DATA CLEANING ON GRAPH DATABASES USING NEO4J: SPELLING CORRECTION USING ONTOLOGY AND VISUALIZATION

By

SARATH KUMAR MADDINANI

Bachelor of Electronics & Communications

Jawaharlal Nehru Technological University

Hyderabad, Andhra Pradesh, India

2008 - 2012

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2016

DATA CLEANING ON GRAPH DATABASES USING
NEO4J: SPELLING CORRECTION USING
ONTOLOGY AND VISUALIZATION

Thesis Approved:

Dr. Johnson P Thomas

Thesis Adviser

Dr. Christopher Crick

Dr. David Cline

Name: SARATH KUMAR MADDINANI

Date of Degree: December, 2016

Title of Study: DATA CLEANING ON GRAPH DATABASES USING NEO4J: SPELLING CORRECTION USING ONTOLOGY AND VISUALIZATION

Major Field: COMPUTER SCIENCE

Abstract: Data cleaning, also called data cleansing is the process of detecting, correcting, or removing errors and inconsistencies from data in order to improve data quality. Data is classified into two types, namely, structured and unstructured. Standard techniques and tools are available to handle structured data. Most of the data generated in today's world is unstructured. A graph database stores data in the form of nodes and relationships between the nodes. Neo4j is a graph database tool which is accessed using the Cypher query language. We define and implement an extensive set of cleaning algorithms for spelling corrections using ontology and other inconsistencies in data in Neo4j. We conclude with the validation of these cleaning algorithms using data visualization techniques. The visualized results of the data before and after applying the cleaning algorithms proves the effectiveness of the proposed cleaning algorithms.

TABLE OF CONTENTS

Chapter Page

I. INTRODUCTION 1
Data Cleaning 1
Problems in Data Cleaning 2
Problems with existing approaches 2
Data Quality Problems 2
Data cleaning in unstructured data 3
Proposed Approach 4
Thesis and Research Objectives 5
Outline of Thesis 5

II. REVIEW OF LITERATURE 6
Data cleaning approaches 6
Data cleaning methods 7
Duplicate Elimination 8
Structured and Unstructured Datasets 15
Graph Databases 16

III. NEO4J 20
Neo4j 20
Cypher Query Language 21

IV. PROPOSED WORK 22
Overview 22
Design and steps of Data Cleaning Model 22
Implementation 25
Algorithms and approaches 25

V. DATA VISUALIZATION & SIMULATION RESULTS 31
Data Visualization & Tools Used 31
Results 32

VI. CONCLUSIONS & FUTURE WORK 39

REFERENCES 41

LIST OF FIGURES

Figure Page

1.1 Classification of data quality problems in data sources 2
2.1 Nodes 17
2.2 Nodes connected through Relationships 17
2.3Node having relationship to itself 17
2.4 Property Graph 19
4.1 Detailed Architecture of Data cleaning 24
4.2 Ontology 29
5.1 Graph for distinct count of current state before and after cleaning 32
5.2 Geolocation for current state before cleaning 33
5.3 Geolocation for current state before cleaning 34
5.4 Graph distribution for professor rating before cleaning 36
5.5 Graph distribution for professor rating before cleaning 37
5.6 Graph for distinct count of current state before and after cleaning 38

CHAPTER 1: INTRODUCTION

Data are symbolic representation of information which are depicted by symbolic values. Data in the real world is dirty. If it is not clean, then data analysis will not deliver reliable results. Data is high quality when it is up to date; data currency which means the data in a database must have the latest values with respect to entity. As data obtained from various sources is collected there would be some changes in the field like last name or salary. So the old values should be updated to maintain currency; complete – it is the extent by which the data meets the expectations; accurate – accuracy of data is determined as an aggregated value over quality criteria integrity, consistency –any given database transaction must change affected data only in allowed ways; easy to use – the availability of the data. Data should not be hard to access, but be readily available to everyone that requires it. Data Cleaning can be addressed as one of the biggest challenge in any organization.

1.1 Data Cleaning

Data cleaning, also called data cleansing is the process of detecting, correcting, or removing errors and other inconsistencies from data to improve the data quality. These problems are usually present in individual data sources. Data cleaning plays a significant role when there is a need to integrate multiple data sources and analyze it. This is because multiple sources contain some redundant or incorrect data in different representations. Hence, duplicate elimination of data has become an important task in order to make data accurate and consistent.

1.2 Problems in Data Cleaning

Data cleaning is considered as one of the biggest problems in data warehousing because of the fact that there is a wide range of possible data inconsistencies. Data warehouses are often used for decision making; thus the correctness of their data is important in order to avoid wrong conclusions. For instance, duplicated or missing information will produce incorrect or misleading statistics.

There are several problems to be considered for a data cleaning i.e; it should satisfy several requirements. Firstly, it should detect, remove all major errors and other inconsistencies in individual data sources and also when integrating multiple sources [1].

1.3 Problems with existing approaches

Data Quality Problems

The following are the major data quality problems to be solved by data cleaning. These are classified into two categories i.e; Single-Source Problems and Multiple-Source Problems [1].

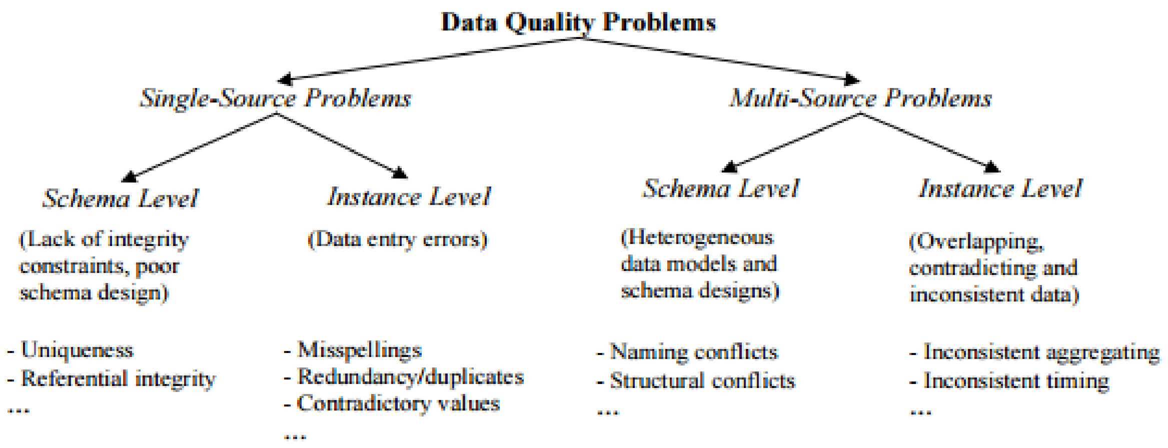


Figure 1.1 Classification of data quality problems in data sources

Single-Source Problems:

The Schema related problems occur due to lack of integrity constraints and poor schema design. Instance specific problems occur due to data entry errors and inconsistencies. Misspellings, Redundancy/duplicates, Contradictory values etc. are some examples. A proper design of the database schema is needed in order to reduce such problems and also integrity constraints should be defined appropriately.

Multiple-Source Problems:

Schema related problems occur due to heterogeneous data models and schema designs like naming conflicts, structural conflicts etc. “Naming conflicts occur when the same name is used for different objects (homonyms) or different names are used for the same object (synonyms)” [1]. Structural conflicts refer to different representations of the same object in different sources, e.g., attribute vs. table representation, different data types, different integrity constraints, etc. Instance specific problems occur due to overlapping, contradicting and inconsistent data like inconsistent aggregating, inconsistent timing etc.

Data cleaning in unstructured data:

It has been known that structured data constitutes to about 20% whereas unstructured data makes up about 80% of the data. Structured data is the one which resides in a fixed field with a record or file in relational databases or spread sheets. Structured data is classified depending on the data model on the type of data and how it is stored, processed and accessed. Structured data is always advantageous in regards to storage, analysis, information retrieval which makes for effective management of data.

Unstructured data is data which cannot be stored in the standard form of relational database or schema/table format, where we have specific columns with metadata and formats associated with the data types stored in the database. Unstructured data often include text and multimedia content. Examples of unstructured data content is email-messages, word processing documents, videos, photos, audio files, presentations, web pages and many other kinds of business documents. Unstructured data is hard to analyze for any computer program. In addition to this there is semi-structured data which doesn’t reside in relational database but have some organizational structure internally which makes it a bit easier to analyses.

1.4 Proposed Approach

We propose a unified approach to deal with data violation constraints, spelling corrections and uncertainty in data using a rule based strategy for unstructured data in graph databases. This process gives us a database which will be clean. First, we apply different data cleaning algorithms on our unstructured Neo4j data for integrity constraints. Then we perform spelling correction on proper nouns using edit distance methods and also using ontology.

Furthermore, after performing data cleaning, we use tools for Data Visualization to show the impact of cleaning. The data is visualized before and after applying the different algorithms. Thus we compare both the results using Tableau. In our approach, we run each step separately i.e; the steps that are proposed in section 4.2 and check the effectiveness of each algorithm in the process of data cleaning.

1.5 Thesis and Research Objectives

Cleaning in unstructured graph databases has not been studied before. Data cleanliness is very important. The main objective of this thesis is to produce effective data cleaning algorithms which provides a clean graph database. Furthermore, we also validate cleaning by data visualization techniques. The research objectives of this thesis are:

- Identification of noisy data and other inconsistencies in unstructured graph databases.
- Correction of data using different methods or algorithms.
- Improving data quality with more accuracy, consistency and completeness.
- Use of ontology based approach for spelling correction along with edit distance method.
- Visualization of data after cleaning and comparison before and after cleaning.

1.6 Outline of Thesis

The rest of the thesis documentation is organized as follows: Chapter 2 presents the literature review which includes details about previous work related to data cleaning methods and graph databases. Chapter 3 contains the details about the graph database used; Neo4j. Chapter 4 includes the description of the proposed approach. Chapter 5 describes the details of data visualization tools used and also the simulation results using visualization tools which represent the efficiency of data cleaning algorithms. Finally, Chapter 6 concludes the thesis and presents ideas for future work.

CHAPTER 2: REVIEW OF LITERATURE

Any data cleaning approach must have automated tool support in order to limit manual intervention and programming effort and also it should be extensible in order to cover other additional sources.

2.1 Data cleaning approaches:

Data cleaning generally involves several phases. Some of them are as follows:

Data Analysis [1]:

A detailed data analysis is required to detect errors and inconsistencies that are to be removed. There are two approaches for data analysis namely, data profiling and data mining. Data profiling focusses mainly on the instance analysis of individual attributes. It derives information such as the data type, length, value range, discrete values and their frequency etc. Data mining helps us to discover specific data patterns in large data sets, e.g., relationships holding between several attributes.

Defining data transformations [1]:

Data transformation and cleaning steps depends on the number of data sources, degree of heterogeneity and "dirtyness" of the data. This process consists of various steps where each step may perform schema and instance-related transformations (mappings). Data transformation implies conversions of data in source system to the respective data types of corresponding domains. These are done by using SQL and other ETL tools.

Validation and correction [1]:

This step corrects each source instance for data entry errors. Misspellings can be done by dictionary lookup. Similarly, address data can be corrected by dictionaries on geographic names and zip codes. Also, with the help of Attribute dependencies (e.g., birthdate – age), the missing values and other incorrect values can be corrected.

Backflow of cleaned data [1]:

After cleaning is performed, the cleaned data should also replace the dirty data in the original sources. This cleaner data avoids redoing the cleaning work for future data extractions.

2.2 Data Cleaning Methods:

The data cleaning process requires some data cleaning methods to improve the quality of data. Data cleaning methods that needs to be applied are selected after anomalies are detected in database. Some methods are employed for detection and some are used for correcting the detected anomalies. Some of the existing methods are:

Statistical Methods [2]: These methods are used for tracking or correcting anomalies. We analyze the data to identify the type of anomaly e.g., Insertion, Deletion or Update anomaly. Statistical measures (mean, standard deviation etc.) are used to find invalid values in the database. If a satisfactory value is not found, an “average value” is set to replace invalid values. This method is also used to handle missing values.

Clustering [2]: This method is used to eliminate noise data. After clustering is applied, pattern matching is done. The chances of identifying a pattern is increased as the cluster already has data sets of similar type.

Pattern Based Cleaning [2]: This method is used to eliminate syntax errors. Parsing is used to identify syntactical anomalies. The occurrence of syntactical errors depends on the file type where the data is present.

Integrity Constraints [2]: Defining integrity constraints while designing a database ensures correctness in all transactions performing data collection. We correct the database by studying these integrity constraints and correct the values that violate these constraints.

2.2.1 Duplicate Elimination [2]:

Improper merging of data results in duplicates. Detecting duplicate tuples and correcting them is a big challenge in data cleaning. Duplicates give incorrect answers for a query. We use different algorithms to detect and remove such values. Some of the techniques for duplicate record detection are as follows:

Edit Distance [5]: This is the minimum number of edit operations of single characters to transform a string 1 into string 2. There are three types of edit operations namely **insert** a character into the string, **delete** a character from the string, and **replace** one character with a different character.

In any simplest scheme, the cost of each edit operation is 1. This method works well for correcting typographical errors, but ineffective for other types of mismatches.

Affine Gap Distance [5]: when strings that have been truncated or shortened (e.g., “John R. Smith” versus “Jonathan Richard Smith”) are matched, then the edit distance method does not work well. In such cases we use the affine gap distance technique. It has two extra edit operations: open gap which is the cost required to open a gap of any length and extend gap which is the cost to extend the length of an existing gap by 1. The cost of extending the gap is smaller compared to the cost of opening a gap. Thus, affine gap penalties have smaller costs compared to edit distance metric.

Smith-Waterman Distance [5]: In this method, the mismatches in the middle of the strings are compared instead of mismatches at the beginning and the end. This metric allows for better local alignment of the strings (i.e., substring matching). For example, the strings “Prof. John R. Smith, University of Calgary” and “John R. Smith, Prof.” can match within a short distance since the prefixes and suffixes are ignored.

Rule-Based Approaches [5]: In this method, rules define whether two records are the same or not. The distance of two records is either 0 or 1. When there is no primary key, use the rules developed by data experts to derive a set of attributes to serve as a “key” for each record. <u>Example:</u> an expert defines rules such as

```
IF age < 22
THEN status = undergraduate
ELSE status = graduate;

IF distanceFromHome > 10
THEN transportation = car
ELSE transportation = bicycle;
```

Through such rules, we can generate unique keys that can cluster multiple records that represent the same real-world entity.

Levenshtein's distance algorithm [3]:

In this algorithm, the first string is referred to as the source string (s) and the second string is referred to as the destination (d). Levenshtein's distance is the number of substitutions, deletions or insertions required to transform the source string to a destination string.

If the source string is 'googl' and the destination string is 'google', then the Levenshtein's distance LD(s,d)=1 because one insertion is required to transform the source into the destination string. It source string is 'good' and destination string is 'good', then Levenshtein's distance LD(s,d)=0, since both strings are identical.

The greater the Levenshtein distance, the more different the strings are.

The Algorithm

Step 1:

Set 'n' = Length of first string 's'

Set 'm'=Length of second string 'd'

If n=0 return m and exit.

If m=0 return n and exit.

Construct a matrix 'C' containing 0..m rows and 0..n columns.

Step 2:

Initialize the first row to 0..n.

Initialize the second row to 0..m.

Step 3:

Scan every character of first string ‘s’ using loop variable ‘i’ from 1 to n.

Step 4:

Scan every character of second string ‘d’ using loop variable ‘j’ from 1 to m.

Step 5:

If s[i] is equivalent to d[j], the cost required is 0.

If s[i] is not equivalent to d[j], the cost is 1.

Step 6:

Set each cell of matrix ‘C’ i.e. d[i,j] equal to the minimum of :

i) The cell immediately above + 1 : C[i-1,j]+1

ii) The cell immediately to the left + 1: C[i,j-1] + 1

iii) The cell diagnostically above and to the left + the cost: C[i-1,j-1]+cost.

Step 7:

Once the iterations of steps 3,4,5,6 are done, the distance between two strings is present in the cell C[n,m].

Below is an illustration of the above algorithm:

String s = ‘gumbo’ , String d = ‘gambol’.

We will calculate the cost required to change the string s into string d using above algorithm.

Step 1: n=5, m=6.

		g	u	m	b	O
	0	1	2	3	4	5
g	1					
a	2					
m	3					
b	4					
o	5					
l	6					

Steps 3 to 6:

When i = 1, the matrix C is as below:

		g	u	m	b	O
	0	1	2	3	4	5
g	1	0				
a	2	1				
m	3	2				
b	4	3				
o	5	4				
l	6	5				

When i=2, the matrix C is as below:

		g	u	m	b	O
	0	1	2	3	4	5
g	1	0	1			
a	2	1	1			
m	3	2	2			
b	4	3	3			
o	5	4	4			
l	6	5	5			

When i=3, the matrix C is as below:

		g	u	m	b	O
	0	1	2	3	4	5
g	1	0	1	2		
a	2	1	1	2		
m	3	2	2	1		
b	4	3	3	2		
o	5	4	4	3		
l	6	5	5	4		

When i=4, the matrix C is as below:

		g	u	m	b	O
	0	1	2	3	4	5
g	1	0	1	2	3	
a	2	1	1	2	3	
m	3	2	2	1	2	
b	4	3	3	2	1	
o	5	4	4	3	2	
l	6	5	5	4	3	

When i=5, the matrix C is as below:

		g	u	m	b	O
	0	1	2	3	4	5
g	1	0	1	2	3	4
a	2	1	1	2	3	4
m	3	2	2	1	2	3
b	4	3	3	2	1	2
o	5	4	4	3	2	1
l	6	5	5	4	3	2

Step 7: The cost/distance between the strings ‘gumbo’ and ‘gambol’ is 2. We require one substitution of ‘u’ to ‘a’ and one insertion of ‘l’ at the end. Hence 2 changes are required.

2.3 Structured and Unstructured Datasets

Structured Dataset:

Data is stored in the form of rows and columns. It conforms to a data model. Attributes in group/table are the same. Some of the sources of Structured Datasets are Relational Databases, Spreadsheets. The limitations of a structured dataset are limited storage and it contains homogenous data only.

Unstructured Dataset [6]:

Data which does not follow any particular format or sequence is termed as unstructured data. It does not confirm to any data model. Most datasets cannot be stored in any standard format of relational database or schema/table format. It is not easily usable by a program.

Unstructured data can be classified into three different types:

1) <u>Data not in proper format:</u> A proper format refers to the correct format specified based on business requirements. Here the data can be maintained in a spreadsheet but cannot be used for any further analysis because of the improper format of the content of the data. Each type of data may have multiple standard formats.

    Example:

    Standard Format for Address as per business requirement –Street, City, State, and Country.

    Address (Format 1): 127 N Duck Street, Stillwater, Oklahoma, USA. (Proper format)

Address (Format 2): 127 North Duck St, Oklahoma, USA. (Improper format - City name missing)

2) Text Based Documents: Here the data is maintained in the form of word documents, presentations, emails, blogs and audio/video files.

3) Different Attributes/Properties for every item: Here item refers to nodes in a graph or an entry/tuple/record in a table. The attributes or properties differ from one tuple to another and thus cannot be maintained in a standard relational database table with fixed columns. Storage of these datasets in relational database tables leads to complexity of Join operations. Hence most of these datasets prefer graph databases to avoid table constraints on the attributes/properties of the tuples or nodes. These datasets on exporting to spreadsheets results in improper format and thus makes it difficult to visualize using standard visualization tools.

2.4 Graph Databases [4]

A graph stores data in the form of nodes and relationships. A graph represents data in a highly accessible way. A graph represents entities as nodes and the edges as relationships. The graph database is a type of NoSQL database that uses graph theory to store, map and query relationships. The nodes and relationship in a graph can have properties. This is called a Property Graph Model.

Nodes: Nodes are used to represent entities. Nodes can have properties. Nodes are connected through relationships.

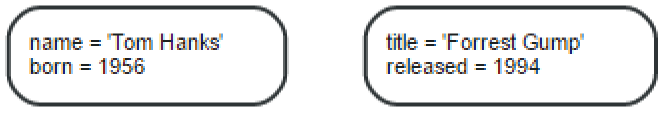


Figure 2.1 Nodes

Relationships:

Each relationship represents an edge. Edges are the lines that connect nodes to nodes or nodes to properties. Relationships between nodes allow us to find related data and enhance graph traversal. Just like nodes, relationships can also have properties.

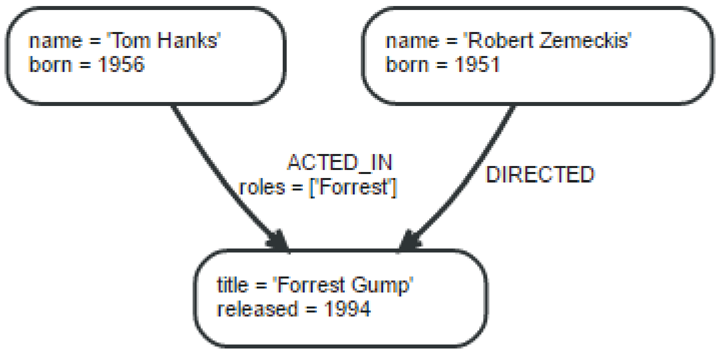


Figure 2.2 Nodes connected through Relationships

A node can have relationships to itself.

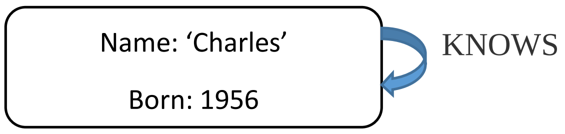


Figure 2.3 Node having relationship to itself

Properties:

Properties are key-value pairs used to add information about nodes and relationships. The values of a property can be numeric values, string values, boolean values etc. NULL is not a valid property value. Instead of storing NULL in the database, NULL can be modeled by the absence of a key. In the above figure, node has the properties name, born. Properties can also be termed as attributes.

E.g: An actor might have a name property and another property date of birth.

(a {name: “Reeves “, born: 1964}) Actor with name and born property.

Labels:

A label is a named graph construct that is used to group nodes into sets; all nodes labeled with the same label belongs to the same set. Many database queries can work with these sets instead of the whole graph, making queries easier to write and more efficient to execute. Any node can have one or more labels. We use labels to distinguish between different types of nodes.

Traversal:

A traversal is how you query a graph, navigating from starting nodes to related nodes. Traversing a graph means visiting its nodes and following relationships according to some rules.

Paths:

A path is one or more nodes with connecting relationships, typically retrieved as a query or traversal result. The shortest possible path has length zero — that is, it contains only a single node and no relationships.

Example of a property graph is as follows [4]:

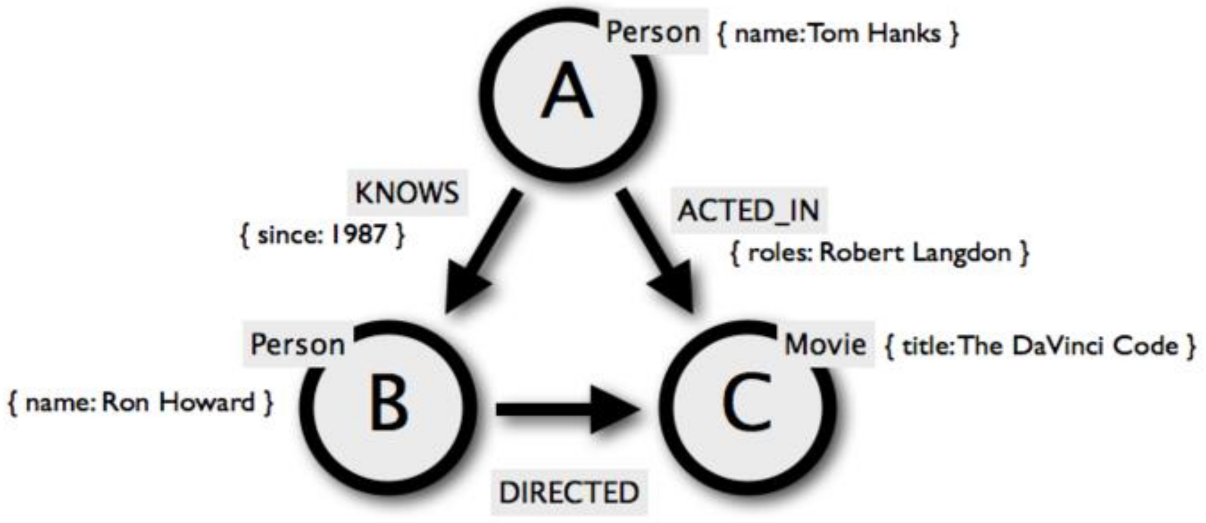


Figure 2.4 Property Graph

Attributes:

An attribute refers to a database field. Attributes describe the instances in the row of a database.

In graph databases, these attributes are termed as properties.

CHAPTER 3 – NEO4J

3.1 Neo4j

Neo4j is a graph database tool whose data model is a property Graph. Neo4j is designed for fast management, storage, and traversal of nodes and relationships. The performance of join operation in the relational database degrades exponentially, but in Neo4j the performance is linear since the navigation is from one node to another.

The advantages of Neo4j are as follows:

- Highly scalable (several billion nodes on a single machine),
- User friendly API,
- Supports efficient graph traversals.

Some of the other benefits of Neo4j are:

Whiteboard friendly [4]: It allows consistent use of the same model throughout the design, implementation, storage, and visualization of any domain. This allows all business stakeholders to participate throughout the development cycle.

Rapid development [4]: Neo4j features supports fast development of highly scalable applications.

Data safety [4]: Neo4j makes sure that data is not lost in case of power failures or crashes with the help of ACID transactions.

3. 2 Cypher Query Language

Cypher is Neo4j’s graph query language. Complex database queries can be expressed through Cypher. Like most query languages, Cypher is composed of clauses. The simplest queries consist of a START clause followed by a MATCH and a RETURN clause. Some of the Cypher clauses are:

- START: specifies one or more starting point’s nodes or relationships.
- MATCH: graph pattern to match.
- WHERE: Provides criteria for filtering pattern matching results.
- RETURN: What to return.
- CREATE and CREATE UNIQUE - Create nodes and relationships.
- DELETE - Removes nodes, relationships, and properties.
- SET - Sets property values.
- FOREACH - updating action for each element in a list.
- UNION - Merges results from two or more queries.

CHAPTER 4: PROPOSED WORK

4.1 Overview

The main objective of this work is to design and implement a step by step approach for data cleaning of unstructured data in Graph Databases. In this chapter we identify the algorithms to be considered for correction of spellings using ontology and visualization. Spelling errors can happen from typographical errors that accidently produce a real word e.g; there for three. Spelling errors can also be any word not found in a dictionary. Misspellings leads to incorrect data analysis. Thus, correction of spellings is an important task in data cleaning to improve data quality. Changes to the data quality of a data cleaning algorithm can be visualized to see if data cleaning has an impact on the distribution of data using visualization by comparing the results before and after cleaning. The validation of other cleaning methods like duplicate elimination, data repair using integrity constraints and calculation of missing values that are considered in another thesis project – “Data cleaning on graph databases using Neo4j: duplicate elimination, data repair and correction of missing values” [8] can be done with the help of data visualization techniques.

4.2 Design and steps of Data Cleaning Model

The proposed steps for data cleaning over Graph databases are as below:

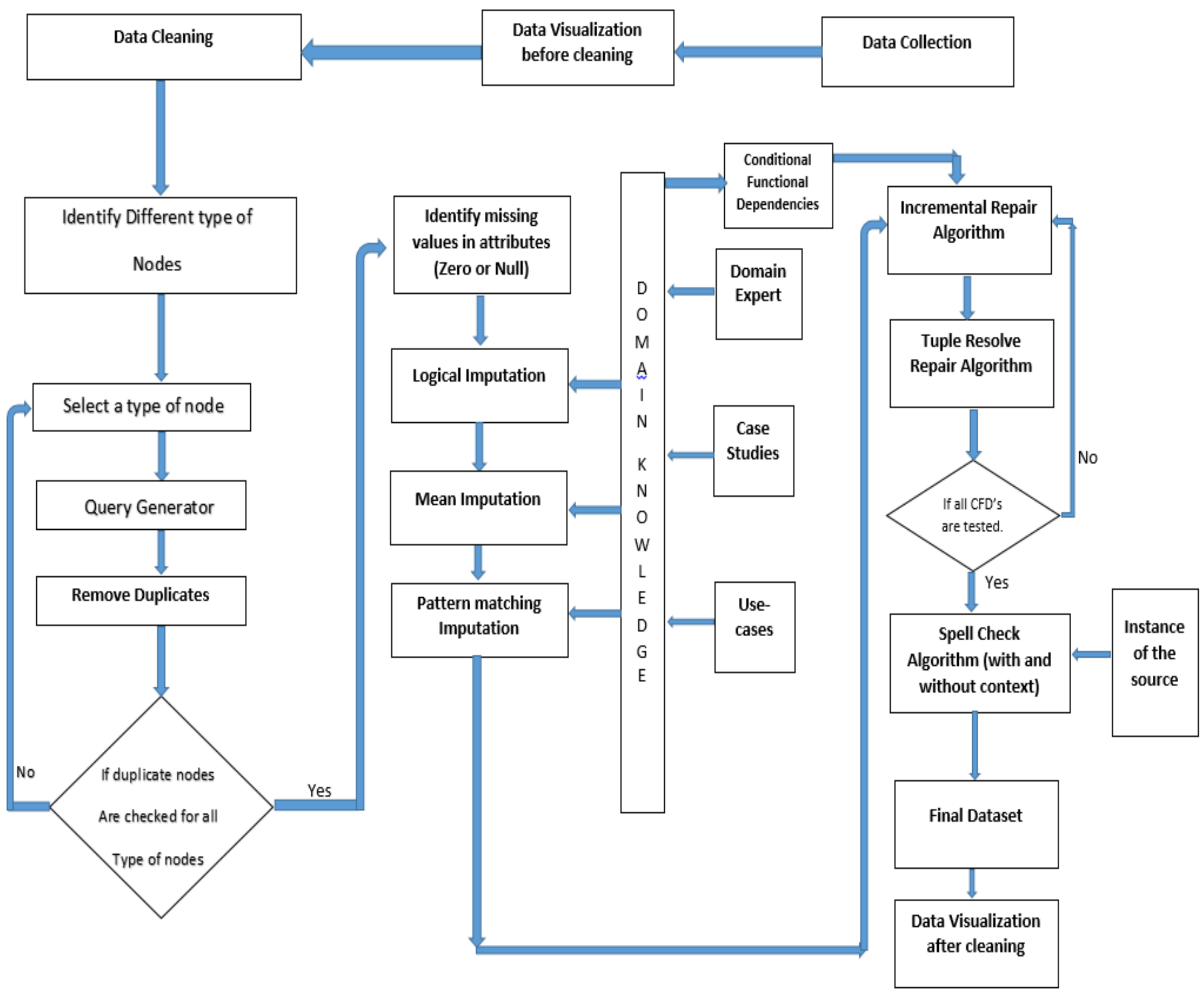
Step 1) Spelling corrections

a) Proper nouns (here cities and states).

b) Using ontology.

Step 2) Visualization of data before and after cleaning to validate the cleaning.

The initial steps like data duplicate elimination, data repair using conditional dependencies and correction of missing values are performed in another thesis project – “Data cleaning on graph databases using Neo4j: duplicate elimination, data repair and correction of missing values” [8]. Hence, this proposed work is an extension to that work. The output from that work will act as an input source for cleaning steps presented in this work and all the results will be visualized to validate the cleaning algorithms. A detailed architecture of the proposed system is shown below:



Data Cleaning
Data Visualization before cleaning
Data Collection
Identify Different type of Nodes
Select a type of node
Query Generator
Remove Duplicates
If duplicate nodes Are checked for all Type of nodes
No
Yes
Identify missing values in attributes (Zero or Null)
Logical Imputation
Mean Imputation
Pattern matching Imputation
DOMAIN KNOWLEDGE
Conditional Functional Dependencies
Domain Expert
Case Studies
Use-cases
Incremental Repair Algorithm
Tuple Resolve Repair Algorithm
If all CFD's are tested.
No
Yes
Spell Check Algorithm (with and without context)
Instance of the source
Final Dataset
Data Visualization after cleaning

Figure 4.1: Detailed Architecture of Data cleaning

4.3 Implementation

The tools used for implementation of these cleaning techniques are Eclipse, Neo4j and Tableau. The programming languages used for implementation are Java and CQL. Java is mainly used to run queries on the graph database by interacting with Neo4j. The Jar file neo4j 2.3.5.jar is used to interact Java with Neo4j. The Class GraphDatabaseService is used to setup up the connection between Neo4j and Java. The ExecutionResult and ExecutionEngine classes in the jar file are used to run CQL queries to interact with the dataset. The information to use variables or attributes in a CQL query is given by domain knowledge. The domain knowledge is given by domain experts in a regular text document format. This information is converted into a CQL query and executed on neo4j using Java.

4.4 Algorithms and approaches

Each and every step is designed separately using a different algorithm and is executed on the datasets in different orders to find the optimum order. For step 1a and step 1b, the algorithm that is used to correct the words is the Levenshtein’s distance algorithm.

Step 1a) Spelling corrections of proper nouns:

In this step, we correct spellings of proper nouns (here cities and states). Incorrect spellings generally occur due to typos. This leads to incorrect analysis when data is used for visualization. We apply Levenshtein's Algorithm to correct spellings of cities and states names in this dataset. A source is a collection of correct spelling words which are grouped based on categories. An instance of the source which contains the correct spellings of cities, states and countries name is used for correction of cities and states names in this dataset. The importance of this step will be shown in visualization of dataset after implementing all the cleaning steps on the dataset.

We define a syntax for each CQL query in Java code which includes the clauses to be used. This code will also determine the type of node, attributes/property values to select from the source depending on the types of attribute/property.

For spelling correction the CQL query clauses will be MATCH, WHERE, RETURN, UPDATE, SET. Thus, we define a syntax for CQL query in the java.

E.g; MATCH(S:type of node) WHERE has (S.property of node), RETURN S;

```
MATCH (S: type of node) WHERE S.primary key =“value”
SET S.Property of node = local_variable.
```

Example:

For a student, there exists two attributes/properties titled- Current Location and Current State.

Entry:

SID: 10, Full Name: John Charles, First name: John, Last name: Charles, Age: 25, Date of Birth: 19-05-1991, **CurrentState: Oklahom, CurrentLocation: stillwate.**

Here, the current state and location are misspelt. So, the algorithm corrects these values with the help of Levenshtein's Algorithm.

Pseudo code for execution:

- Step 1: The CQL query to retrieve students with property gender are retrieved into Java

  The Java code reads the list and interprets the CQL query as below:

    - ▶ The first column in the list indicates the node type.
    - ▶ The second column in the list indicates the where attribute.
    - ▶ The third column in the list indicates the source list of values.
    - ▶ The fourth column indicates the operation.
    - ▶ The fifth column indicates the number of characters to be check.

The values in these columns are assigned to string variables in Java and the CQL query is generated using string concatenation methods in Java. The CQL query generated will be

- ▶ MATCH(S:type of node) WHERE has (S.property of node), RETURN S;
- ▶ Update where conditions is S. property of node =local_variable.

- Step 2: Start iteration for all the node instance returns in above query.
- Step 3: The next step is the check for the value of attribute for the type of node.
- Step 4: We check the value of attribute in the source

```
    If exists then no update.
    Else
        MATCH (S: type of node) WHERE S.primary key =“value”
        SET S.Property of node = local_variable.
```

- Step 5: If all node instances are not checked go to step 2.

  Else go to step 6

- Step 6: exit

Step 1b) Spelling corrections using ontology (context):

Context is any information that can be used to characterize the situation of an entity. An ontology provides a vocabulary that describes a domain of interest and a specification of the meaning terms used in the vocabulary. In this thesis work, we use context ontology.

In certain cases, spelling corrections leads to ambiguity. The spelling might be correct but it will be in-correct as per the context. The values in an attribute/property are compared with a list of acceptable values that are given by a domain expert. After performing ontology matching, we check whether the word matches with any of those acceptable values and update the word accordingly.

Without the ontology we cannot automate the process and put the words in context. In this work, we use domain ontology. Particular meanings of terms applied to a domain are provided by domain ontology. Here, we have used the education domain. This domain ontology is applied on node of the type professor with the property professor rating. The values in this attribute/property are compared with a list of acceptable values that are given by a domain expert. We check whether the word matches with any of those acceptable values and update the word accordingly.

The below figure shows the ontology that are used for our dataset.

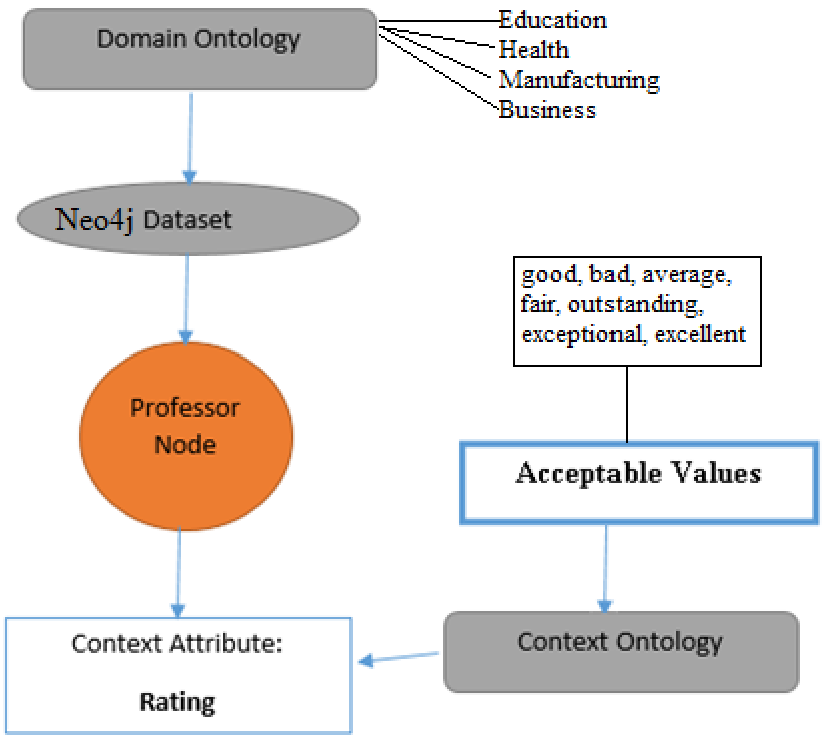


Figure 4.2: Ontology

Steps for cleaning data using ontology:

- Identify the context attributes based on the domain knowledge.
- Check for the spellings of attribute values using Levenshtein's Algorithm.
- If the attribute value doesn't fit to the context after comparing with the acceptable values using context ontology, then correct the word using Levenshtein's Algorithm.

<u>Example:</u> Considering the attribute - Overall Rating for a professor.

The string under Overall Rating is misspelt and is shown as **far**. However this is incorrect. The word 'Far' is a correct word but incorrect as per the context and should be corrected to 'Fair'. However when we calculate the Levenshtein's distance, it turns out to be one for 'far' and 'fair'.

In this scenario, we will define positive, negative and neutral words for ratings and will be cleaning the words close to the words present in the dataset based on the context of the word. Hence, the word ‘Far’ will be corrected to ‘Fair’. In case of date of birth, if the month is present in words, we check for the spelling and correct it if necessary and change into the required format.

Observations on spell check algorithms:

Levenshtein’s algorithm is linked with context to improve efficiency in cleaning. The spelling corrections are effectively done using the source instance and Levenshtein's algorithm. For the current dataset, we observed that this data cleaning technique achieved an accuracy up to 99.99% as the source is selected based on domain knowledge. The accuracy may differs in case of larger datasets since complete cleaning cannot be achieved.

Step 2) Visualization of data before and after cleaning to measure the impact of cleaning:

This step mainly focusses on validation of cleaned data. This step helps us to show the effectiveness of the proposed algorithms in the process of cleaning. This step is explained in detail in chapter 5.

CHAPTER 5 – DATA VISUALIZATION & SIMULATION RESULTS

5.1 Data Visualization & Tools Used:

The main goal of using data visualization is to visualize data, communicate information clearly and effectively and to show the effectiveness of proposed algorithms in the process of cleaning.

The leading benefit of data visualization is that it not only provides graphical representation of data but also allows changing the form, removing what is not required, and browsing deeper to get further details. This provides a great advantage over traditional methods. With the help of it data can be viewed in multiple ways effectively by dividing data findings. It provides an additional sense to the data by making patterns.

Tableau:

One can see and understand data, reports and dashboards faster through unique, easy-to-use visual analytics technology using tableau [7]. Tableau makes it easy to share web-based dashboards, reports and graphics with a few clicks. Using tableau, there is no need for us to create thousands of reports to represent every possible slice of the data; we can just create the base templates, and then customize it while we are using the product.

Tableau was the ideal choice for these reasons; it's powerful, highly interactive, and easy-to-integrate into our existing infrastructure and makes it much easier to identify actionable insights and share them throughout the company.

5.2 Results:

Implementing the defined methods and cleaning algorithms were done using the Java programming language and the results were visualized using Tableau. Applying the cleaning steps we clean the dirty data and create visuals of clean data that helps to draw meaningful conclusions about data. In this thesis, we simulate the defined methods to clean data and show how data cleaning process impacts on analysis of data. To do this we compare visuals of clean data and visuals of unclean data.

Dataset:

In this study, the dataset we used is a university database. The nodes in this dataset are Student, Faculty and Courses. The relationships in this dataset are Enrolled and TakenBy. Each node has one or more attributes. One of the attribute is the primary key. This data is corrupted with improper values and spelling corrections.

Graphs:

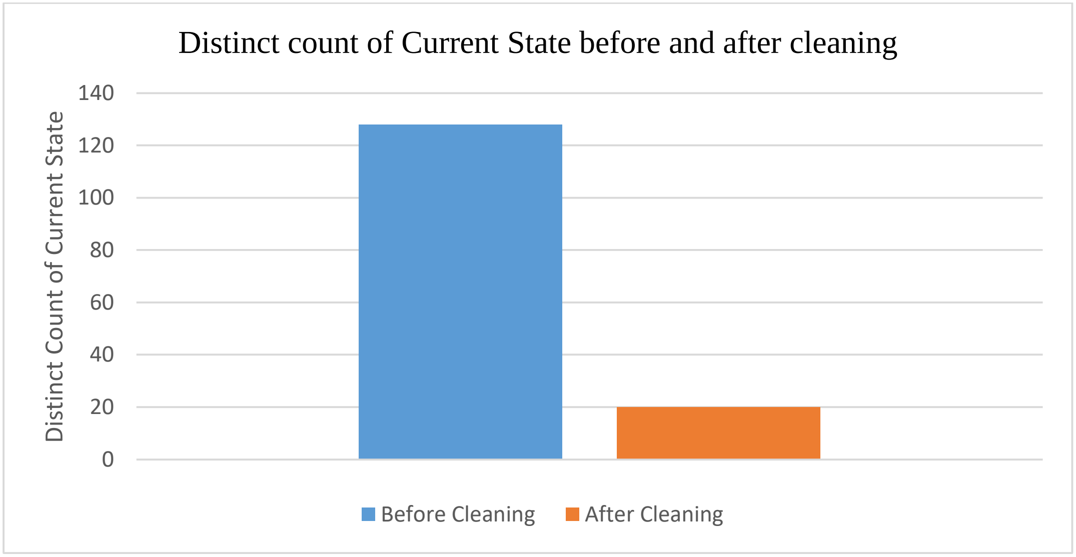


Figure 5.1 Graph for Distinct count of Current State before and after cleaning

This graph shows that the distinct count of current state value is much greater before cleaning since it contains incorrect values. There is a greater reduction in the count value after cleaning is applied. The values of current state can also be represented on a geolocation map as follows:

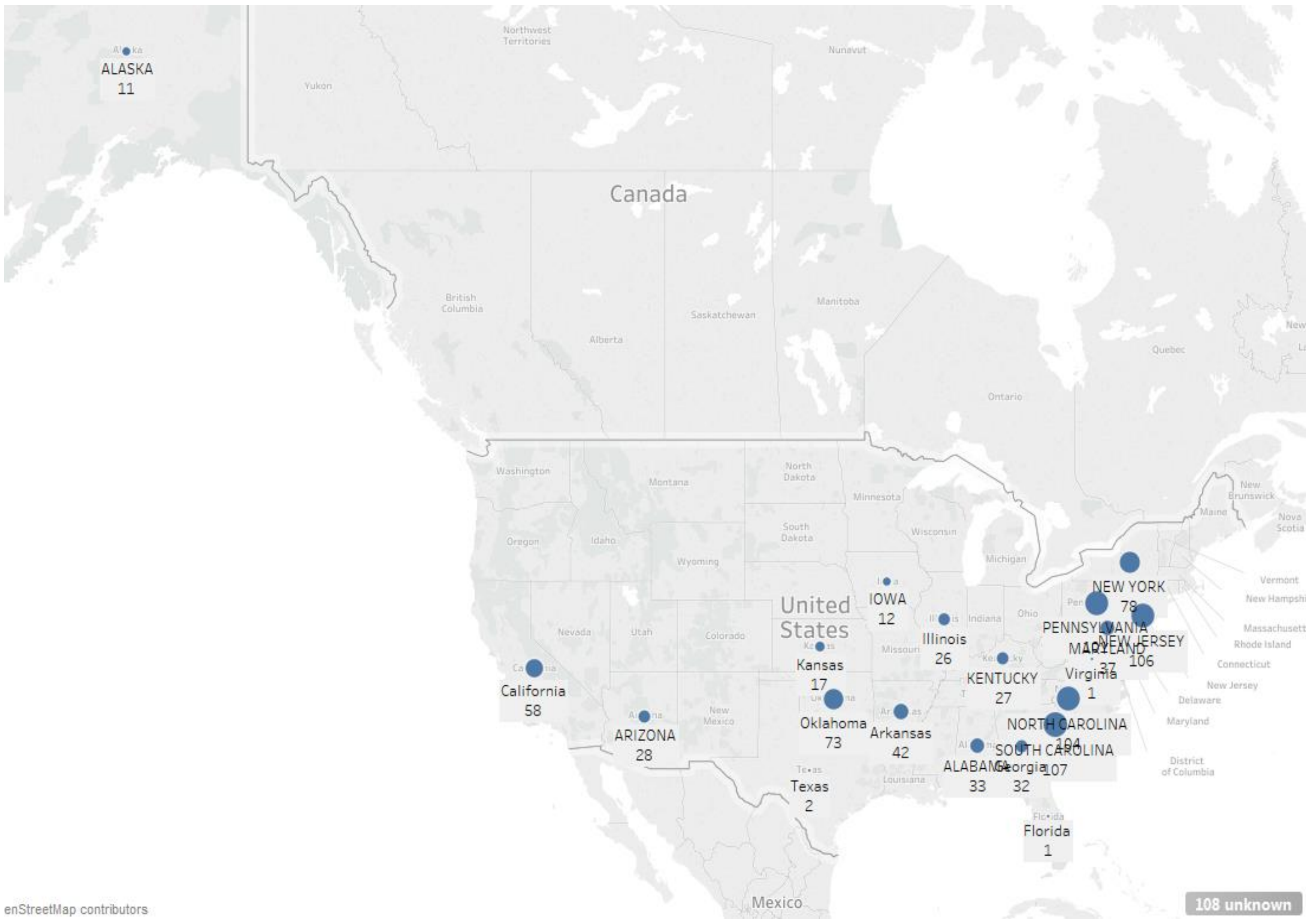


Figure 5.2 Geolocation for Current State before cleaning

In Figure 5.2, the current state values are represented on a map. From the map, we can see that there are 108 unknown values which are incorrect values that can't be represented on the geolocation map. This suggests that the data contains some incorrect values for the attributes current state.

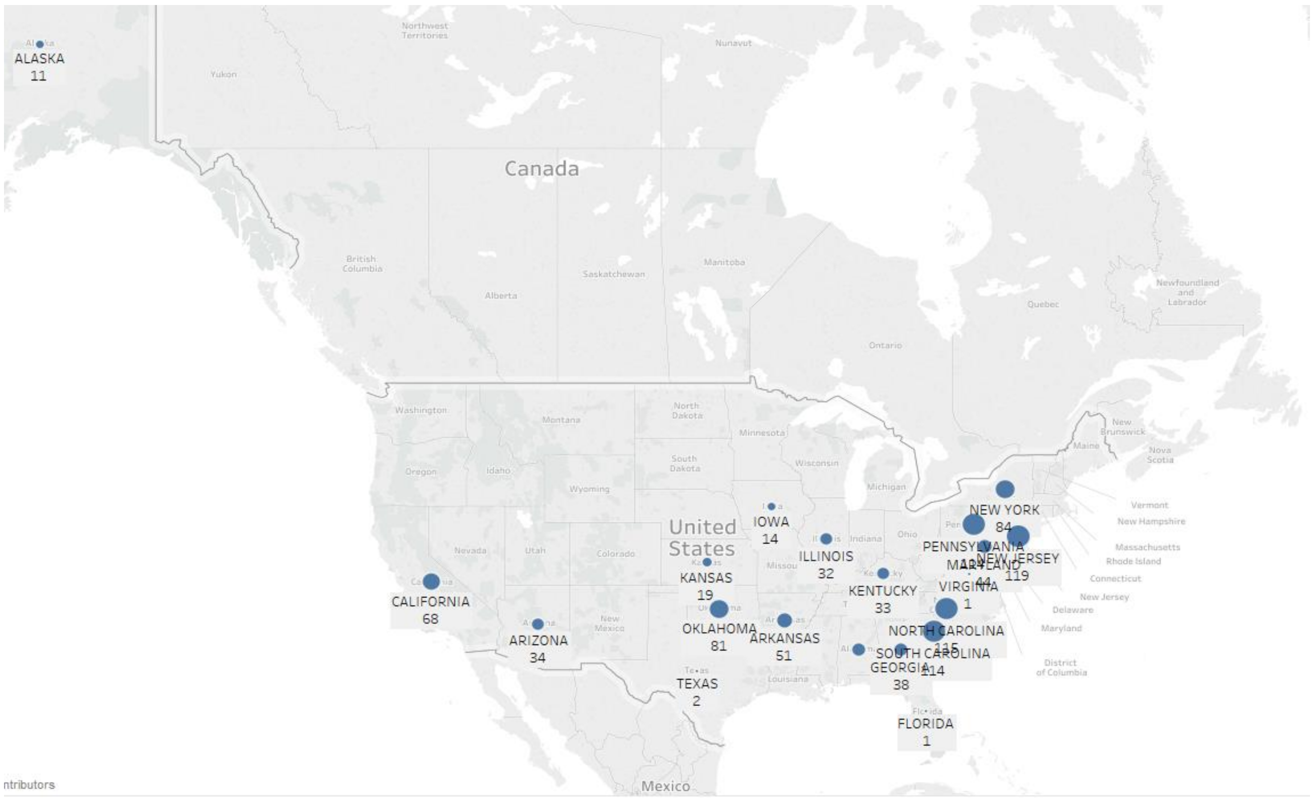
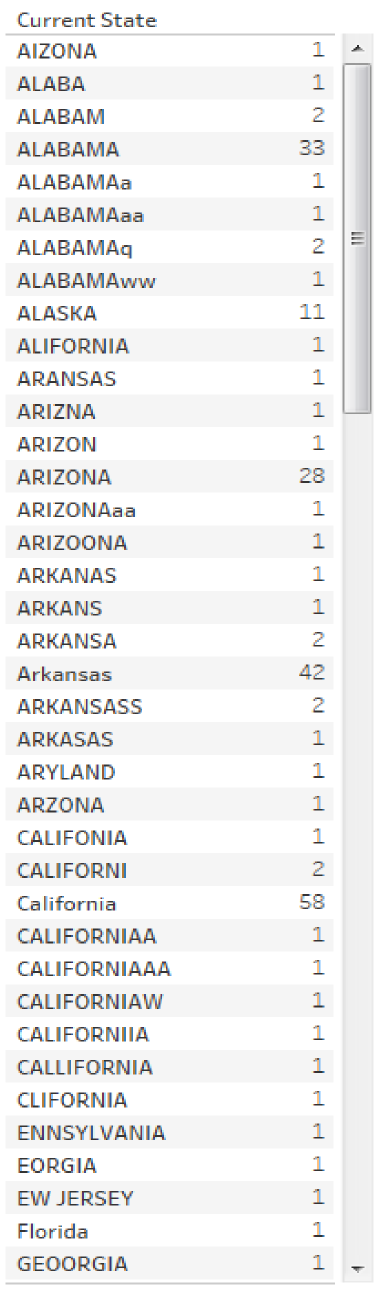


Figure 5.3 Geolocation for Current State after cleaning

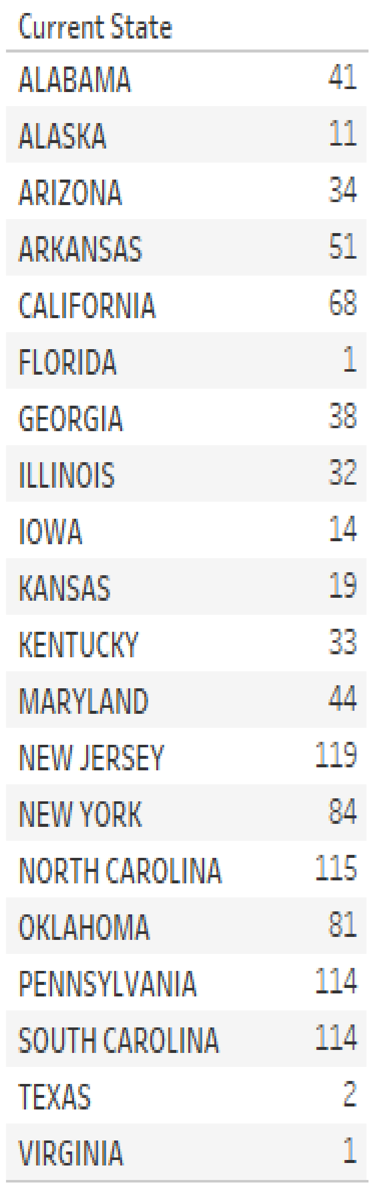
In this figure, the incorrect or misspelt values of the attribute correct state have been corrected using the data cleaning algorithm. The change in the values of count of each state can be observed on the map. This implies that the accuracy of data is almost 99.99%, which can be observed from the map and also from the values in figure 5.1

The values of the current state attribute before cleaning and after cleaning can also be observed in text format as shown below:

Current State



Current State	
AIZONA	1
ALABA	1
ALABAM	2
ALABAMA	33
ALABAMAa	1
ALABAMAaa	1
ALABAMAq	2
ALABAMAww	1
ALASKA	11
ALIFORNIA	1
ARANSAS	1
ARIZNA	1
ARIZON	1
ARIZONA	28
ARIZONAaa	1
ARIZOONA	1
ARKANAS	1
ARKANS	1
ARKANSA	2
Arkansas	42
ARKANSASS	2
ARKASAS	1
ARYLAND	1
ARZONA	1
CALIFONIA	1
CALIFORNI	2
California	58
CALIFORNIAA	1
CALIFORNIAAA	1
CALIFORNIAW	1
CALIFORNIIA	1
CALLIFORNIA	1
CLIFORNIA	1
ENNSYLVANIA	1
EORGIA	1
EW JERSEY	1
Florida	1
GEOORGIA	1



Current State	
ALABAMA	41
ALASKA	11
ARIZONA	34
ARKANSAS	51
CALIFORNIA	68
FLORIDA	1
GEORGIA	38
ILLINOIS	32
IOWA	14
KANSAS	19
KENTUCKY	33
MARYLAND	44
NEW JERSEY	119
NEW YORK	84
NORTH CAROLINA	115
OKLAHOMA	81
PENNSYLVANIA	114
SOUTH CAROLINA	114
TEXAS	2
VIRGINIA	1

Before Cleaning

After Cleaning

The graphs for spelling correction using context based ontology is shown below:

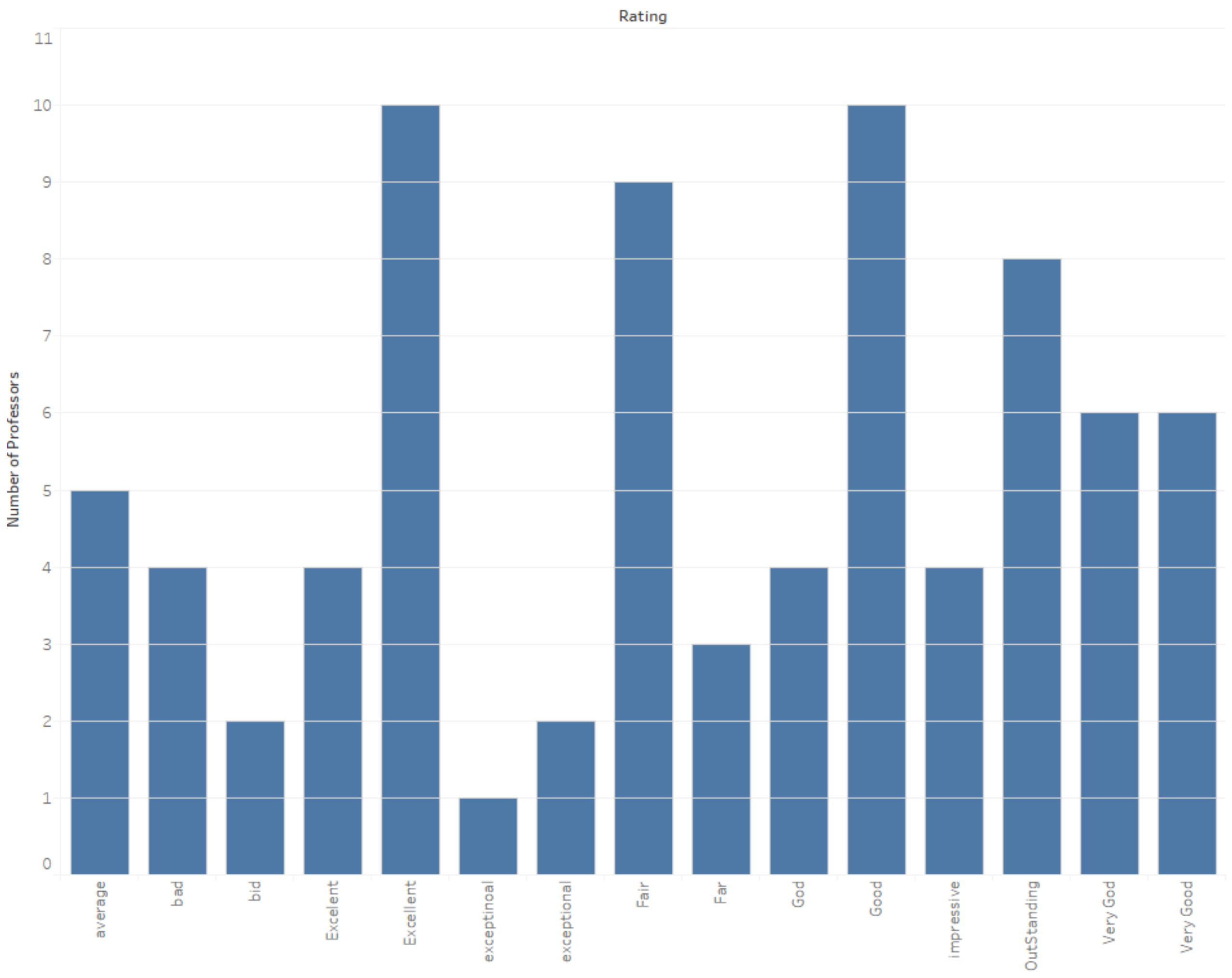


Figure 5.4 Professor rating before cleaning

In figure 5.4, the graph shows that are few values like bid, god and far in the professor rating attribute which doesn't fit into the context for rating. These spellings are correct but these are incorrect as per the context. The correct values should be bad, good and fair. So, we apply the cleaning algorithm for spelling correction using ontology in order to correct these values.

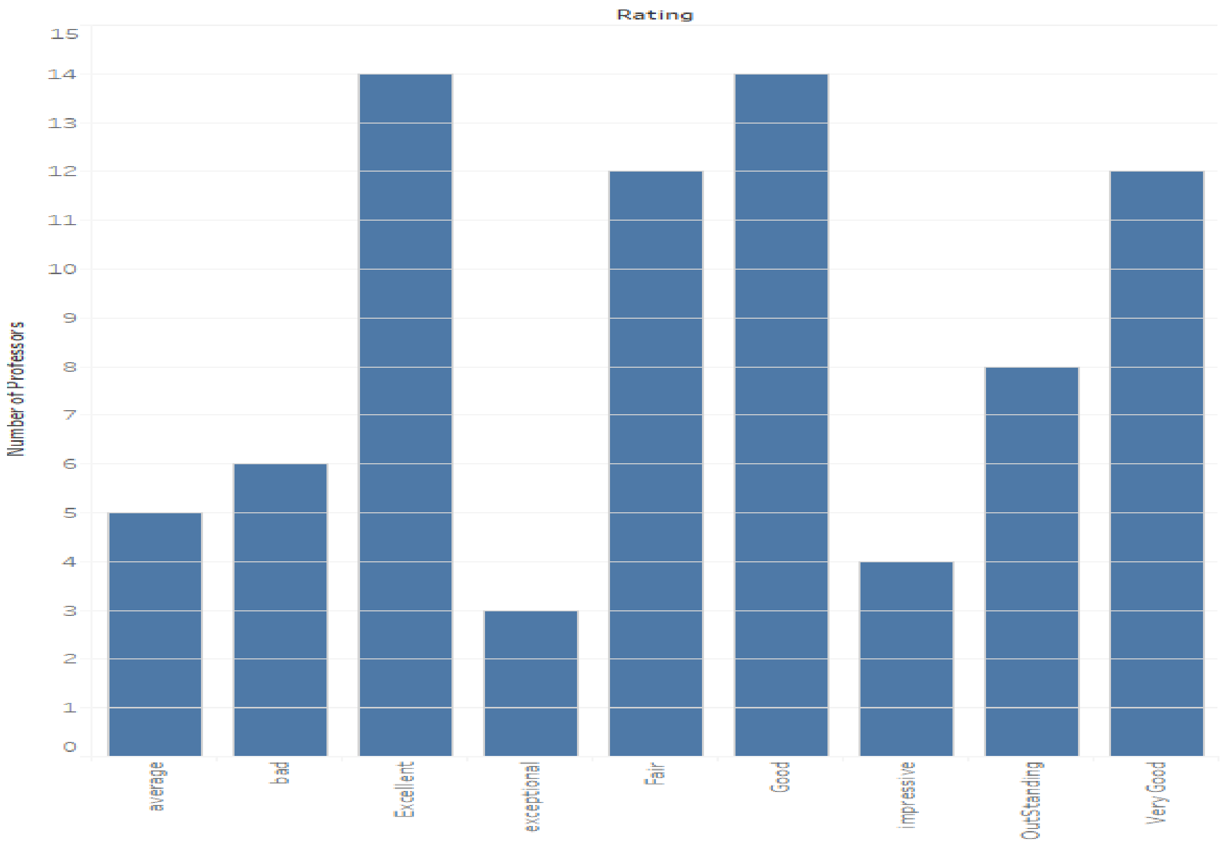


Figure 5.5 Professor rating after cleaning

In figure 5.5, the graph shows that the improper values of the attribute are corrected after cleaning. The difference in the distribution of the values can be observed in the graphs before cleaning and after cleaning.

Distinct count of Current State before and after cleaning

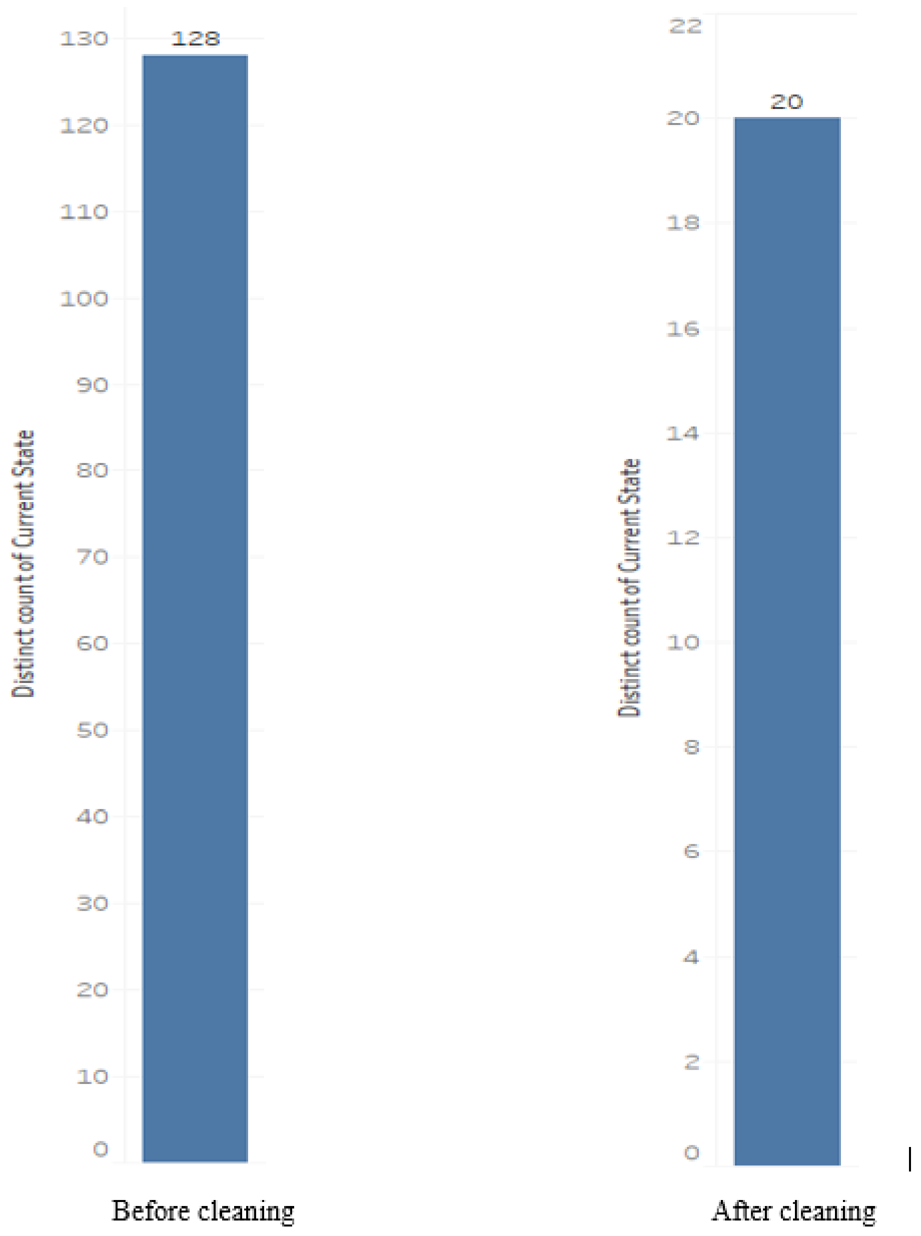


Figure 5.6 Distinct count of Current State before and after cleaning

The above figure shows the visualization graphs for the distinct count of current state attribute value before and after cleaning. We can observe that cleaning has impact on visualization.

CHAPTER 6 – CONCLUSIONS & FUTURE WORK

Data cleaning in graph databases is studied which was not done before. In this work, different cleaning algorithms have been proposed and implemented on unstructured graph databases. This work presents an approach to measure the impact of data cleaning on data quality using different cleaning algorithms. In this work, the spelling correction of proper nouns are performed on the corrupted data and also we use context ontology to correct the spellings. Data visualization techniques are used to validate the cleaning algorithms. This study reported on a range of findings after comparing the data with the results of visualization before and after cleaning. The data quality is enhanced and measured by visualization techniques using the tool Tableau. The results are then interpreted in the context of cleaning impact. The following are the main findings of this thesis.

Levenshtein's algorithm on spelling using context: The algorithms are used to correct spellings based on context (ontology) which helped in eliminating the improper attribute values which do not fit as per the context in the dataset thus improving data quality. For the current dataset, the data cleaning technique has achieved an accuracy up to 99.99% as the source is selected based on domain knowledge. The accuracy may differ in case of larger datasets since complete cleaning can't be achieved.

Cleaning has an impact on visualization: Data cleaning has reduced the outliers and data visualization provided a clearer picture. Thus making it more efficient to analyze the data.

In this thesis, the algorithms are applied on graph database that is compatible with Neo4j. In future, this study can be extended and applied on other graph databases such as Oracle Spatial and Graph, ArangoDB, GraphDB, AllegroGraph. Data cleaning algorithms for correcting missing values can be extended using heuristic approaches with minimal intervention of domain knowledge and applied on other graph databases. Data cleaning algorithms which work on defining outliers can be applied on graph databases to improve the data quality.

In this present work, Tableau is the only tool used for visualization. In future, other data visualization methods can be employed. Using a database, which has a large number of attributes is another area for further work.

References:

[1] E. Rahm, H. Do. Data Cleaning: Problems and Current Approaches. Bulletin of the Technical Committee on Data Engineering, Vol. 23, No. 4, 2000.

[2] H. Müller and J.-C. Freytag. Problems, methods, and challenges in comprehensive data cleansing. Technical report, Humboldt-Universität zu Berlin, Institut für Informatik, 2003.

[3] M. Gilleland, Levenshtein Distance in Three Flavors, http://www.merriampark.com/ld.htm.

[4] The Neo4j Developer Manual v3.0, http://neo4j.com/docs/developer-manual/current

[5] Ahmed K. Elmagarmid , Panagiotis G. Ipeirotis , Vassilios S. Verykios, Duplicate Record Detection: A Survey, IEEE Transactions on Knowledge and Data Engineering, v.19 n.1, p.1-16, January 2007

[6] P. Saravana Kumar, M. Athigopal, S. Vetrivel, Extract Transform and Load Strategy for Unstructured Data into Data Warehouse Using Map Reduce Paradigm and Big Data Analytics in International Journal of Innovative Research in Computer and Communication Engineering, December 2014.

[7] Tableau, http://www.tableau.com/learn, [October 10, 2016]

[8] Data cleaning on graph databases using Neo4j: duplicate elimination, data repair and correction of missing values.

VITA

Sarath Kumar Maddinani

Candidate for the Degree of

Master of Science

Thesis: DATA CLEANING ON GRAPH DATABASES USING NEO4J: SPELLING CORRECTION USING ONTOLOGY AND VISUALIZATION

Major Field: Computer Science

Biographical:

Education:

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in December, 2016.

Completed the requirements for the Bachelor of Sciences in Electronics and Communications at Jawaharlal Nehru Technological University, Hyderabad, India in 2012.

Experience:

Graduate Teaching Assistant, Oklahoma State University, Stillwater, OK, USA
August 2015 – December 2016
Worked as Graduate Teaching Assistant in the department of Computer Science. Assisted students answering technical queries during the lab hours.

iOS Application Developer, Oklahoma State University, Stillwater, OK, USA
May 2015 – August 2015
Developed Poinsettia Care Tips iOS app using Objective C. This app provides information regarding the tips that should be taken care for poinsettia plant. Also worked on developing few other apps.

Systems Engineer, Infosys Limited, India November 2012 - July 2014
Baker Hughes Well Link Wireline Enhancement 1.0:
Developed ADF modules of the application based on identified architecture. At the same time suggested better alternatives to the requirements, there-by making application more user-friendly. Developed the basic UI modules of the application in ADF. Performed unit testing of developed modules. Prepared Design Document, UTP's for newly created and modified modules.
Technologies Used**:** Java, JSP, JSF, JavaScript, HTML, CSS, Oracle ADF, MYSQL