspects that affect the quality of the semantic conversion is the quality and suitability of its data model with the intended usage. The quality of a data model strongly depends on the following aspects:

- Correctness: Is the data structure properly modeled and presented for future conversion?
- Granularity: does the model capture enough information to be useful? Are all the expected data present?
- Consistency: Is the direction of relations consistently done?

2.2.4 Quality of the Linkage

This principle is related to the quality of links between two datasets.
Connectedness: Is the combination of datasets done at the correct resources? Frameworks like Silk ease the linking process but don’t tackle per se the quality of the links that are generated. The quality depends on the link generation configuration. The quality is however improved if your data is linked to some reference dataset.

- Isomorphism: Are the combined datasets modeled in a compatible way? Are the combined models reconciled?

- Directionality: After the linkage, is the knowledge represented in the resulting graph of resources still consistent?

2.3 Related Work

Semantic Web is widely accepted topic and lot of research is getting done in this area especially in the area of data quality issues and especially in finding correct and accurate links in between the data sets. There are numerous amounts of dimensions in data quality issues to pursue. One of the dimensions is regarding the quality of data Source. Checking how query responds to the query, detection of dead links in Web, no dereferenced forward links, detection of available links, when an URI returns an error such as 4x/5x error, response code for broken links, and misreported content types after clinking the links. This dimension is pursued as part of this thesis and generated automated tool/framework to find the good and bad links when URI is searched. All bad links are taken into account and saw what are the response code coming and the reason for failing and recommended the correct URLs for bad URLs. Some of the algorithms related to quality of links and data quality issues are discussed below.

In the article "Repairing broken links in the web of data" [29], they introduced a method for fixing broken links source. Broken links which is based on the source point of links and discover the new address of the broken entity. To this end, they introduced two datasets called superior and inferior. Through these datasets, their method creates a graph structure for each entity that needs to be observed over time. This graph is used to identify and discover the new address of the detached entity. The proposed model is evaluated only with domain of person entities.

ALEX (Automatic Link Exploration System) [15] is a system that uses PARIS (PARIS is a probabilistic holistic automatic linking algorithm that is fully automatic and does not require any prior information. It also produces better quality links than other approaches.) to produce the candidate links which is starting point for this system. ALEX starts with a set of automatically generated links that can be produced using any automatic linking algorithm referred to as candidate links. ALEX removes incorrect links rejected by the user but the main focus of ALEX is to find new links that are similar to the links approved by user. The way ALEX finds similar links as follows: An entity in the RDF data set is represented by the URI. Each entity has a set of at-

tributes. System represents a link between two entities from different data sets by a set of features made up of the attributes of the two entities. A feature is a pair of attributes where the first attribute comes from the first entity and the second comes from the second entity. Each feature has a value which is the similarity score of the value of the two attributes. When a user approves a link by approving a query answer based on this link, ALEX chooses one feature and finds new candidate links for which the value of this feature is within a narrow range around the value of the feature of the approved link. The goal of this system is to refine new links in DBPedia by removing incorrect links to external pages or resources. The distinct feature of ALEX is that it not only removes incorrect links from the set of candidate links but also discovers new links that are not part of this set.

In the review of Quality Assessment for linked open data article [10], systematic approaches for assessing the quality of LOD are presented. They gathered existing approaches and compared and group them under a common classification scheme. In particular unified and formalized commonly used terminologies across papers related to data quality and provide a comprehensive list of the dimensions and metrics. Additionally qualitatively analyzed the approaches and tools using a set of attributes. Mainly aim of the article is to provide researchers and data curators a comprehensive understanding of existing work, thereby encouraging further experimentation and development of new approaches focused towards data quality specifically for LOD. This paper in detail compared commonly used technologies related to data quality, 23 different dimensions and their formalized definitions, metrics for each of the dimensions along with a distinction between them being subjective or objective and comparison of tools used to assess data quality. In order to justify the need of systematic review they first conducted a search for related surveys and literature reviews. The study compares the frameworks and identifies 20 dimensions common between them. Additionally they did a comprehensive review which surveys 13 methodologies for assessing the data quality of data sets available on the Web, in structured or semi structured formats. This survey focuses only on structured data and on approaches that aim at assessing the quality of LOD. They not only identified existing dimensions but also introduced new dimensions relevant for assessing the quality of LOD. Furthermore described quality assessment metrics corresponding to each of dimensions and identified whether they are objectively or subjectively measured.

Some projects have proposed solutions to finding bad URLs simplifying greatly the task of identifying bad URLs. Project Online Checker: This checks your Websites and blogs for dead links and can validate both internal and external URLs. And also reports standard error codes like 404 (Page not Found etc.) for all bad URLs. This can reference stale and dead links. Alpha Software: This also checks for bad URLs and can find dead links and also identifies what kind of error code page is giving after clicking it. Even lot of projects like Wilders Security Forms checks bad URLS and finds dead URLs. Dead Link Checker: This project crawls through the Website and identifies bad URLs to correct them. Xenu Link Sleuth is the project after installing the Xenu Software and opened the tool, checks URL and makes you enter Website's domain. After identifying broken links it creates an excel sheet to track link redirect processes. Xenu gives us a list of broken links. The Xenu report has the advantage of not displaying URLs that are recorded in your analytics because of a typing error. All the links shown in Xenu are actually existing links that live on the site. But these URLs may still contain character errors. For example, links #4 and #5 are caused the by the same problem: "#" is replaced by "%2523." When we identify these instances, we need to determine the cause and fix the same problem across all instances. This step will be complete once you've located the links that need to be redirected and documented them all in your Broken Link Redirect Report.

Chapter 3 Architecture of LinkWiper System

This chapter presents the architecture of LinkWiper System. Section 3.2 presents the crowd sourcing concepts used in this automated tool. Section 3.3 presents the security involved in system and section 3.4 reports about inconsistent state recovery.

Linked open data allow interlinking and integrating any kind of data on the web. Links between various data sources play a key role insofar as they allow software applications (e.g., browsers, search engines) to operate over the aggregated data space as if it was a unique local database. In this new data space, where DBPedia, a data set including structured information from Wikipedia, seems to be the central hub, we analyze and highlight outgoing links from this hub in an effort to discover broken links [30].The paper reports on an experiment to examine the causes of broken links and proposes some treatments for solving this problem. This architecture is valid for any LOD Source. We use DBPedia and LOD interchangeably in the rest of the thesis.

In this section we present the architectural details of system tool/framework and how a user interacts with the system and how the system processes a user's request. From the Figure-4 we can see that there are five major components in the system tool/framework.

Let us elaborate the operations in the components one by one and how the system behaves internally. There are many operations that will be performed in the system according to the user's input. We will go into more details on critical techniques later step by step on this thesis. There is user interface which is more user friendly showing user friendly messages to user at ever point. The user can enter the URL to search for the content in DBPedia. When user types the URL to search for that URL in DBPedia, search results contains set of URLs which can contain some bad URLs. These bad URLs are given as input to Component1.

3.1 Search Techniques

Input to this component is set of bad URLs generated after searching the URL entered in DBPedia. Output to this component is set of URLs sorted out after consolidating the URLs coming from the Search Engines. Starting point to this operation is crowd sourcing worker searches URL in DBPedia and gets the list of good and bad URLS. When this set of bad URLS are used as input to this component, output is obtained as set of URLs relative to the set of bad URLS which re

processed through other components.

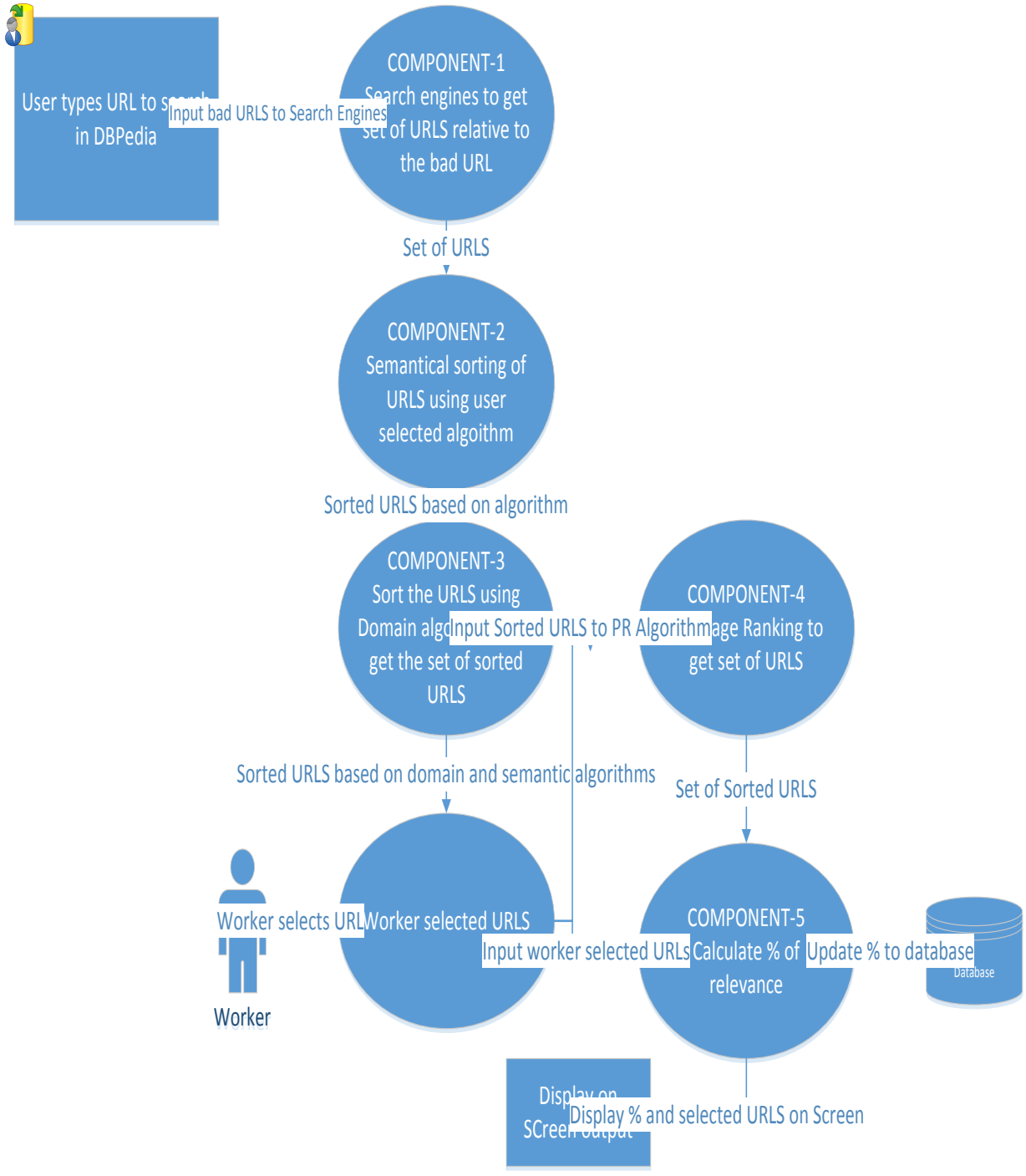


Figure 3.1 - Component Diagram

The algorithm below is used for making use of search engines Google Search Engine API [14].

3.1 Algorithm - Google Search Engine API

```
String google = "http://ajax.googleapis.com/ajax/services/search/Web?v=1.0&q=";
String search = "stackoverflow";
String charset = "UTF-8";

URL url = new URL(google + URLEncoder.encode(search, charset));
Reader reader = new InputStreamReader(url.openStream(), charset);
GoogleResults results = new Gson().fromJson(reader, GoogleResults.class);
```

The figure shows the java snippet of code which fetches the results using the RESTFUL Services Ajax Google API. Further Google Results will be processed and fed as input to the Component2.

Query to test this code is bad URL www. redff.com. Google offers a public search Webservice Which returns JSON. Java Offers java.net.URL and java.net.URLConnection to fire and handle HTTP requests. JSON in java can be converted into java bean output.

String google = http://ajax.googleapis.com/ajax/services/search/Web?v=1.0&q=www.redff.com

Google Results will bring the query format search data which will be processed by further algorithms.

3.1.1 Semantic text matching

Input to this algorithm below is URLs for comparison and Output is the sorted URLs based on the distance calculated by the algorithm. There are different algorithms to calculate distance between URLs, those are given as option to user to select in the User Interface. One of the algorithms is stated below and explained.

3.2 Algorithm - Levenshtein Distance Calculation method

Input : Two Strings for comparison
Output: Sorted Strings based on algorithm.

```
1 public class LevenshteinDistance {
2    private static int minimum (int a, int b, int c) {
3        return Math.min (Math.min (a, b), c);
```

```
4   }
5
6   public static int computeLevenshteinDistance(CharSequence lhs,
7    CharSequence rhs) {
8
9     int[][] distance = new int[lhs.length() + 1][rhs.length() + 1];
10
11      for (int i = 0; i <= lhs.length (); i++)
12         distance[i][0] = i;
13      for (int j = 1; j <= rhs.length (); j++)
14         distance [0][j] = j;
15
16      for (int i = 1; i <= lhs.length(); i++)
17         for (int j = 1; j <= rhs.length(); j++)
18            distance[i][j] = minimum(
19                  distance [i - 1][j] + 1,
20                  distance[i][j - 1] + 1,
21                  distance [i - 1][j - 1] + ((lhs.charAt(i - 1) ==
22                    rhs.charAt(j - 1)) ? 0 : 1));
23
24      return distance[lhs.length()][rhs.length()];
25   }
26   }
```

Two examples of the resulting matrix as shown in Figure-3.2 (hovering over a number reveals the operation performed to get that number):

		k	**i**	**t**	**t**	**e**	**n**
	0	1	2	3	4	5	6
s	1	1	2	3	4	5	6
i	2	2	1	2	3	4	5
t	3	3	2	1	2	3	4

		S	**a**	**t**	**u**	**r**	**d**	**a**	**y**
	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6

t	4	4	3	2	1	2	3
i	5	5	4	3	2	2	3
n	6	6	5	4	3	3	2
g	7	7	6	5	4	4	3

d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

Figure 3.2 - Example for Levenshtein Algorithm

3.1.2 Domain Algorithm

Input to this component is the sorted URLs after applying semantic text matching and output is the sorted list of URLs after applying domain algorithms [16]. In this phase first all the domain names of the input URLs are fetched and if any two URLs have same domain name they are sorted based on the semantic text matching algorithm (see 3.1.2 Component, how it works) and finds least distance URL and final sorted list according to domains is displayed as output.

3.3 Algorithm - Domain Algorithm

Input: List of URLs to compare for domain names of URLs.
Output: Sorted List of URLS by domain algorithm.

```
1   Get the list of URLS to compare for domains
2   Get the domain names of all the URLS and store in domainNames List of
3                       Strings.
4         Call getDomainName (String URL)
5         Begin
6                 Check the URL what is it Starting with?
7                       If it is starting with http or https then save the url as is.
8                       Else if URL is not starting with http or https then prefix the URL
9                     With “http://”
10                      End If
11                Pass the URL to URL class.
12                Get the host out of that URL and store it in host variable.
13                If host starts with www then
```

```
14                    Get the sub string ("www".length () +1) from the host String
15              End If
16              Return host
17              End Method
18        Initialize matched Strings List as new Array List<String> ()
19        If there are badURLS then
20        Begin
21              Loop through Bad URLS
22                    Get the matched Domains out of badURL
23                    Call compareDomainsWithBadURL (domain names, badURL)
24                    Begin
25                          Initialize domain Strings and matched Strings as empty list
26                         Of Strings.
27                          If lists are not empty then begin
28                                Loop through domains
29                                      Compare with badURL domain name
30                                            Do necessary logic and processing
31                                             For strings to match.
32                                            If match Store it in matched Strings
33                                            End If
34                                End loop
35                          End IF
36                                End
37              Now similar domains URLS Compare and get the distance between them
38             And sort it out and display.
39 End
```

A URI (Uniform Resource Identifier) is a representation of a resource available to your application on the network. You can retrieve host of a URL by using Request Object or URI.

Using our code, let's take example of URLs as follows to retrieve domain names.

http://www.rediff.com,

http://www.rediff.com/login.php

http://www.yahoo.com

Line1 gets the URLs to fetch domain names. Line2 calls fetching domain names function to fetch all the domain names of the URLS and store it in domain name list. Line 4 to 17 will work on fetching domain names. In our case, we get the domain names as rediff.com, rediff.com, ya-

hoo.com. Line 18 declares one empty array list to store matching domain names with the domain name of bad URL and rest of the lines loop through all the domain names and matching domain names will be stored in the matchedStrings list. In our case, bad URL is www. redff.com and domain name for this URL is rediff.com and matched strings could be rediff.com not yahoo.com .

3.1.3 Page Ranking Technique

Input to this algorithm stated below is the list of URLs coming as output from the algorithm stated above in 3.2.3 section. In this phase URLs will be sorted out based on Page Ranking Algorithm [17].

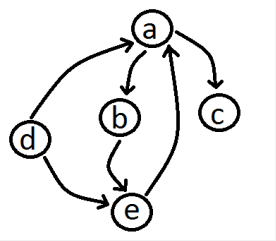
3.4 Algorithm - Page Ranking

```
1  import java.util.*;
2  import java.io.*;
3  public class PageRank {
4
5  public int path[][] = new int[10][10];
6  public double pagerank[] = new double[10];
7
8  public void calc(double n)
9  {
10            double init;
11           double c=0;
12           double temp[] = new double[10];
13           int i,j,u=1,k=1;
14           init = 1/n;
15           System.out.printf(" n value:"+n+"\t init value :"+init+"\n");
16           for(i=1;i<=n;i++)
17                    this.pagerank[i]=init;
18                    System.out.printf("\n Initial PageRank Values , 0th Step \n");
19                   for(i=1;i<=n;i++)
20                            System.out.printf(" Page Rank of "+i+" is :\t"+this.pagerank[i]+"\n");
21
22
23                         while(u<=2)
24                        {
25                        for(i=1;i<=n;i++)
26                        {  temp[i]=this.pagerank[i];
27                           this.pagerank[i]=0;
```

```
28                      }
29
30                  for(j=1;j<=n;j++)
31                  for(i=1;i<=n;i++)
32                   if(this.path[i][j] == 1)
33                  {  k=1;c=0;
34                     while(k<=n)
35                        {
36                          if(this.path[i][k] == 1 )
37                             c=c+1;
38                              k=k+1;
39                          }
40                       this.pagerank[j]=this.pagerank[j]+temp[i]*(1/c);
41                      }
42
43                 System.out.printf("\n After "+u+"th Step \n");
44                 for(i=1;i<=n;i++)
45                 System.out.printf(" Page Rank of "+i+" is :\t"+this.pagerank[i]+"\n");
46
47                  u=u+1;
48          }
49  }
50  public static void main(String args[])
51 {
52              int nodes,i,j,cost;
53              Scanner in = new Scanner(System.in);
54              System.out.println("Enter the Number of WebPages \n");
55              nodes = in.nextInt();
56              PageRank p = new PageRank();
57              System.out.println("Enter the Adjacency Matrix with 1->PATH & 0->NO PATH
58               Between two WebPages: \n");
59                for(i=1;i<=nodes;i++)
60                for(j=1;j<=nodes;j++)

   61              {
   62                    p.path[i][j]=in.nextInt();
   63                  if(j==i)   p.path[i][j]=0;  }  p.calc(nodes);        }     }
```

There are 5 Web pages represented by Nodes a, b, c, d, e. The hyperlink from each Webpage to the other is represented by the arrow head as shown in Figure-3.3 nodes.



steps	A	B	C	D	E
0	0.2	0.2	0.2	0.2	0.2
1	0.3	0.1	0.1	0.1	0.2
2	0.25	0.15	0.15	0.05	0.1

Figure 3.3 - Example for Page ranking Algorithm

As shown in Figure 3.3 data, at 0th Step we have all Webpages PageRank values 0.2 that is 1/5 (1/n). To get PageRank of Webpage A, consider all the incoming links to A. So we have half the Page Rank of D is pointed to A and Full Rank of E is pointed to A. So it will be (1/5)*(1/2) + (1/5)*(1/1) which is (3/10) or 0.3 the Page Rank of A. Similarly the Page Rank of B will be (1/5)*(1/2) which is (1/10) or 0.1 because A's PageRank value is 1/5 or 0.2 from Step 0. Even though we got 0.3 of A's PageRank in Step 1 we are considering 0.3 when we are Calculating Page Rank of B in Step 2. The general rule is, we consider (N-1) step values when we are calculating the Page Rank values for Nth Step. In Similar way we calculate all the Page Rank Values and Sort them to get the most important Webpage to be displayed in the Search Results.

3.1.4 Percentage of relevancy

Set of recommended URLS selected by worker is taken as input to algorithm stated below. Percentage of relevancy of crowd sourcing worker selected URLs is the output of this component. This component makes use of component4 (3.1.4 section) output of sorted list of URLS based on page relevancy. Based on this list if worker selects apt to that selection, then they will be given high percentage of ranking to that crowd source worker. And the same way the percentages will be calculated based on the order of list of page ranked URLs. And if there are two set of bad URLs then average of percentage is taken and updated to the crowd source worker profile to make use for next selections relevancy of particular worker. And these credibility algorithms are discussed in next section 3.2.

3.5 Algorithm - Percentage of relevancy

Input: Set of recommended URLs selected by worker.
Output: calculated Percentage of Relevancy

1: If worker selects highly possible URL then assign 100%, if second possible URL then 75%, or 50% or 25%.
2: u(x)← Average of two selected URLs percentage.
3: Output is u(x).

Page ranking list: www.rediff.com – 1, www.rediff.com/index.php - 2, www.rediff.com/sports – 3, www.rediff.com/news - 4

If the worker selects the same URL as 1 then 100% is assigned or 75% or 50% or 25% are assigned as in line 1 of algorithm. After calculating all the URLs percentage average is taken as in line2 and that is percentage of relevancy (output as in line3).

3.2 Crowd Sourcing Techniques

Crowd Selection is essential to crowd sourcing [18] applications, since choosing the right workers with particular expertise to carry out specific crowd sourcing tasks is extremely important. The central problem is simple but tricky: given a crowd sourcing task who is the right worker to ask? Currently, most existing work has mainly studied the problem of crowd-selection for simple crowd sourced tasks such as decision making and sentiment analysis. Their crowd selection procedures are based on the trustworthiness of workers. However for complex tasks such as document review and question answering, selecting workers based on the latent category of tasks is a better solution.

For every new worker added in system is created with attributes of certain density of percentage. Each user will be assigned attributes highest education degree, number of publications, profession so on. While entering their profile every worker will be put to personal questionnaire and based on their answers and attributes described above density attributes are initialized. There will be questions generated by system, based on the answers generated by questionnaire estimated reliabilities will be generated. D[i] is the sum of all density attributes. R[i] is the sum of all reliabilities. P[i] is the percentage generated by adding all the attributes. M is the mean calculated for all percentages and N is the mean calculated for all reliabilities R[n]. M and N are calculated for 100%. Average of M and N is the output. These steps are stated below in the Initial Credibility Calculation Algorithm

3.6 Algorithm - Initial Credibility Calculation Algorithm

1: **Input**: Worker Profile density attributes D[i] = {d [1], d [2], d[i] } and estimated reliabilities R from n control questions {R[n] ={r[1]…r[n]}.
Output: Initial Credible Percentage calculated from this algorithm
2: Each density attribute's d[i] calculated percentage is p[i].
3: P[i] is added for all attributes.
4: Mean M is calculated for all percentages P[i] -> {p[1],…p[m]} i.e. M = {p[1]+p[2]+…p[m]}/m
5: M is calculated for 100% M = M*100
6: Percentage based on p[i] is calculated for r[i].
7: Mean N is calculated and taken percentages p[j] -> {p[1]..p[w]}i.e. N={p[1]+p[2]+…p[w]}/w
8: N is calculated for 100%, N=N*100
9: Average of M and N is calculated and % is assigned as initial credibility to Worker.
10: Output is Average of M and N. i.e., Output = (M+N)/2

There are three kinds of groups in the crowd source workers. One with high percentage of credibility, one with medium percentage of credibility and one with less percentage of credibility. In this paper for answering queries generated by system i.e. to select the correct URL out of recommended URLs also we followed two different algorithms to select workers out of all workers. Selection of workers from crowd of workers is done by using the below two algorithms:

1. Random Algorithm selects three workers randomly from three groups of workers (high credible group, medium credible group, low credible group) and randomly selects one worker from the selected three workers. Thus the selection.

The minimum number of workers in the high credible group is considered as Min [workers (i)] and maximum number of workers in this high credible group is considered as Max [workers(i)]. Random number is generated within maximum and minimum of numbers and value 1 is added to the random number generated as in line1 in the algorithm. Similarly one user form medium credible group and one user from low credible group are selected. Out of these persons one random person is selected using the random number generated using the above same logic as in line 39. That user is the selected person using this Random Algorithm.

3.7 Algorithm - Random Algorithm:

Input: Set of three group of Workers w[i], y[i], z[i] where i ranges from 1..x
Output: Worker selected from this algorithm
1 Select random worker from first group.

```
2       Begin
3                       Minimum number of workers in the group is
4                       assigned as Min [w ( i )].
5                       Maximum number of workers in the group is
6                        assigned as Max [w ( i )].
7                       Random is generated within Max and Min of
8                       Numbers and value 1 is
9                     added to the random number assigned.
10      End.
11 Select random worker from second group.
12        Begin
13                      Minimum number of workers in the group is
14                       assigned as Min[y(i)].
15              Maximum number of workers in the group is
16                        assigned as Max[y(i)].
17                  Random is generated within Max and Min of
18              Numbers and value 1 is
19              added to the random number assigned.
20 End.
21 Select random worker from third group.
22 Begin
23              Minimum number of workers in the group is
24                  assigned as Min[z(i)].
25              Maximum number of workers in the group is
26                    assigned as Max[z(i)].
27              Random is generated within Max and Min of
28              Numbers and value 1 is added to the random number assigned.
29 End.
30 Select random worker p(i) from x(i),y(i),z(i) these selected three workers.
31 Begin
32                      Minimum number of workers in the group is
33                       assigned as Min[p(i)].
34                      Maximum number of workers in the group is
35                       assigned as Max[p(i)].
36              Random is generated within Max and Min of
37                 Numbers and value 1 is added to the random number assigned.
38 End.
39 Output is p(i), random worker from the three workers.
40 End.
```

2. Worker's Algorithm selects three workers from all groups randomly and assigns random number to each worker (from 1 to 10 value to them) and will take logarithmic values of those numbers and sort them and smallest value person is selected as worker.

As input three groups of workers(high credible group, medium credible group and low credible group) w[i], y[i] and z[i] where I ranges from 1 to x. Minimum number of workers in this group is assigned as Min [w(i) + y(i) + z(i)] and Max [w(i) + y(i) + z(i)]. Three random workers are selected out of this group as in step1. And selected workers is assigned random number as tag and let these workers be R(w), R(y), R(z). Take logarithmic value of these assigned tag numbers. Let these be Log[R(w)] , Log[R(y)], Log[R(z)] as calculated below in the algorithm at line 3. Sort these values as in line 4 and Smallest number is taken as output as in line 6. The algorithm below states the below steps.

3.8 Algorithm - Workers Selection Algorithm:

Input: Set of three groups of Workers w[i], y[i], z[i] where i ranges from 1...x
Output: One Worker selected from this algorithm.
1: Select three random workers from all groups.
Begin

Minimum number of workers in the group is assigned as Min [w (i) + y (i) + z (i)].
Maximum number of workers in the group is assigned as Max [w (i) + y (i) + z (i)].

Random is generated within Max and Min of Numbers three times and value 1 is added to the random number assigned.

End.
2: Assign random number from 1 to 10 to these selected workers R(w), R(y), R(z).
3: Take logarithmic value of these assigned numbers. Log(R(w)), Log(R(y)), Log(R(z)).
4: Sort these values of numbers Log(R(w)), Log(R(y)), Log(R(z)).
5: Pick the Smallest number out of the three sorted values of(Log(R(w)), Log(R(y)), Log(R(z))).
6: **Output:** Smallest of (Sorted Order (Log(R(w)), Log(R(y)), Log(R(z)))).

After answering the questions/URLs generated by system, every worker has option to provide feedback about their favorite worker in the system. That percentage of credibility assigned by the worker along with the previous history of credibility based on the update credibility algorithm, new percentage will be calculated and update the worker's percentage as their current credibility.

Input to this algorithm is peer review percentage pr(x) of worker x, initial credibility percentage i(x) and based on the group percentage will be assigned as in line1 and let that be y(x). Output

u(x) is average of three values pr(x), i(x) and y(x). Algorithm for Update Credibility is shown below:

3.9 Algorithm - Update Credibility Algorithm:

Input: Peer review percentage of worker x is pr(x), initial credibility of worker x is i(x), three groups g1,g2,g3 where g1 is the high credible group, g2 is the medium credible group and g3 is the low credible group.
Output: Update Credibility Percentage
1: If worker belongs to high credible group then worker will be assigned 100%, medium credible group will be assigned 70% and low credible user will be assigned 30%. This percentage is collected and assigned to variable y(x).
2: u(x)← (pr(x)+i(x)+y(x))/3
3: Output is u(x).

Chapter 4 Implementation & Validation

This chapter presents implementation and validation details of system. And also reports different experiments results. Section 4.1 presents implementation details of system. Section 4.2 presents Validation of example issue with the system. Section 4.3 reports experimental results.

4.1 Implementation Details of System

Our recommendation System is developed as a Web application built using distributed multi-tiered application model (as shown in Figure-4.1.1.1) for enterprise applications that interfaces with a relational database management system. We used J2EE along with Struts JSP as the main application framework interfacing with MySQL database and Tomcat as the server. Java programming enables secure and high performance software development on multiple platforms. Java also supports multiple features which we can go into details in the following sections.

4.1.1 System Tiers

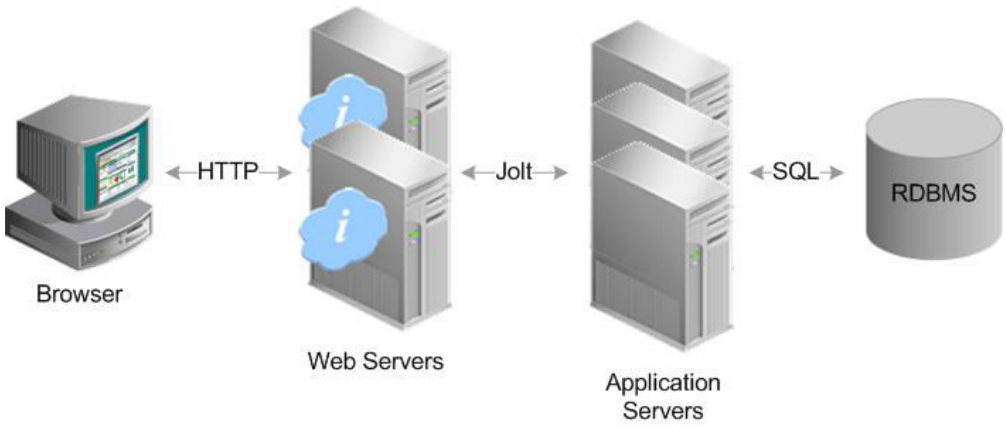


Figure 4.1- System Tiers

Our J2ee application system includes the following tiers. See Figure-4.1. Client tier components that run on client machines. Web tier components that run on the J2ee Server. Enterprise

Information System (EIS) that runs on the EIS Server. A Client tier is composed of the following components. A Web Client which consists of two parts: (1) dynamic Web pages that contain multiple types of markup languages such as HTML, XML etc. which are generated by components running in the Web-tier and (2) a Web browser. The Web tier is composed of the following component. JSP pages which are text based documents that are translated to servlets at run time. They execute as servlets but allow a more natural approach to creating static content. This is used in our system to display the Web pages to the user. The Enterprise Tier is composed of the following components. A Database system that employs MYSQL. In our case, the J2ee application needs access to enterprise information system for database connectivity.

The advantage of using J2ee is that it runs using the Java Virtual machine (JVM) such as Tomcat, WebSphere, and JBOSS etc. Every J2ee Program can be executed on any system platform as long as there is a JVM. This provides flexibility and system-platform independence. JSP is a java based application framework intended to simplify development integration of Web based user interfaces. It is a server side request driven Model-View-Controller framework used to construct user interfaces using components.

This architecture offers a clear separation between presentation and behavior. JSP ensures that applications are well designed with greater maintainability by integrating the MVC design pattern into its architecture. User Interface (UI) components represent View (typically in JSP), managed beans represent Model and Servlets is the controller. All requests are handled by this controller. Every request passes through and is examined by Servlet that calls various actions on the model (managed beans). The application is not restricted to MySQL database. We chose it as a default database. It can be linked to any relational database by modifying the Java database Connector (JDBC) driver. JDBC is the software component enabling the java application to interact with a database. It converts JDBC calls directly into a MySQL-specific protocol. We used the Apache Jena Java framework for interacting with the Ontology. Jena provides a collection of tools and java libraries to help semantic Web and linked data applications, tools and servers.

4.1.2 Configuration

One of the main important aspects of using J2ee is the ability to separate the Web interface from the business logic of the application. We start off by creating the Enterprise application module that will add automatically two different projects specific to it. The first project is known as Java Web handles the user interaction and the display of information (Web Application Archive) and the second project known as EJB module handles all business logic and any interaction with the database (Enterprise Java Bean). The link between these two projects is the actual Enterprise Application that coordinates the communication between them. The Web Application Archive (WAR) is used to distribute a collection of Java Server Pages, Java Servlets, Java Classes, XML files, tag libraries, static Web pages (HTML and related files) and other resources that together constitute a Web application. The EJB classes and the deployment descriptor should be bundled up in a JAR file. The WAR and JAR are the only necessary files to be deployed on the application Server. In this way, we have the ability and flexibility to deploy the projects on different servers.

In order to implement a J2EE Application, it needs to run on application Server. When creating the enterprise application, the appropriate Java EE version should be picked (Java EE6 in our case) as well as the application server (Glass Fish, JBoss, and WebSphere). Since we will be also using JSP, tiles in the framework, it should be also added as the main framework for the Java Web project. Otherwise tags and libraries will not be recognized in the application. After defining all the previous steps, we need to start with the configuration of the application server file (Web.xml). Fig 4.2 displays a sample configuration file for the application Server. In our case, since the application is integrated with JSP and MySQL, it also requires configuration of the server's global data source and its underlying JDBC Connection pool (resources.xml). Figure 4.3 displays a sample configuration file for the data source of the application server.

Establishing the connectivity between the MySQL database and the enterprise application is done through the Tomcat Server to which it deployed. This communication is made possible with the JDBC API. We start by creating a connection pool on Tomcat server. JDBC driver is required in order for the server to communicate directly with the MySQL database. A Connection pool contains a group of reusable connections for a database. Creating a new physical connection is time consuming, that is why the application server maintains a pool of available connections. In this way the performance of database connections is improved. Connection pools use

a JDBC driver to create physical connections to database. In order to enable our enterprise application access to the MySQL Database, a JDBC source that uses the connection pool should also be created. JDBC resource will provide the application with the means of connecting to the database. All external libraries used in this application such as MySQL, log4j etc. should be added to the appropriate folder as a JAR file so that the application can use their classes and methods.

4.1.3 Directory Structure

As we previously mentioned, the enterprise application has Web project that contains the user interface and interaction, and the java project containing the business logic and the communication with the database.

The root directory of the Web application is called the document root. It contains a private directory named WEB-INF. Any files that reside under the WEB-INF directory are private otherwise they are public. Since Web.xml and struts-config.xml are two important configuration files for the application, they reside under the WEB-INF directory so that they cannot be accessed directly from browser by specifying the URLS to their paths. The JSP pages reside in the Web folder that is under the root folder. They are the main interface files to the application. All the styling and client side manipulation files (CSS/JavaScript) reside in a separate folder also under the Web folder. We also separated the back office files from the rest of the interface files to distinguish between them. The source packages are in the src directory. They contain backing beans or managed beans that are UI Components used in a particular page. A typical JSP with tiles application contains one or more managed beans. Our managed beans typically handle login/logout services, registration forms, calls to the session beans and basic services that are required from a UI interface. The Test packages reside in a separate folder under the root directory. All the test files and test suites are handled here.

4.1.4 Error Logging

Logging is an important and pretty useful mechanism for every application. It can help developers debug and improve their code or test its functionality. Every application has a tendency to crash or throw an error at any point in time. This can be result of many factors such as lack of

memory allocation, poor exception handling or even hardware problems affecting and preventing the application from running efficiently. Therefore it is very important to figure out the reason behind all these errors. We used log4j library to log all the information and errors that result in the application. Every method implemented has its own logger information that feeds it from start to finish. As long as the application is running, the log file will continue to save the information about the method on call. The log file is very useful when the developer wants to debug a certain problem or crash in the system. It is a good starting point if the built in debugger is of no use. The log file can be used to determine the reason behind the error and save time.

4.2 Validation of System

This section describes the validation of real world system problem and its solution using the system. This experiment shows the performance and accuracy of the techniques used in the system and shows the improvement over the traditional techniques used and the latest algorithm techniques and data structures like hash maps.

When user types "**http://dbpedia.org/resource/Cambridge**" in the URL tab user will get set of good URLs related to Cambridge and if there are any bad URLs in DBPedia the system will list out the bad URLs in the page results of Cambridge. Our system will take the list of bad URLs and finds the recommended URLs to the user. Suppose **www.redff.com** is the bad URL outcome from the list of URLs. Figure 4.2shows the main screen where user can type the URL and Figure 4.3 shows the list of good and bad URLs listed out by system. Figure 4.4 shows the links (possible URLs generated by search engines and refined list coming from the search engines) that works for dead URLs. In Figure 4.4 it shows the domain names of the refined search URLs and domain names are listed. And also figure displays the domain names that match with bad URLs, after domain names are fetched, semantic text matching algorithm is applied and domain names that are filtered out with same domain name that match with bad URL are applied with semantic text matching.

Example: www.rediff.com, www.rediff.com/index.php

For both the URLs domain name will be matching with bad domain URL rediff.com. So semantic text matching algorithm is applied and less distance URL is filtered.

After all these results fetching there is link on Figure 4.5 top-left, **"Go to User Recommendations Page**". When user clicks on that link it navigates to User recommendations page as shown in Figure 4.5. There the results of recommendation URLs after applying all the algorithms will be displayed in the order of semantic text algorithm selected in first page as shown in Figure 4.2. In our case it is "**Block Distance**" Algorithm. Then two workers who are in session will be picked using Randomizing algorithm and Workers Selection Algorithm. These workers will pick the URL which they think is close to the answer. As soon as they submit the "Submit" button, alerts of whatever they are selected as shown in Figure-4.7 will be displayed to user. After selecting the URL the accuracy of selected URL in terms of percentage is calculated using page ranking algorithm and is displayed as per the Figure-4.8. And in the same screen there is "**Peer Review**" link that takes to the page as shown in Figure-4.9. There the same worker can give peer review rating to other worker in system, which calculates the update credibility percentage and updates in the system. This ends the use case.

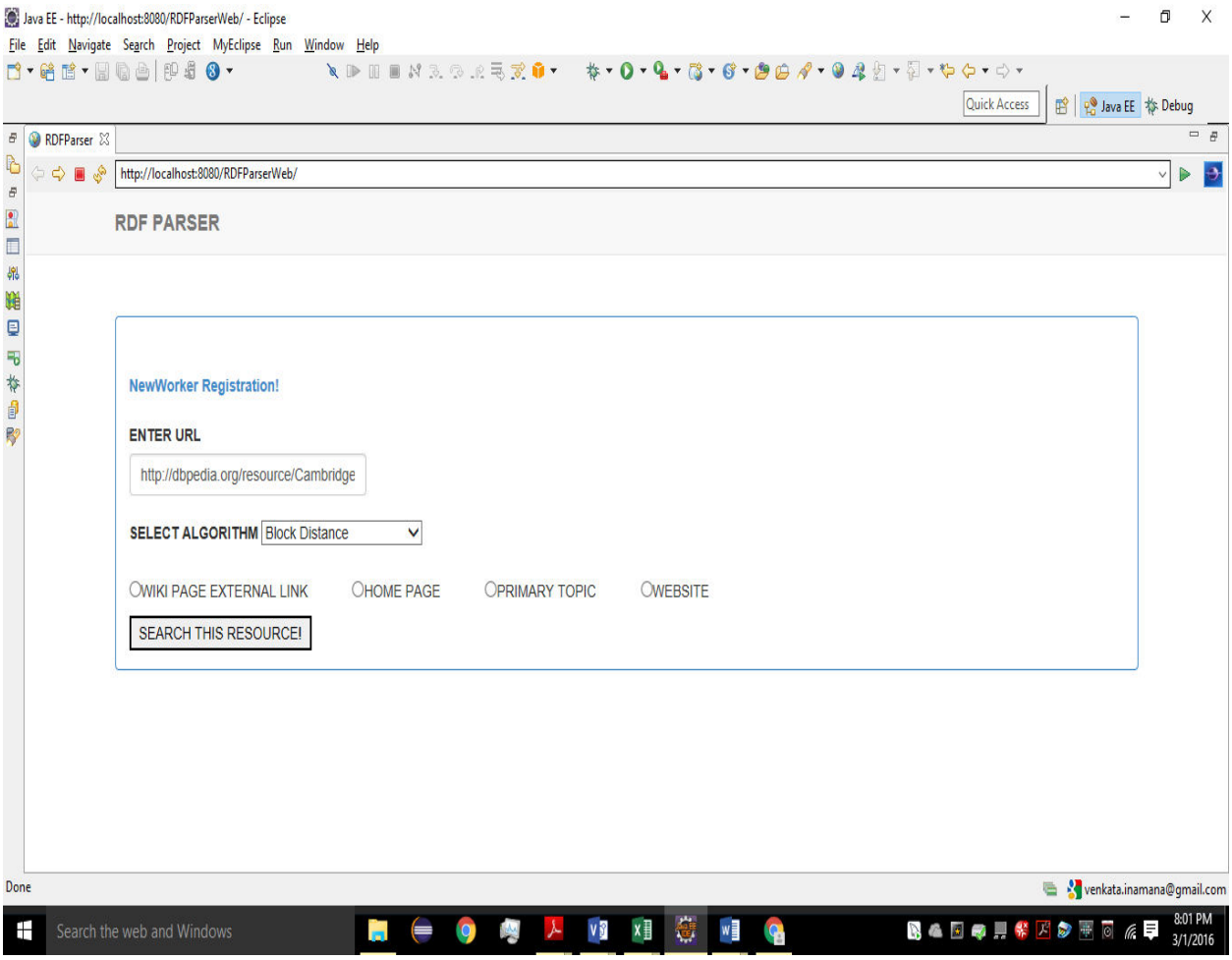


Figure 4.2 - Main Screen

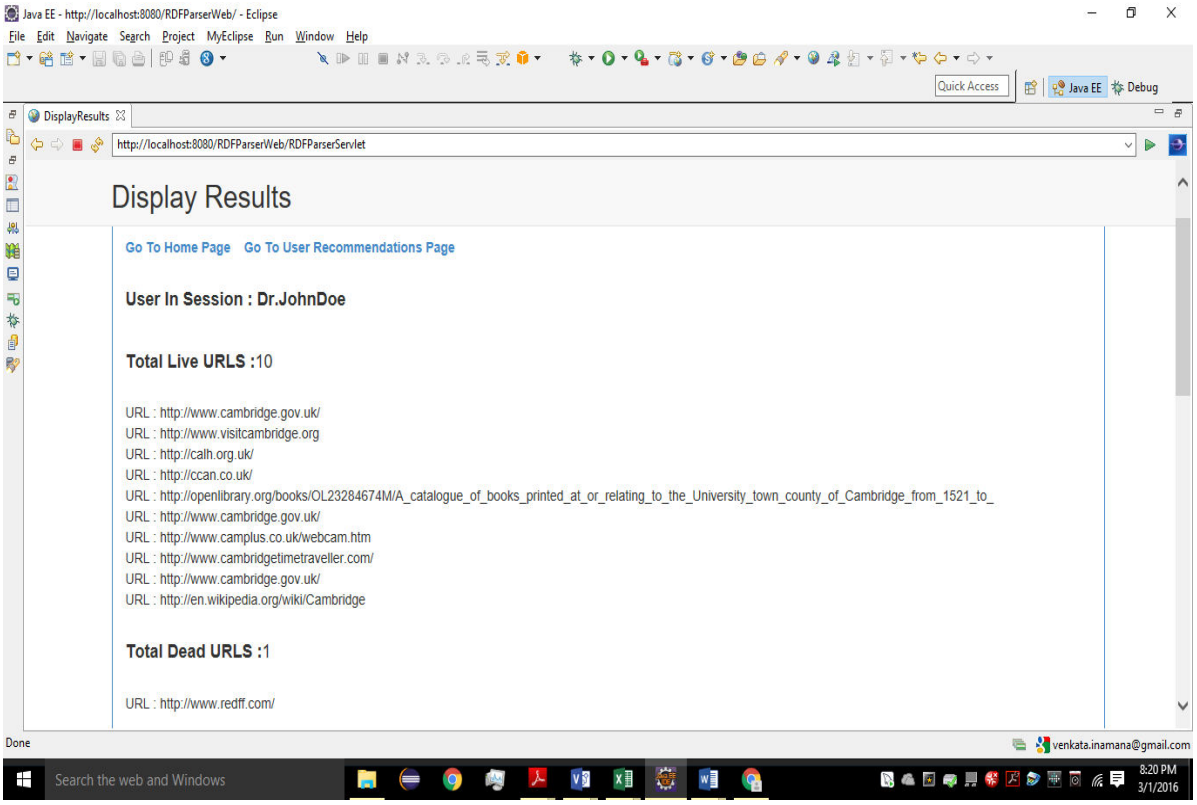


Figure 4.3 - Display Results Screen

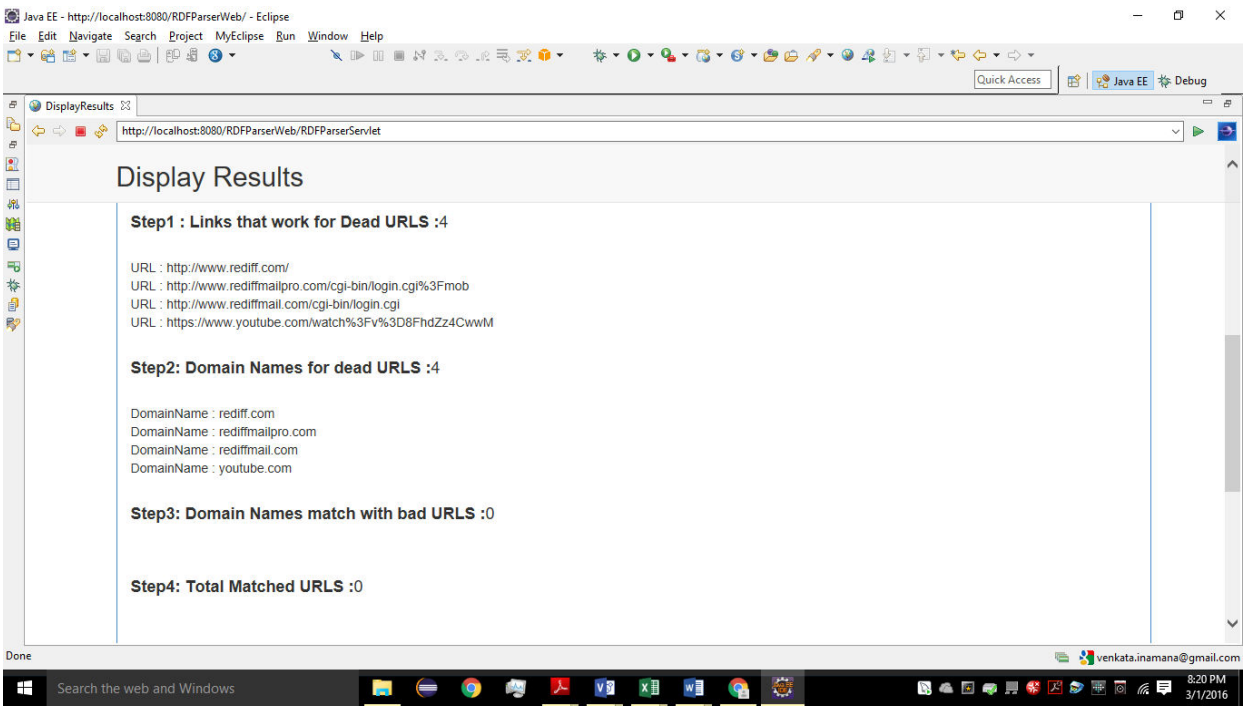


Figure 2 - Display Results Screen (Cont'd)

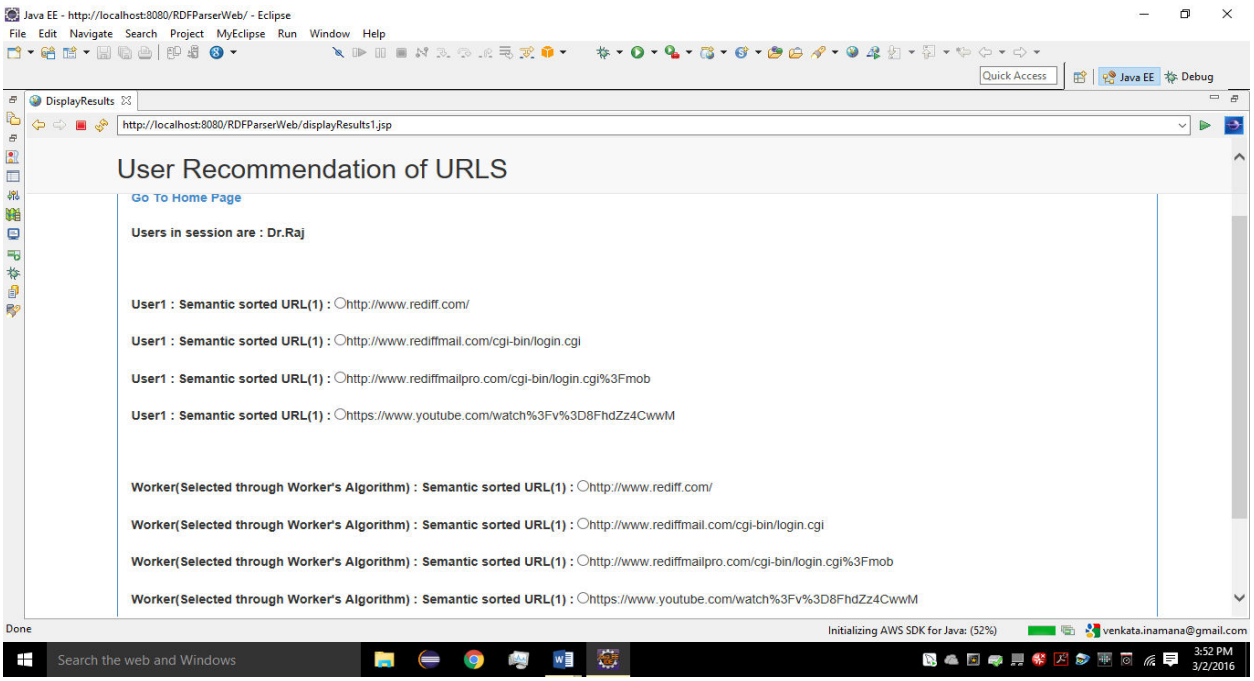


Figure 4.5 - User Recommendation Screen

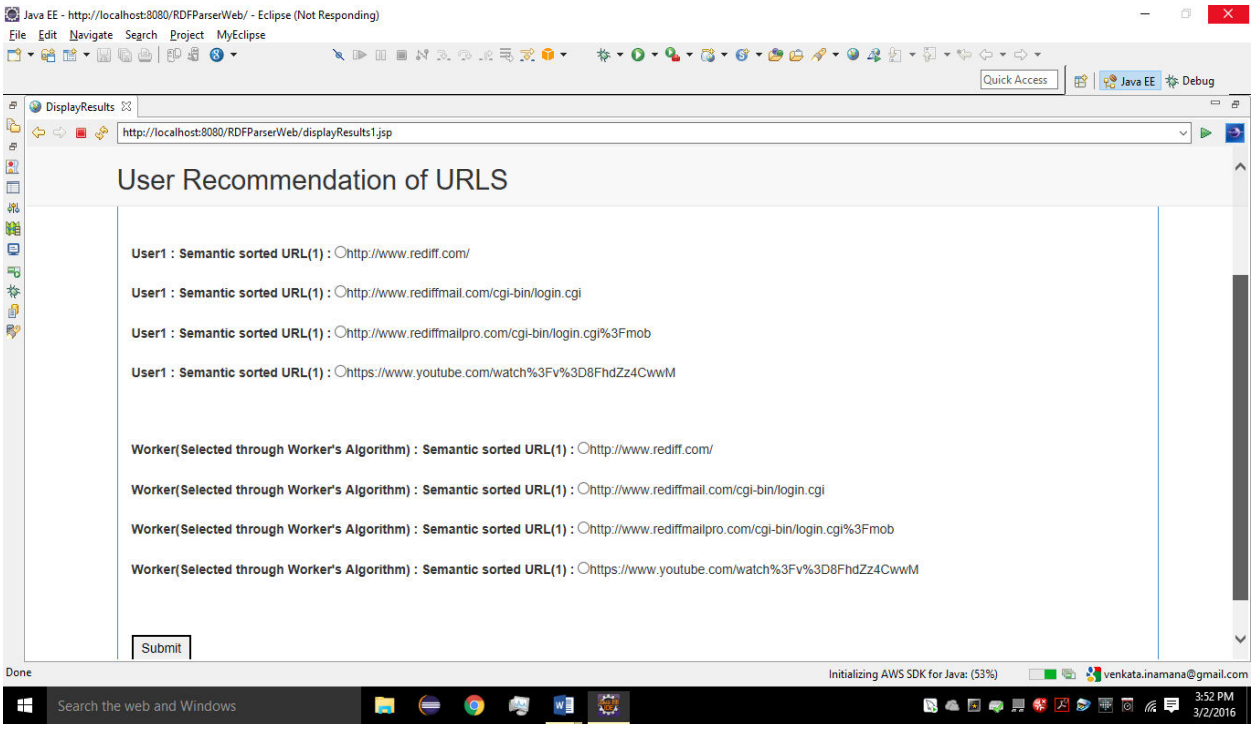


Figure 4.6 - User Recommendation Screen (Cont'd)

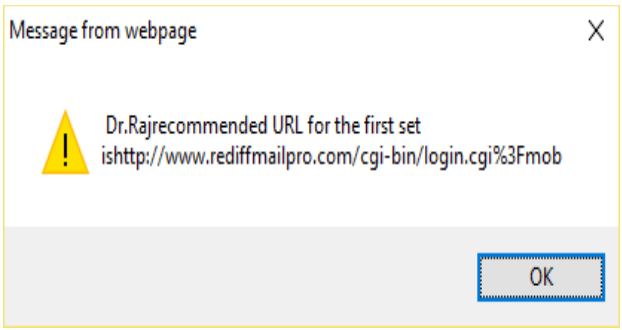


Figure 3 - popup message

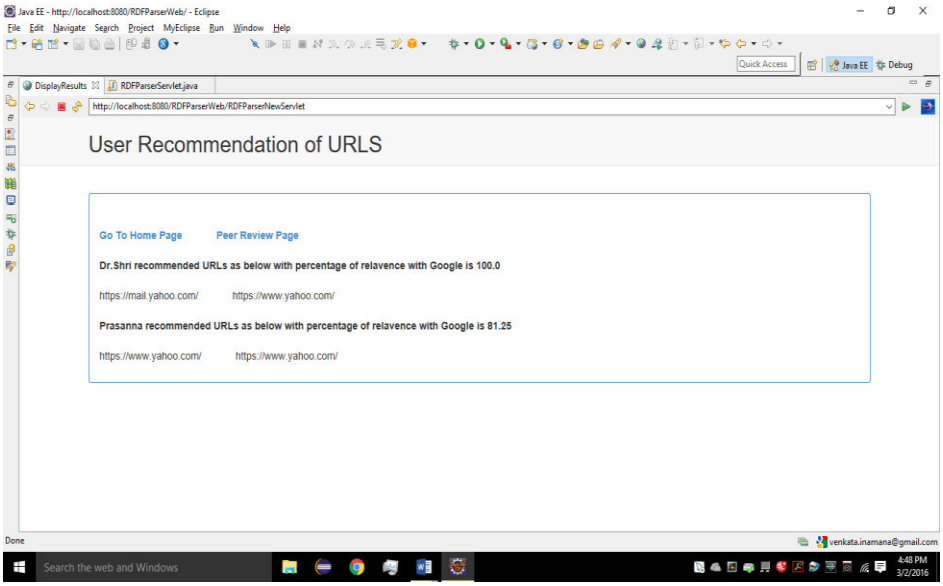


Figure 4.8 - Page relevancy percentage Screen

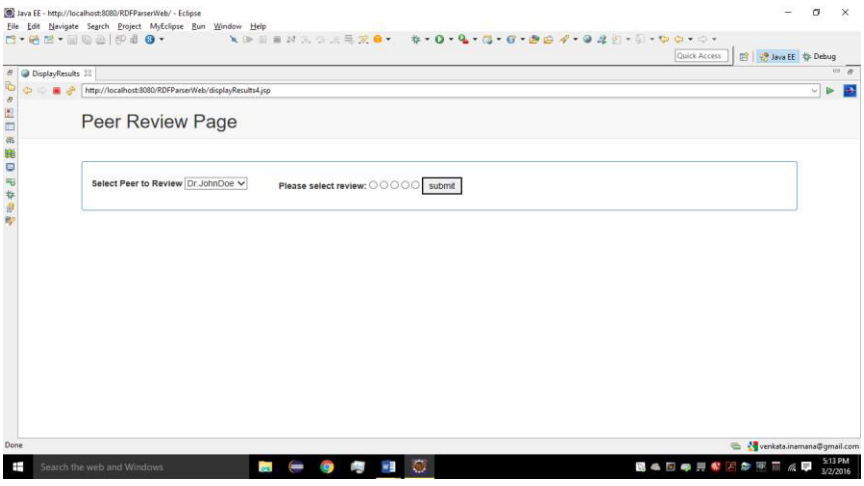


Figure 4 - Peer Review Screen

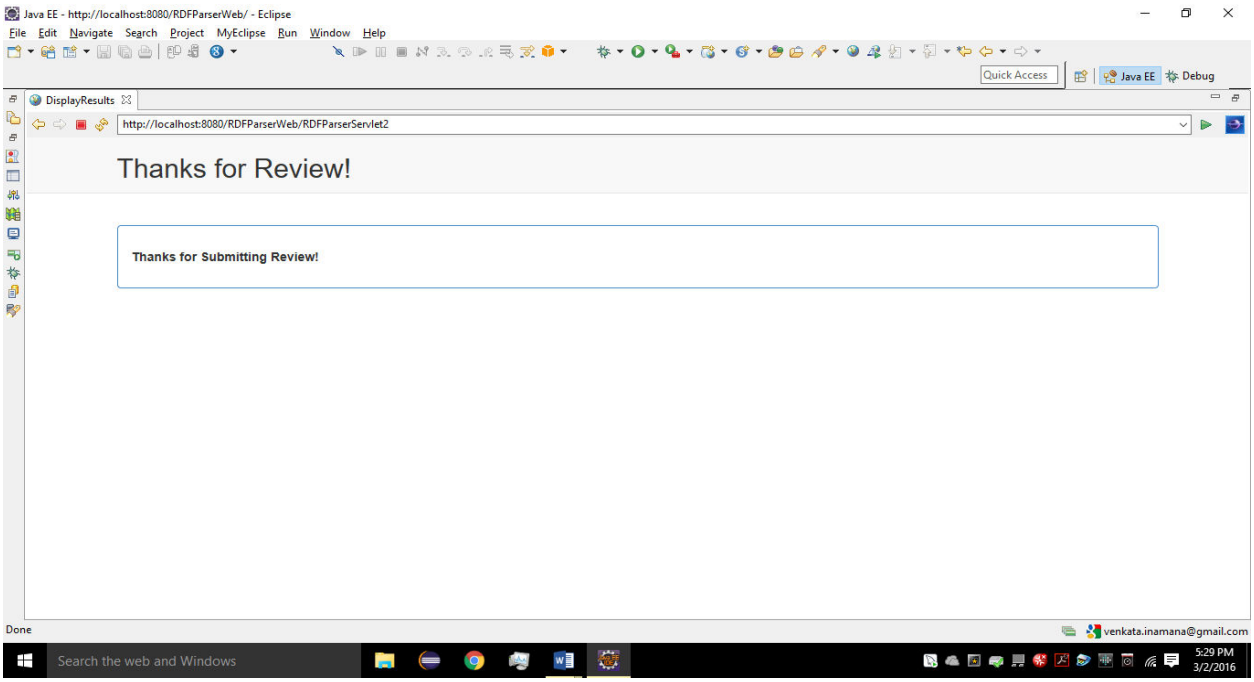


Figure 5 - Peer Review Response Screen

4.3 Experiments

To evaluate the performance of our recommended techniques and algorithms, we performed several test cases and experiments. Experiment programs are implemented in Junit with Intel quad core (2.2 GHz) CPU and 4GB memory running on Windows Operating System. We simulated experiments to generate tables, queries and test data. The simulated meta-data and information are used to analyze the output and the recommendation system. We generated 600 bad URLs in our experiments. We analyzed different scenarios in algorithms and the results generated in the system.

We used JUnit as the unit testing framework that has been designed for the purpose of writing and running tests. Unit testing is very important and critical in developing any application to monitor how the application is behaving and whether the results of experiments are expected. The framework used creates a relationship between development and testing. We start coding according to the specifications and use the test runners to verify how much the output deviates from the intended goal. Therefore while developing the application; we performed test cases on specific functionalities to verify the outcome. A test case is a code fragment that checks another

code unit (method) works as expected. Using this approach, we are capable of easily correcting bugs as they are found and changing requirements as we proceeded with the system. We conducted several experiments to test the effectiveness of our recommendation system. Some of which are evaluation of the recommendation technique using Junit for the algorithms, comparing the efficiency of algorithms using matrix vs not using matrix and using the search algorithm vs estimated outputs generated by system to determine how efficiently system is working.

Using the unit testing process will make sure that all the modifications in the code will not break the system since all the alterations and test methods do not interfere with the code. In the experiments below, we measure the precision and recall [21] of the recommendation system for a different number of generated test data sets. Precision is also called positive predictive value and is the fraction of retrieved instances that are relevant. Precision is based on understanding and measure of relevance. Suppose a computer program for recognizing dogs in scenes from a video identifies 7 dogs in a scene containing 9 dogs and some cats. If 4 of the identifications are correct, but 3 are actually cats, the program's precision is 4/7 while its recall is 4/9. When a search engine returns 30 pages only 20 of which are relevant while failing to return 40 additional relevant pages, its precision is 20/30 = 2/3. Aim of the graph is to measure the precision percentage and plot graph between 4-correct URL (System generated 4 URLs and predicted output 4 URLs – 4/4), 3 correct URLs (System generated 4 URLs and predicted output 4 URLs – 3/4), 2 correct URL (System generated 4 URLs and predicted output 4 URLs – 2/4), and 1 Correct URL (System generated 4 URLs and predicted output 4 URLs – 1/4).

Percentage of Precision is 0.75 for the system generated outputs vs predicted outputs and calculated percentages of precision as shown in Figure 4.11.

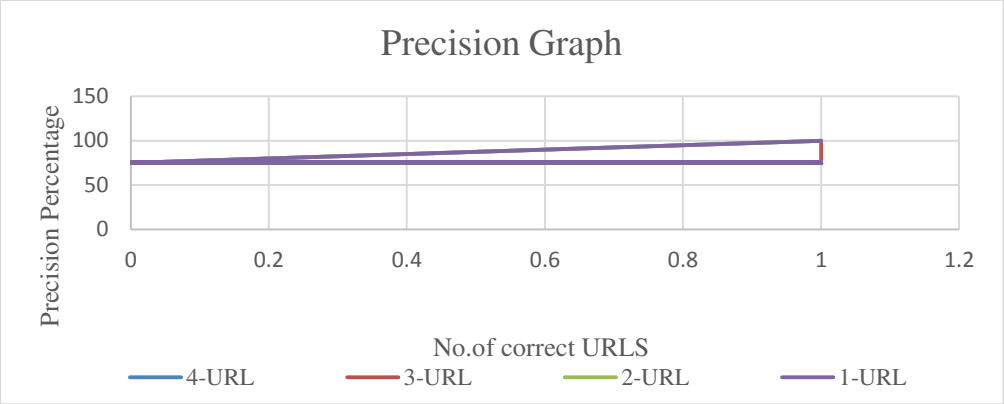


Figure 4.11 - Precision Graph

Recall is also called Sensitivity. Recall is based on understanding and measure of relevance. Suppose a computer program for recognizing dogs in scenes from a video identifies 7 dogs in a scene containing 9 dogs and some cats. If 4 of the identifications are correct, but 3 are actually cats, the program's precision is 4/7 while its recall is 4/9. When a search engine returns 30 pages only 20 of which are relevant while failing to return 40 additional relevant pages, its recall is 20/60 = 1/3.

Aim of the graph is to measure the recall percentage and plot graph between 4-correct URL (System generated 4 URLs and predicted output 4 URLs – 4/4), 3 correct URLs (System generated 4 URLs and predicted output 4 URLs – 3/4), 2 correct URL (System generated 4 URLs and predicted output 4 URLs – 2/4), and 1 Correct URL (System generated 4 URLs and predicted output 4 URLs – 1/4).

Percentage of Recall is 0.75 for the system generated outputs vs predicted outputs and calculated percentages of recall as shown in Figure 4.12.

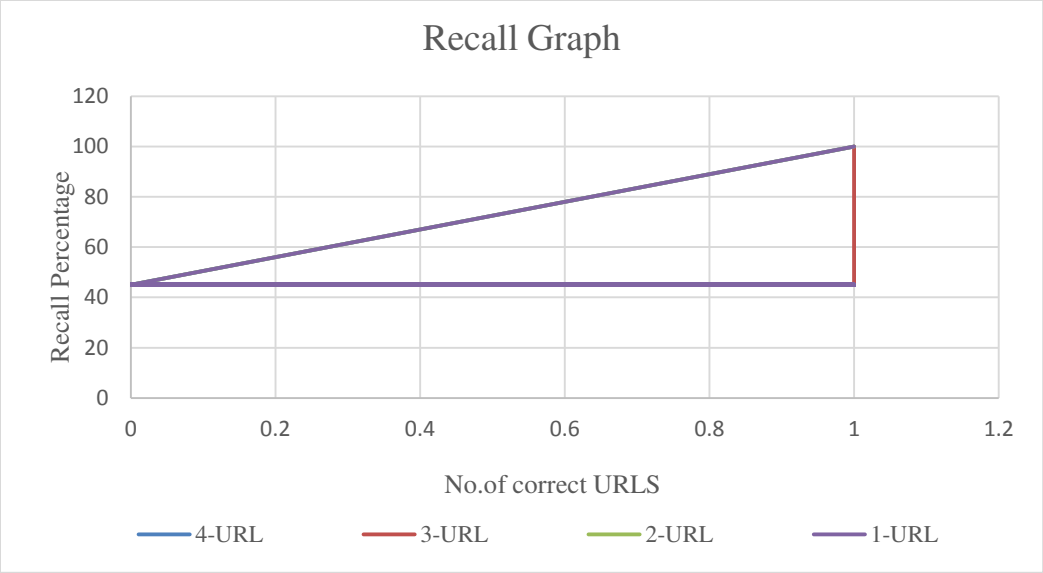


Figure 4.12 - Recall Graph

The following set of experiments display the results of the recommendation system for different set of bad URLs. Each of the figures shows the curves along with the results. These curves are the comparison of system generated results vs estimated output results.

Example:

Suppose bad URL is www. Yhoo.com the following are the system and estimated outputs. The following is the generated data (i.e., estimated output of system)

www.yahoo.com

www.sports.yahoo.com

www.news.yahoo.com

www.mail.yahoo.com

And system generated output are:

www.yahoo.com

www.sports.yahoo.com

www.news.yahoo.com

www.mail.yahoo.com

Now system generated output matches for 4 URLs. That means output is 100% as shown in Figure 4.13.

In this way we generated outputs for 600 bad URLs and generated graphs for 4 set of correct URLs (i.e., 4 URLs generated by system and 4 predicted URLs matches), 3 set of correct URLs (i.e., out of 4 URLs generated by system and 4 predicted URLs matches if 3 set matches) as shown in Figure 4.14, 2 set of correct URLs as shown in Figure 4.15 and 1 set of correct URLs as shown in Figure 4.16

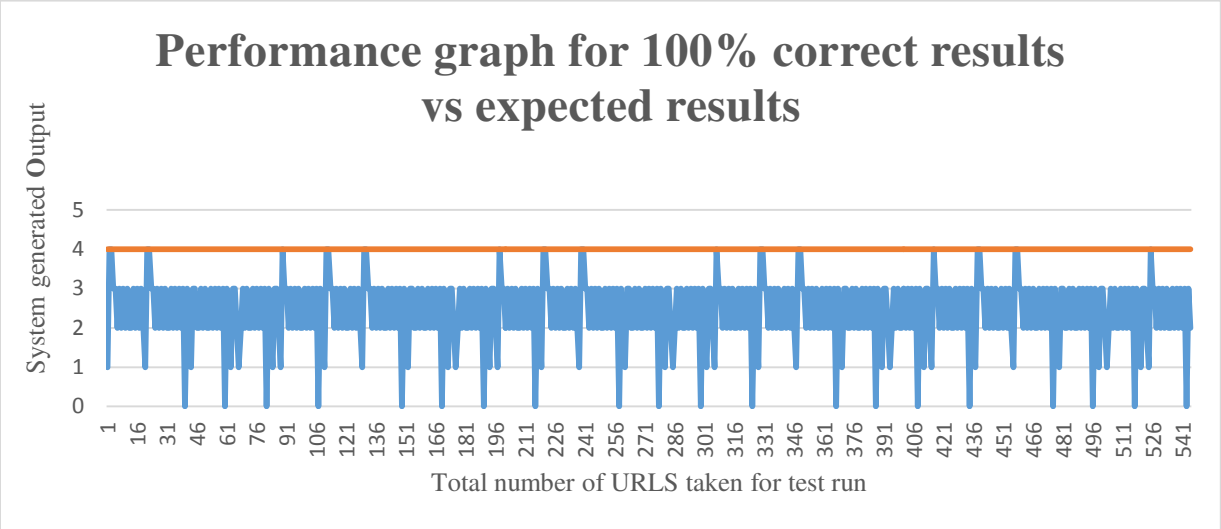


Figure 4.13 - Performance Graph for 100% correct results

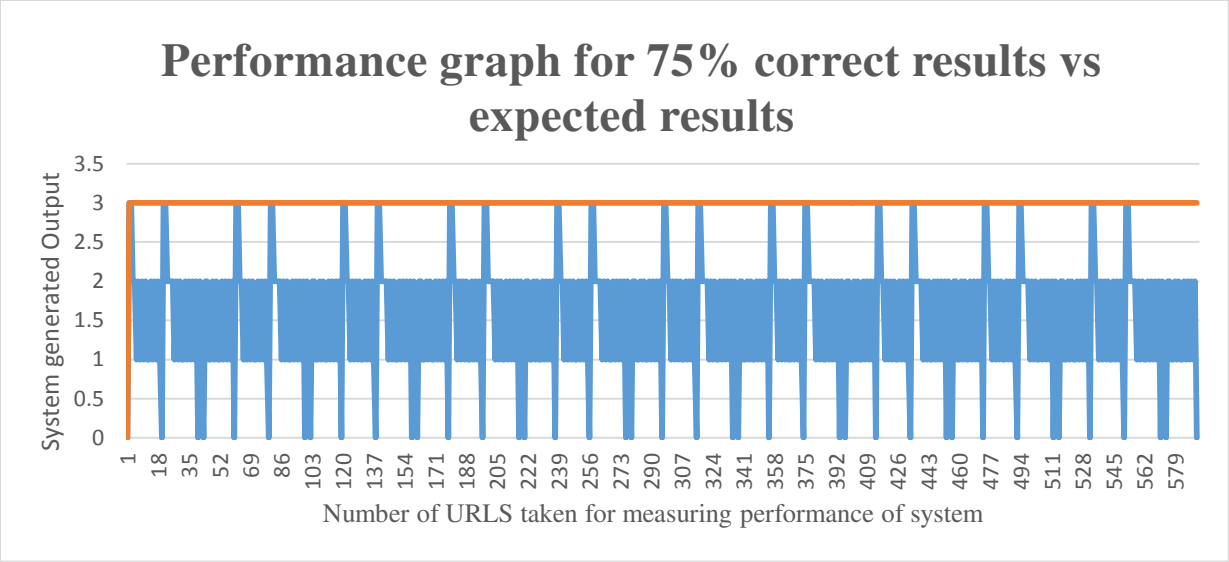


Figure 4.14 -Performance Graph for 75% correct results

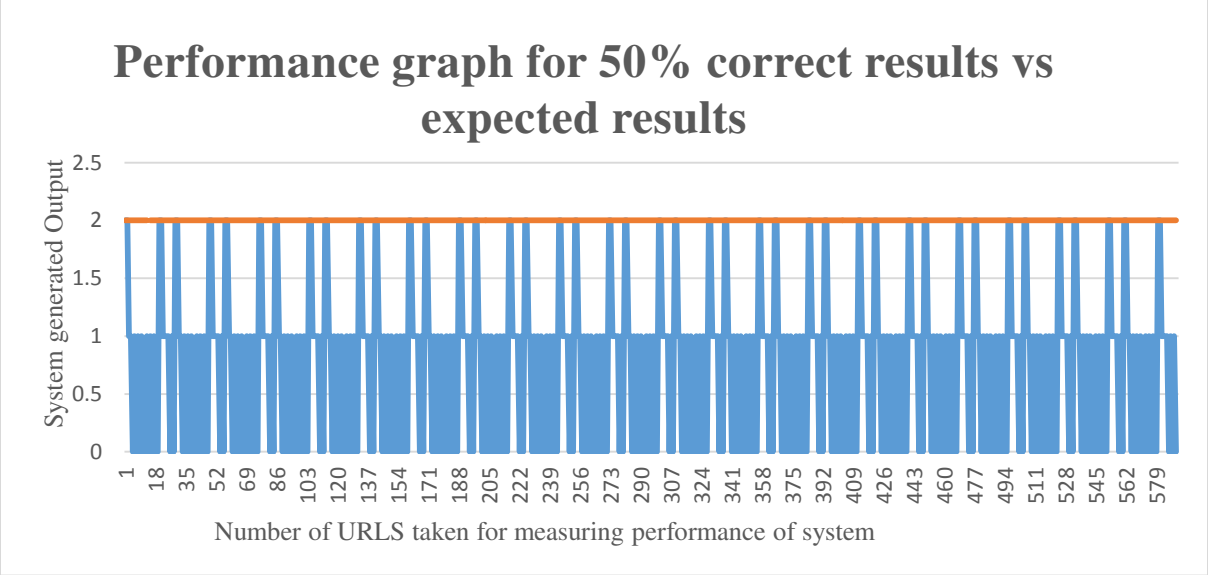


Figure 4.15 - Performance Graph for 50% correct results

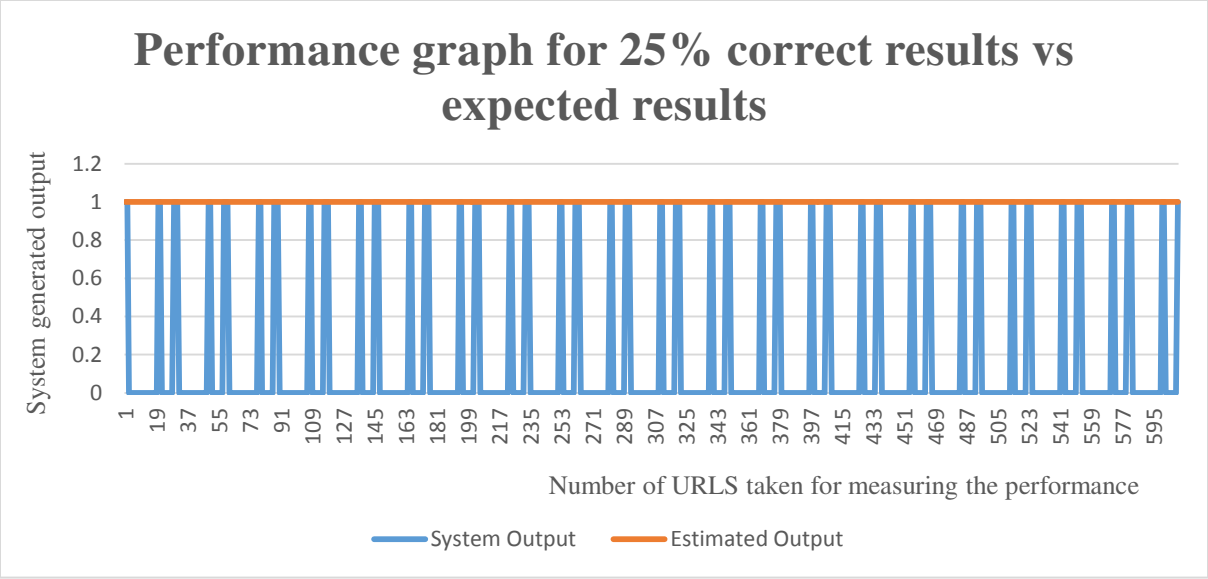


Figure 4.16 - Performance Graph for 25% correct results

And we conducted Junit test cases for crowd sourcing algorithms also. These test cases will measure the performance of algorithms (like time taken, number of times same worker is repeated as output) and all the data attributes are considered to analyze the test results. And for both the algorithms (Randomizing and Workers selection algorithm) results came out so well and performance is so high and major outcome observed is there is no repetition of workers coming as output. The graphs plotted are against 100 workers and 100 times experiment is conducted and number of times same worker is repeatedly picked as output from the algorithm. Graphs show that they are picked maximum 2 times but not many times. Below are the graphs (as shown in Figure 4.17 and Figure 4.18) plotted based on our experiments.

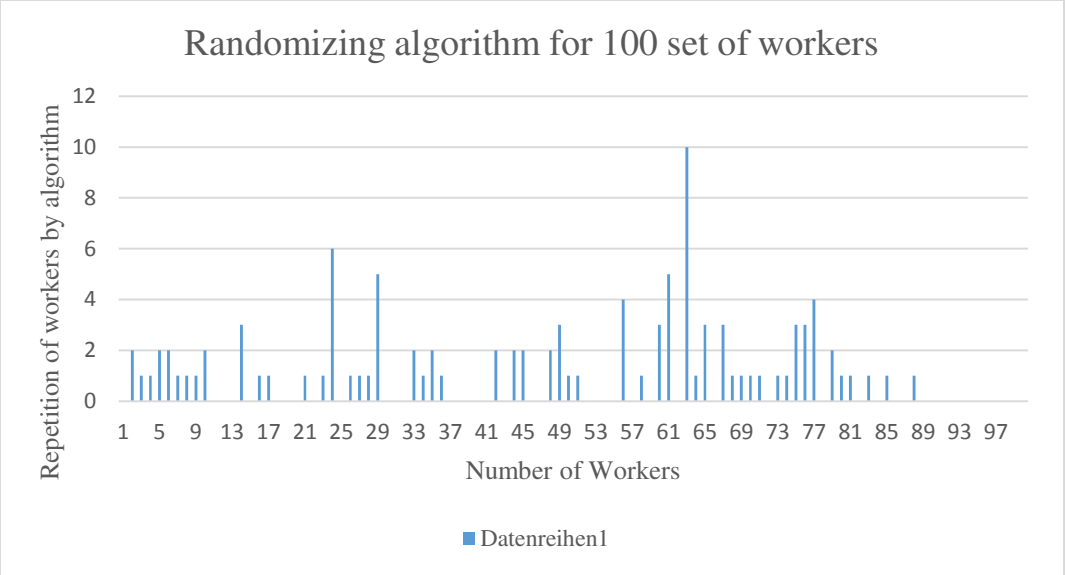


Figure 4.17 - Random Algorithm Performance Graph

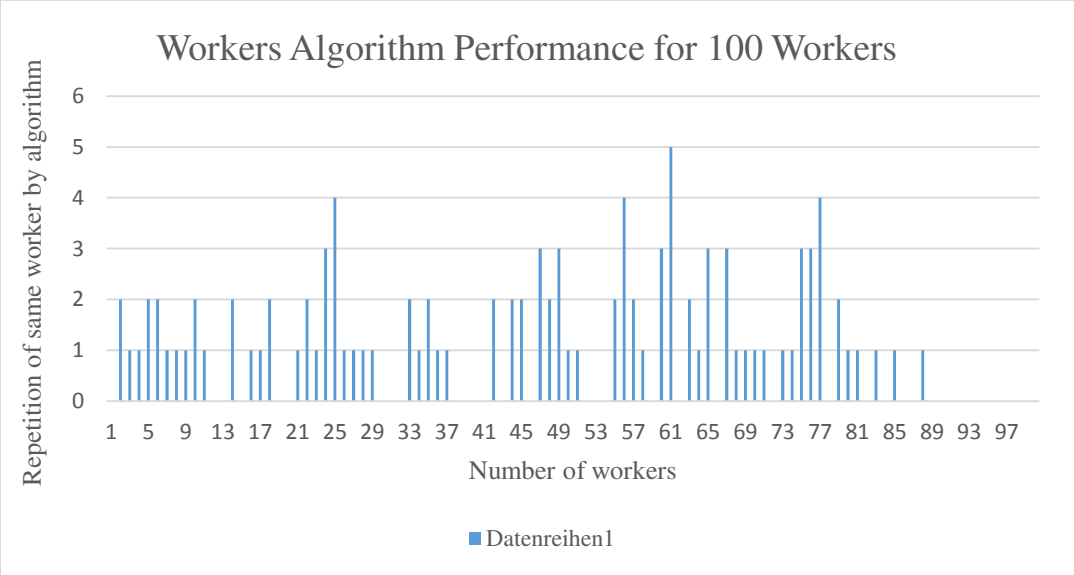


Figure 4.18 - Workers Algorithm Performance Graph

After analyzing all the experiments results, our recommended techniques are proved to be reasonably precise. It also proved to be much faster and more efficient. The results are returned fastest to user. One of the reasons is data structures are implemented as linked hash maps. The main advantage of using hash maps over other data structures is speed especially when the data is large. That alone is very important factor to minimize the time of the search process.

Chapter 5 Conclusion

There is an increasing demand to find the recommendation of good set of URLs that matches closely for bad URLs. With such an increase in demand, there should be a way to help the users minimize the amount of work that they need to do with specific recommendations to bad URLs in a system. Some recommendations are very accurate and some are around 75% accurate. No existing work has been addressed in this issue and also has not been reported in the literature.

In this thesis, we have presented recommendations that are useful for the users to further traverse if they are looking for the same URL. Recommendation of URLs is a complicated problem. It involves techniques such as domain matching, semantic text matching as well as search algorithms which are based on search engine results.

We have presented a set of rules that assign from multiple algorithms whenever they are applicable. Each approach we have made use of, is discussed. To achieve the accuracy at every step we have ensured our developed code and also unit tested so that no issues will encounter at a later point. Our experiment with lot of test data shows the effectiveness and performance of system.

Basically the goal of the research is proposing a metric-driven framework for predicting the quality of linked open data sets from an inherent point of view. To achieve this goal, we have followed an approach which is started by analysis of well-known frameworks. To put the proposed algorithms and metrics in place, an automated tool is developed to ensure good links recommendation for bad URLs.

An important direction for future work is confirming the effectiveness of **LinkWiper** System through user studies making use of real world applications on the Linked Data Open Cloud. In such studies, users are likely to generate some incorrect feedback, which would enable us to validate the robustness of **LinkWiper** System beyond our current set of experiment.

References

[1]- S. Embury, B. Jin, S. Sampaio, and I. Eleftheriou. On the feasibility of crawling linked data sets for reusable defect corrections. In Proceedings of the 1st Workshop on Linked Data Quality (LDQ2014), volume 1215. CEUR Workshop Proceedings, 2014.

[2] RDF - Z. Abedjan, T. Gruetze, A. Jentzsch, and F. Naumann. Profiling and mining RDF data with ProLOD++. In 30th IEEE International Conference on Data Engineering (ICDE), pages 1198–1201, 2014.

[3]Semantic web: S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In Proc. Int. Semantic Web Conf. (ISWC). 2007.

[5]Data quality principles- [1] C. Batini and M. Scannapieco. Data Quality: Concepts, Methodologies and Techniques (Data-Centric Systems and Applications). Springer-Verlag New York, Inc., Secaucus, NJ, USA,2006.

[6]wwwC. Bizer, T. Heath, K. Idehen, and T. Berners-Lee. Linked data on the web. In Proc. Int. World Wide Web Conf. (WWW), 2008.

[7]Tim Berners Lee C. Bizer, T. Heath, and T. Berners-Lee. Linkeddata-the story so far. Int. Journal on Semantic Web and Information Systems, 5(3), 2009.

[9]SPARQL - B. Quilitz and U. Leser. Querying distributed RDF data sources with SPARQL. In The Semantic Web: Research and Applications. Springer, 2008.

[10] – Quality assessment for linked data – Anisa Rula, Amrapali Zaveri,,published in semantic web journal.net

[11]ALEX –Ahmed El-Roby, Ashraf Aboulnaga, Automatic Link exploration in linked data, published in SIGMOD'15, ACM international conference of management of data.

[12] .Pipino, L.L., Lee, Y.W., Wang, R.Y.: Data quality assessment. Communications of the ACM 45, 211-218 (2002)

[13]. Lee, Y.W., Strong, D.M., Kahn, B.K., Wang, R.Y.: AIMQ: a methodology for information quality assessment. Information & management 40, 133-146 (2002)

[14]. Batini, C., Cappiello, C., Francalanci, C., Maurino, A.: Methodologies for data quality assessment and improvement. ACM Computing Surveys (CSUR), vol. 41, pp. 16 (2009)

[15]. Naumann, F., Rolker, C.: Assessment methods for information quality criteria. In: 5'th Conference on Information Quality pp. 148-162. (2000)

[16]. Batini, C., Scannapieca, M.: Data quality: concepts, methodologies and techniques. Springer (2006)

[17]. Behkamal, B., Kahani, M., Paydar, S., Dadkhah, M., Sekhavaty, E.: Publishing Persian linked data; challenges and lessons learned. In: 5th International Symposium on Telecommunications (IST), pp. 732-737. IEEE, (2010)

[18]. Helfert, M.: Managing and measuring data quality in data warehousing. In: WorldMulticonference on Systemics, Cybernetics and Informatics, pp. 55-65. (2001)

[19]. ISO: ISO/IEC 25012- Software engineering - Software product Quality Requirements and Evaluation (SQuaRE). Data quality model, (2008)

[20]. A. Aboulnaga and K. El Gebaly. μbe: User guided source selection and schema mediation for internet scale data integration. In IEEE Int. Conf. on Data Engineering (ICDE), 2007.

[21]..M. Acosta, A. Zaveri, E. Simperl, D. Kontokostas, S. Auer, and J. Lehmann. Crowdsourcing linked data quality assessment. In Proc. Int. Semantic Web Conf. (ISWC). 2013.

[22]. S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In Proc. Int. Semantic Web Conf. (ISWC). 2007.

[23]. D. Aumueller, H.-H. Do, S. Massmann, and E. Rahm.Schema and ontology matching with OMA++. InProc. ACM SIGMOD Int. Conf. on Management of Data, 2005.

[24]. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. Scientific American, 2001.

[25]. I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. ACM Trans. On Knowledge Discovery from Data (TKDD), 2007.

[26]. C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. Int. Journal on Semantic Web and Information Systems, 5(3), 2009.

[27]. C. Bizer, T. Heath, K. Idehen, and T. Berners-Lee.Linked data on the web. In Proc. Int. World Wide Web Conf. (WWW), 2008.

[28]. Roomba - Automatic Validation, Correction and Generation of Dataset Metadata, by Ahmed Assaf, Aline Senart, and Raphael Troncy, published in International World Wide Web Conference, (WWW), 2015.

[29]. Repairing broken RDF links in the web of data by Mohammad Pourzaferani, Mohammad Ali Nematbakhsh , published in International Journal of Web Engineering and Technology 8(4):395-411 · February 2013

[30]. Analyzing broken links on the web of data: An experiment with DBpedia by Enayat Rajabi, Salvador Sanchez-Alonso, Miguel –Angel Sicilia , published in April 08/2014.