2023-1

A Computational Model of Trust Based on Dynamic Interaction in the Stack Overflow Community

Patrick O'Neill
Technological University Dublin

A Computational Model of Trust Based on Dynamic Interaction in the Stack Overflow Community

Patrick O'Neill

A dissertation submitted in partial fulfilment of the requirements of Technological University Dublin for the degree of MSc. in Computer Science (Data Analytics)

September 2022

DECLARATION

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Analytics), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed: _____Patrick O'Neill_____

Date: 24 October 2022

ABSTRACT

A member's reputation in an online community is a quantified representation of their trustworthiness within the community. Reputation is calculated using rules-based algorithms which are primarily tied to the upvotes or downvotes a member receives on posts. The main drawback of this form of reputation calculation is the inability to consider dynamic factors such as a member's activity (or inactivity) within the community. The research involves the construction of dynamic mathematical models to calculate reputation and then determine to what extent these results compare with rules-based models. This research begins with exploratory research of the existing corpus of knowledge. Constructive research in the building of mathematical dynamic models and then empirical research to determine the effectiveness of the models. Data collected from the Stack Overflow (SO) database is used by models to calculate a rule-based and dynamic member reputation and then using statistical correlation testing methods (i.e., Pearson and Spearman) to determine the extent of the relationship.

Statistically significant results with moderate relationship size were found from correlation testing between rules-based and dynamic temporal models. The significance of the research and its conclusion that dynamic and temporal models can indeed produce results comparative to that of subjective vote-based systems is important in the context of building trust in online communities. Developing models to determine reputation in online communities based upon member post and comment activity avoids the potential drawbacks associated with vote-based reputation systems.

Keywords: *Stack Overflow, Question-Answer, Prediction, Regression, Computational Trust*

ACKNOWLEDGEMENTS

I would first like to express my sincere thanks to my supervisor Prof. Luca Longo for providing me help and encouragement throughout the research project.

TABLE OF CONTENTS

DECLARATION ..... **I**

ABSTRACT ..... **II**

ACKNOWLEDGEMENTS ..... **III**

TABLE OF CONTENTS ..... **IV**

TABLE OF FIGURES ..... **VIII**

TABLE OF TABLES ..... **XI**

TABLE OF ACRONYMS ..... **XII**

1 INTRODUCTION **1**

1.1 BACKGROUND 1

1.2 RESEARCH PROJECT/PROBLEM 1

1.3 RESEARCH OBJECTIVES 2

1.4 RESEARCH METHODOLOGIES 3

1.5 SCOPE AND LIMITATIONS 4

1.6 DOCUMENT OUTLINE 5

2 LITERATURE REVIEW **6**

2.1 COMPUTATIONAL TRUST BEGINNINGS 6

2.2 COMPUTATIONAL TRUST MODELS 6

2.3 REPUTATION MODELS 11

2.4 MACHINE LEARNING TRUST MODELS 13

2.5 SUMMARY 14

3 DESIGN AND IMPLEMENTATION **17**

3.1 DATA AVAILABILITY 19

3.2 DATA COLLECTION 19

3.3 DATA IMPORT 20

3.4 DATA QUALITY 21
3.5 DATA UNDERSTANDING 21
3.5.1 Descriptive Statistics Categorical Data 21
3.5.2 Descriptive Statistics Continuous Data 23
3.5.3 Visualizations Continuous Data 24
3.5.4 Visualizations Categorical Data 25
3.6 DATABASE SCHEMA 26
3.7 DATABASE INSTALLATION 26
3.8 QUARTERLY DATA DUMPS 27
3.8.1 Data Dump Files 27
3.8.2 Ubuntu for Windows 27
3.8.3 XML Parsers 27
3.8.4 Database Loading 28
3.8.5 Data Dump Issues 28
3.9 STACK EXCHANGE DATA EXPLORER 29
3.10 MODELLING 29
3.10.1 Rules Based Model 29
3.10.2 DIBRM Model 31
3.10.3 DIBRM Topic Model 36
3.11 EVALUATION 38
3.11.1 Hypotheses Testing 38
3.11.2 Strength and Limitations of Design 40
4 RESULTS AND EVALUATION 41
4.1 EXPLORATORY CORRELATIONS 42
4.2 CORRELATION TEST 1 – STACK OVERFLOW IN-HOUSE V RULE-BASED REPUTATION MODELS 44
4.3 CORRELATION TEST 2 - RULE-BASED V DIBRM MODEL REPUTATION MODELS 47
4.4 CORRELATION TEST 3 - RULE-BASED V DIBRM TOPIC MODEL REPUTATION MODELS 50
4.5 CORRELATION RESULTS SUMMARY 53
4.6 HYPOTHESIS TESTING OUTCOME 53

5 CONCLUSION 55

5.1 RESEARCH OVERVIEW 55
5.2 PROBLEM DEFINITION 55
5.3 DESIGN/EXPERIMENTATION, EVALUATION & RESULTS 55
5.4 CONTRIBUTIONS AND IMPACT 56
5.5 FUTURE WORK 56

BIBLIOGRAPHY 58

6 APPENDIX 62

6.1 DATA DESCRIPTOR 62
6.2 ENTITY TO TABLE COLUMN MAPPING 64
6.3 R CODE 65
6.4 DATABASE SCHEMA 84
6.4.1 Table DDL 84
6.5 XML FILES 90
6.5.1 Comments.xml 90
6.5.2 Posts.xml 90
6.5.3 Votes.xml 91
6.6 XML PARSERS 91
6.6.1 Comments XML Parser 91
6.6.2 Posts XML Parser 92
6.6.3 Votes XML Parser 93
6.7 ORACLE SQL*LOADER FILES 94
6.7.1 Control Files 94
6.7.2 Batch Files 97
6.8 SEDE SQL QUERIES 97
6.8.1 Data Volumes SQL 97
6.8.2 Data Extraction SQL 97
6.9 DATABASE VIEWS 99
6.9.1 Rules-based Model 99
6.9.2 DIBRM Models 100
6.10 PL/SQL PROCEDURE CODE 102
6.10.1 Rules based 102

6.10.2 Rules based Topic *105*
6.10.3 DIBRM Procedure Code *109*
6.10.4 DIBRM Topic Procedure Code *111*
6.11 IMPLEMENTATION ARTIFACTS 114

TABLE OF FIGURES

FIGURE 3.1 - OVERALL RESEARCH DESIGN ARCHITECTURE 17
FIGURE 3.2 – STACK OVERFLOW DATA VOLUMES (WEBSITE) 20
FIGURE 3.3 – SCORE FEATURE HISTOGRAM 24
FIGURE 3.4 – REPUTATION FEATURE HISTOGRAM 24
FIGURE 3.5 - SCORE DENSITY HISTOGRAM 24
FIGURE 3.6 - REPUTATION DENSITY HISTOGRAM 24
FIGURE 3.7 - SCORE FEATURE BOXPLOT 24
FIGURE 3.8 – REPUTATION FEATURE BOXPLOT 24
FIGURE 3.9 – COMMENT YEAR BAR CHART 25
FIGURE 3.10 – POSTTYPE FREQUENCY BAR CHART 25
FIGURE 3.11 - POST YEAR BAR CHART 25
FIGURE 3.12 – VOTETYPE FREQUENCY BAR CHART 25
FIGURE 3.13 - RULES-BASED REPUTATION ALGORITHM FLOWCHART 30
FIGURE 3.14 - CUMULATIVE INTERACTION V ACTIVITY SCATTERPLOT 32
FIGURE 3.15 - DYNAMIC REPUTATION PROFILE FOR USERID 300 33
FIGURE 3.16 - DIBRM REPUTATION ALGORITHM FLOWCHART 34
FIGURE 3.17 - DIBRM PROCESSING NODE DETAIL 34
FIGURE 3.18 -DYNAMIC TOPIC REPUTATION PROFILES FOR USERID 300 36
FIGURE 3.19 - DIBRM TOPIC REPUTATION ALGORITHM FLOWCHART 37
FIGURE 3.20 - DIBRM TOPIC PROCESSING NODE DETAIL 37
FIGURE 4.1 – COMMENT VOL. V REPUTATION SCATTERPLOT 42
FIGURE 4.2 - POST VOLUME V REPUTATION SCATTERPLOT 42
FIGURE 4.3 - VOTE VOLUME V REPUTATION SCATTERPLOT 42
FIGURE 4.4 – CORRELATION MATRIX 42
FIGURE 4.5 - DIBRM REPRODUCED MODEL 43
FIGURE 4.6 - DIBRM ORIGINAL MODEL (MELNIKOV, LEE, RIVERA, MAZZARA, & LONGO, 2018) 43
FIGURE 4.7 - STACK OVERFLOW REPUTATION (HISTOGRAM) 45
FIGURE 4.8 - STACK OVERFLOW REPUTATION (Q- Q PLOT) 45
FIGURE 4.9 - RULES-BASED REPUTATION (HISTOGRAM) 45
FIGURE 4.10- RULES-BASED REPUTATION (Q-Q PLOT) 45
FIGURE 4.11 - SCATTERPLOT OF STACK OVERFLOW AND RULES-BASED REPUTATION .. 46

FIGURE 4.12 - PEARSON CORRELATION RESULTS 46
FIGURE 4.13 - PAIRED T-TEST 47
FIGURE 4.14 - EFFECT SIZE 47
FIGURE 4.15 - RULES-BASED REPUTATION (HISTOGRAM) 48
FIGURE 4.16 - RULES-BASED REPUTATION (Q-Q PLOT) 48
FIGURE 4.17 - DIBRM REPUTATION (HISTOGRAM) 48
FIGURE 4.18 - DIBRM REPUTATION (Q-Q PLOT) 48
FIGURE 4.19 - SCATTERPLOT OF DIBRM VERSUS RULES-BASED REPUTATION 49
FIGURE 4.20 – SPEARMAN RANK CORRELATION RESULTS 49
FIGURE 4.21 - RULES-BASED TOPIC REPUTATION (HISTOGRAM) 51
FIGURE 4.22 - RULES-BASED TOPIC REPUTATION (Q-Q PLOT) 51
FIGURE 4.23 - DIBRM TOPIC REPUTATION (HISTOGRAM) 51
FIGURE 4.24 - DIBRM TOPIC REPUTATION (Q-Q PLOT) 51
FIGURE 4.25 - SCATTERPLOT OF DIBRM VERSUS RULES-BASED TOPIC REPUTATIONS .. 52
FIGURE 4.26 – SPEARMAN RANK CORRELATION RESULTS 52
FIGURE 6.1 – ORACLE DATABASE SCHEMA DATA MODEL 116
FIGURE 6.2 - ORACLE VIRTUAL BOX CONFIGURATION 116
FIGURE 6.3 - LINUX VM WITH PRE-INSTALLED ORACLE DATABASE 116
FIGURE 6.4 - STACK OVERFLOW DATA DUMPS 117
FIGURE 6.5 - DOWNLOADED DATA DUMP FILES 117
FIGURE 6.6 - DECOMPRESSED XML FILES 117
FIGURE 6.7 - UBUNTU FOR WINDOWS SCREENSHOT 117
FIGURE 6.8 - PARSER EXECUTION STATS. 118
FIGURE 6.9 - CSV FILE RECORD COUNTS 118
FIGURE 6.10 - POSTS DATA LOAD LOG 119
FIGURE 6.11 - COMMENTS DATA LOAD LOG 119
FIGURE 6.12 - VIRTUAL MACHINE STORAGE ISSUE 119
FIGURE 6.13 - VOTES DATA LOAD LOG 119
FIGURE 6.14 - SEDE TOOL SCREENSHOT 120
FIGURE 6.15 - COMMENTS DATA LOAD LOG 120
FIGURE 6.16 - USERS DATA LOAD LOG 120
FIGURE 6.17 - POSTS DATA LOAD LOG 120
FIGURE 6.18 – POST TYPES DATA LOAD LOG 120

FIGURE 6.19 – VOTES DATA LOAD LOG 121
FIGURE 6.20 – VOTE TYPES DATA LOAD LOG 121

TABLE OF TABLES

TABLE 2.1 – TRUST CONTEXT VARIABLES 9
TABLE 3.1 - NODE DETAIL LISTING 18
TABLE 3.2 - OVERALL STACK OVERFLOW DATA VOLUMES 19
TABLE 3.3 - IMPORTED DATA TO R DATA FRAME MAPPING 20
TABLE 3.4 - DF_COMMENTS VARIABLE STATS BY FREQUENCY 21
TABLE 3.5 - DF_POSTS VARIABLE STATS BY FREQUENCY 22
TABLE 3.6 - POST TYPE BY PERCENTAGE 22
TABLE 3.7 - DF_USERS VARIABLE STATS BY FREQUENCY 22
TABLE 3.8 - DF_VOTES VARIABLE STATS BY FREQUENCY 23
TABLE 3.9 - VOTE TYPE BY PERCENTAGE 23
TABLE 3.10 – CONTINUOUS DATA DESCRIPTIVE STATISTICS 23
TABLE 3.11 - RULES-BASED PROCESSING NODE DETAIL 31
TABLE 3.12 - CORRELATION COEFFICIENT 39
TABLE 4.1- CORRELATION TESTS 41
TABLE 4.2 - IMPORTED DATA 41
TABLE 4.3 - DESCRIPTIVE STATISTICS 44
TABLE 4.4 - PERCENT OF STANDARDIZED SCORES OUTSIDE THE ACCEPTABLE RANGE ... 45
TABLE 4.5 - DESCRIPTIVE STATISTICS 47
TABLE 4.6 - PERCENT OF STANDARDIZED SCORES OUTSIDE THE ACCEPTABLE RANGE ... 48
TABLE 4.7 - DESCRIPTIVE STATISTICS 50
TABLE 4.8 - PERCENT OF STANDARDIZED SCORES OUTSIDE THE ACCEPTABLE RANGE ... 51
TABLE 4.9 - CORRELATION RESULT SUMMARY 54
TABLE 6.1 - STACK OVERVIEW ENTITIES LIST 114
TABLE 6.2 - DATA DESCRIPTOR DETAIL 115
TABLE 6.3 – XML PARSER STATS. 118
TABLE 6.4 - DATA LOADING STATS 119
TABLE 6.5 – ORACLE DATABASE DATA LOADING STATS 121

TABLE OF ACRONYMS

Cases

CQA: Community Question Answering 1
CSV: Comma-Separated Value 27
DDL: Data Definition Language 26
DIBRM: Dynamic Interaction-Based Reputation Models 1, 15, 35
LSA: Latent Semantic Analysis 14
LTTM: Longo's Temporal Trust Model 7
ML: Machine Learning 14
NLP: Natural Language Processing 14
OSN: Online Social Network 6
PL/SQL: Procedural Language for SQL 2, 55
RF: Random Forest 14
SEDE: Stack Exchange Data Explorer 4
SO: Stack Overflow ii
SOM: Self-Organizing Map 7
SQL: Structured Query Language 4
TIM: Trust Inference Measuring 9
TPS: Trust Paths Searching 9
WSL: Windows Subsystem for Linux 27

1 INTRODUCTION

1.1 Background

Community Question Answering (CQA) websites have been around since the early 1990's and continue to grow in popularity. These websites allow registered members to ask questions to which they receive expert answers. CQA sites utilize a crowdsourcing sourcing model to obtain answers to posted questions. Members are primarily motivated to ask questions, by self-education through acquiring information (Choi, 2013) and to answer questions to enhance their reputation (Raban & Harper, 2008). CQA sites can host a broad range of topics, e.g., Yahoo! Answers, or can be corporate or specialist topic sites. Stack Overflow is a CQA website specializing in the topic of computer programming. Members can upvote or downvote questions, answers, and edits, which determines a value for a user's reputation. Computational Trust applies the human notion of trust to the digital world, that is seen as malicious rather than cooperative (Marsh, 1994). User reputation is a measure of how much the community trusts the user. This research focuses on models for the calculation of computational trust for the Stack Overflow community.

1.2 Research Project/problem

Problem Statement

The method used by (Melnikov, Lee, Rivera, Mazzara, & Longo, 2018) and (Yashkina, et al., 2019) to determine the efficiency of the Dynamic Interaction-Based Reputation Models (DIBRM), as detailed in 3.10.2, has the following gaps, rendering the efficiency of the DIBRM model inconclusive.

- The member reputation and historical reputation calculated by the rule based and DIBRM models are not comparable, without at minimum using scaling, both models calculate reputation differently, are on entirely different scales and are not comparable. The rule-based value is calculated based purely upon a member's peer voting whereas member reputation and historical reputation are calculated from member posts and comments.

- The algorithm used to calculate efficiency will converge to 100% efficiency as the volume of members in a community increase.

Hence the motivation of this research is to accurately determine the extent with which rule based and DIBRM model member calculated reputations compare, statistical correlation testing methods are required (i.e., Pearson and Spearman).

An additional gap relates to the calculation dynamic reputation in the context of the interaction i.e., the topic. Trust between individuals relates to the context (i.e., the topic) of the interaction. For example, if a mechanic serviced your car in the past, and did a good job, your trust with him or her would increase, in the context of car servicing, however this would not necessarily imply that your trust in the context of him or her fixing a leaking roof would likewise increase.

Research Question

To what extent do models, built utilizing dynamic interaction or temporal factors, approximate subjective voting of users within the Stack Overflow community?

1.3 Research Objectives

The research will be carried out using four sequential general objectives, where each is broken down by multiple specific objectives:
1) To design an experiment to determine the extent of the relationship between rules-based and dynamic interaction-based models.

- Identify a dataset.
- Execute an initial data collection to understand the data.
- Identify entities and features required for models.
- Design optimal method for data collection.
- Design the models to calculate rules-based and dynamic reputations.

2) To implement the design using the following tools and programming languages - Oracle Database, Oracle SQL*Loader, SQL, Oracle Procedural Language for SQL PL/SQL, Python and R.

- SQL is written to perform the data collection from the SEDE tool.
- XML parsers are written using python to parser the Stack Overflow Data Dump files and pipe to CSV files.
- CSV file data is loaded into the Oracle database using Oracle SQL*Loader tool.
- Rules-based and DIBRM models are implemented using PL/SQL.
- SQL is written to extract the model outputs to CSV.
- R is written to import model output data, create data visualizations and to execute statistical correction.

3) To run the experiment and run code

- SQL is run in SEDE tool to extract user, post, comment, and vote data from Stack Overflow database.
- Oracle SQL*Loader is run to load the data into the Oracle database.
- PL/SQL Model code is run to calculate member reputations.
- SQL is run to extract the model outputs data and for R integration.
- R is run to import that model output data into R.

4) To analyse findings and to answer the research question.

- Data visualizations are used to analysis the findings of the research.
- Correlation tests are executed for hypothesis testing by determining if there is a statistically significant result and additional the correlation coefficient is used to determine the extent of the relationship.

1.4 Research Methodologies

The type of research is secondary whereby existing Stack Overflow research and data will be collected and used to test the hypothesis.

The research objective methodology is quantitative, involving the systematic empirical investigation of quantitative Stack Overflow properties, phenomena, and their relationships. Mathematical models are developed in order to confirm the hypothesis. Research will provide the fundamental connection between empirical observation and the quantitative relationships in the data. All collected data will be numerical and analysed quantitative.

The research form includes Exploratory, Constructive and Empirical.

Exploratory research was utilized to structure and identify new problems in the evaluation of the quality of information in online communities. This helped to determine the best research design, data collection method and subject to select.

Constructive research was utilized in order to develop a solution to the research problem which also led to the development of the research hypothesis.

Empirical research was utilized to test the feasibility of the mathematical model using empirical / experimental evidence.

An inductive reasoning method was used to develop mathematical models i.e., bottom-up method from data to theory. Data was collected by observation; patterns in data were analysed using mathematical models, tentative hypothesis was created and then theory.

1.5 Scope and Limitations

The domain of the research is reputation systems and computational trust for the Stack Overflow community. Marsh's Ph.D. Thesis (Marsh, 1994) was the first publication referencing these domains. In recent years the domains of reputation systems and computational trust have become invaluable to improve computer-computer and human-computer interactions (Braga, Niemann, Hellingrath, & Neto, 2018). Both trust and reputation are subjective however the main distinction lies in the fact that trust is personal whereas reputation is not (Marsh, 1994).

A provable assumption is that publicly available Stack Overflow data required to support the execution of mathematical models is accessible via Data Dump downloading (Melnikov, Lee, Rivera, Mazzara, & Longo, 2018). During the design phase of the project the SEDE tool[1], was used to produce the dataset however does have a limitation restricting data extraction to 50k records per SQL query. A further limitation of the research is due to the researchers limited access to computational resources for data parsing and storage. R is used for data analysis and according to is limited to processing of data volumes in the region of one million records[2].

[1] https://data.stackexchange.com/stackoverflow/query/new;

Date Accessed: 12-Jun-2022

[2] https://www.r-bloggers.com/2013/11/five-ways-to-handle-big-data-in-r;

Date Accessed: 12-Jun-2022

A delimitation of the research, due to feasibility in the research timeframe, is that computation of reputation scores for CQA communities other than Stack Overflow are excluded, e.g., Wikipedia or Math Overflow. An additional delimitation, also due to feasibility to complete within the research timeframe is the building of a novel mathematical model for computational trust.

1.6 Document Outline

This section provides a summary of the five chapters contained in this document:

- Chapter 2 contains the Literature Review which was completed by reviewing and examining in detail research to date in the area of computation trust. Deep dives are taken into previous researcher theories and mathematical models used for assessing trust, particularly in the area of online communities.
- Chapter 3 provides the details of the phases of the Design and Implementation process. The Data Understanding phase begins by providing an overall integrated architectural design for the research project and then in sequence moves into the areas of data accessibility, initial data collection, describing the data, exploring the data and verifying data quality. The Data Preparation phase begins by providing detail of various data collection methods and techniques used for integration into a local database. Data selection additional discussed together with details related to data parsing, data loading and associated tools. The Modelling phase discusses the design of each mathematical model used in this research and how each are implementation. Detailed formulas for each model are provided, including flows charts, variable inputs, outputs, and processing logic. The Evaluation phase details the experiments completed to test the hypothesis using statistical correlation testing methods such as Pearson and Spearman. Finally, this chapter ends by outlining the strengths and limitations of the design.
- Chapter 4 discusses the Results and Evaluation of the model experiments, testing the research hypothesis. The correlation test results are presented including examining the strengths and limitations of the results and evaluation approach.
- Chapter 5 contains the conclusion, summarising the results found and highlighting areas for future work in the area of computational trust.

2 LITERATURE REVIEW

This chapter discusses the research conducted in the domain of Computational Trust, focussing specifically on trust models for online social networks (OSN). Here the design, implementation, and verification of Computational Trust (and Reputation) models, in date sequence are discussed and reviewed.

2.1 Computational Trust Beginnings

Marsh's Ph.D. Thesis (Marsh, 1994) was the first publication referencing the concept of trust in digital domains. Computational Trust applies the human notion of trust to the digital world, that is seen as malicious rather than cooperative (Marsh, 1994). In recent years the domains of reputation systems and computational trust have become invaluable to improve computer-computer and human-computer interactions (Braga, Niemann, Hellingrath, & Neto, 2018). Both trust and reputation are subjective however the main distinction lies in the fact that trust is personal whereas reputation is not (Marsh, 1994). Research around building computation models of trust and reputation for online communities' main purpose is to build the trustworthiness of communities.

2.2 Computational Trust Models

Marsh introduces a model to derive a value for Situational Trust (Marsh, 1994). See Equation 2.1 for the formula.

$$T_x(y,\alpha) = U_x(\alpha) \times I_x(\alpha) \times \widehat{T_x(y)}$$

Equation 2.1 - Situational Trust

where,

$$\widehat{T_x(y)} = \frac{1}{|A|} \sum_{\alpha \in A} T_x(y)$$

- Basic trust (T_x) is basic trust agent x holds derived from past experiences.
- General trust ($T_x(y)$) is a value representing the amount of trust agent x has for another y, not related to any specific situation. A value between -1 and 1 where 0 represents no trust.
- Utility ($U_x(\alpha)$) is the amount of known agent x gain from situation α.

- Importance ($I_x(\alpha)$) of situation α to agent x.
- Situational trust (T_x (y, α)) is the trust agent (x) has in agent (y) at situation α.

This model also introduces the notion of "reciprocation", whereby if an agent x helps an agent y in the past and y refuses to help x, then the trust x has in y will be reduced.
Longo et al. (Longo, Dondio, & Barrett, 2007) investigated the use of temporal-based factors, such as activity, frequency, regularity, and presence, as evidence of an entity's trustworthiness. A new algorithm called Longo's Temporal Trust Model (LTTM) was introduced and an evaluation were carried out using Wikipedia data involving 12000 users and 94000 articles. Algorithm prediction metrics were compared with Wikipedia ratings and satisfactory results were found. A good prediction rate was 60%, bad prediction rate was less than 20%, so it was determined that this approach can be useful in trust measurement and could be aggregated with more traditional methods like past direct experience and recommendation. The main drawback, of using temporal factors, found in the research, was the amount of information required, which may be difficult to collect for certain environments. Longo et al. (Longo, Pierpaolo, Riccardo, Barrett, & Butterfield, 2009) proposed a methodology to continuously align the LTTM model in force with the changing context within dynamic applications such as forums, blogs and p2p systems. The self-adaptation was reflected in the auto-organisation of the trust function aimed at assessing an agents' trustworthiness. The dataset used for evaluation was extracted from Finanzaoline[3] containing over more than 30,000 users, 1,000,000 threads and more than 11,000,000 messages. The preliminary results showed a good gain in the quality of prediction and that the methodology was promising. Longo et al. (Longo, Barrett, & Dondio, 2009) performed a context-dependent comparison between explicit human judgements, provided by 25 volunteers, and implicit judgements derived by using Computational Trust techniques. The evaluation was conducted using 12 websites it was demonstrated how, considering a digital entity as a website, human explicit judgement can be strongly correlated to the implicit derived value on the same entity. However, due the low volume of participants the results were deemed tentative. This was addressed using the unsupervised Kohonen neural network or self-organizing map (SOM) (Longo & Barrett, 2009) which enabled a large number of users behaviour patterns on internet webpages to be analysed, and to cluster common behaviours. This

[3] https://www.finanzaonline.com; Date Accessed: 02-Oct-2022

could be further adopted with Computational Trust model, to estimate the degree of trustworthiness of webpages. Further research by Longo et al. (Longo, Barrett, & Dondio, 2009) (Longo, Dondio, & Barrett, 2010) introduced a new Computational Trust model based on Information Foraging Theory to rank websites in order to build up a third generation *Social Search* engine based on implicit collaboration. 100 university students were recruited to explicitly evaluate the usefulness of 12 thematic websites and experiments was performed implicitly gathering their web-browsing activity. The research shows that, by considering the same searching query, *Social Search* was more effective than the Google Page Rank algorithm. In addition, it is shown that trust techniques can improve the quality of *Social Search* engine results (Dondio & Longo, 2011). Dondio et al. (Dondio & Longo, 2014) presented a knowledge-based system to compute the trustworthiness of digital entities. Starting from the set of presumptions that humans routinely use for assessing trust, the research describes a model to deploy a trust metric around those presumptions, called trust schemes. Here the efficacy of the trust model was evaluation for the online community FinanzaOnline.com, with a dataset of 80,000 registered users and about 9 million messages. A level of trustworthiness was calculated for each member and compared against an explicit poll by 298 users. The results here show the trust schemes could efficiently approximate the human judgment about trust in the context of a large online community.

Tomáš Švec et al. built a Multi-Context Trust Model using Python a mathematical model of trust to calculate trust for Facebook members based upon seven trust contexts (Tomáš & Samek, 2013). Equation 2.2 shows a priority vector for the model i.e., a weighted priority for a given context (1, 3, 2, 2, 1, 2, 3). Whereas Equation 2.3 provides the formula to calculate the trust value.

$$P = (T_S, T_N, T_C, T_F, T_P, T_G, T_L)$$

Equation 2.2 - Priority Vector

$$Tx = \frac{S \cdot T_S + N \cdot T_N + C \cdot T_C + F \cdot T_F + P \cdot T_P + G \cdot T_G + L \cdot T_L}{S + N + C + F + P + G + L}$$

Equation 2.3 - Trust Equation

Variable	Trust Context	Description
S	Interaction time span.	The longer the timespan spent on the community the larger the trust.
N	Number of interactions	The total count of interactions, i.e., posts, comments and likes. The larger the number of interactions the greater the trusts.
C	Number of characters	The number of characters in a message associated to the credibility and hence the trust a member holds.
F	Interaction regularity	The more regular members engage with the community the great their trust.
P	Photo tagging	The higher the volume of tags a member receives the great the trust.
G	Group membership	The more groups two member share the higher the trust between them.
L	Common interests	People who share common interests will have higher trust.

Table 2.1 – Trust Context variables

Table 2.1 explains the seven context variables used by the model. Although the research had difficulties acquiring member consent to access their data, due to data privacy concerns. Overall, for the sample of members who participated the results show that the model could evaluate the correct trust with 48.3% probability.

Hamdi et al. built a mathematical Trust Inference within online Social Networks (TISoN) model (Hamdi, Bouzeghoub, Gancarski, & Yahia, 2013). Here the research describes the design and implements a novel Trust Paths Searching (TPS) algorithm together with a Trust Inference Measuring algorithm (TIM) for computational trust. Experimention using data from advogato.com to measure the effectiveness of TISoN concluded that their algorithm generated high quality trusted networks.

Gambhir et al. propose an Action-based Trust Model algorithm which calculates trust in online communities based upon actions a member performs in the community, leaving the user in control of their own reputation (Gambhir, Doja, & Moinuddin, 2014). Community actions that are used by the algorithm to calculate trust are: liking a post, commenting on a post, sharing a post, tag an image, posting a text as a status message, posting an image, posting a link or posting a video. The algorithm uses the trust factors shown in Equation 2.4 to calculate trust.

Factor	Description	Weight
Weight for Action (Wa):	Each action has an associated weight. A post or a share are given the highest weight since they involve the most member interaction effort and hence add more to trust.	Share= .008 Post= .008 comment= .007 like= .006 dislike = .006 tagging =.005 Post= .008
Weight for Post (Wp)	Each post type has an associated weight associated. Posting a photo for example is given more weight that posting a URL link.	Photo=.003 Message=.003 Video=.002 Link=.001
Weight for Category (Wc)	The category of member post, whether it be a violent image, or an inspirational quote will influence trust also. The latter increase and the former decreasing.	Sensitive category= -.009 Non-sensitive category= .001
Post Credibility (Pc):	Posted message are checked and verified against member chosen category and compared with a database of terms appropriate to that category. If they match pass the trust increases other message is forwarded for manual review.	

Equation 2.4 - Trust Factors

Singh et al. proposed a hybrid trust model for an online social network currently utilizes an action-based model and Context recommender (Singh & Yi Chin, 2016). Here the researchers built a hybrid multi-faceted model incorporated an enhanced trust algorithm, an enhanced context-based including a recommender-based trust, which was validated during user acceptance testing (UAT). The multi-faceted model of trust build was based on eight trusts attributes: honesty, reputation, competency, credibility, confidence, reliability, belief and although there is no standard method for producing the accuracy of the model overall the results achieved were deemed an improvement of the existing mechanism.

The research of Dutta et al. designed and implemented a trust based recommender system, called Context Aware Recommender Model (CARS), which factors in both trust

and context, to avoid data overload, when users are searching for content (Dutta & Kumaravel, 2016). The model uses the equations shown in Equation 2.5, Equation 2.6 and Equation 2.7 to determine the trust value that a target user c holds for a specific user p. The simularity term, sim (c,p) of Equation 2.5, is determined using Pearson Correlation.

$$P_{c,i} = \bar{r}_c + \frac{\sum_{p \in M} sim(c,p)(r_{p,i} - \bar{r}_p)}{\sum_{p \in M} |sim(c,p)|}$$

Equation 2.5 - Predicted rating for target user c on item i

$$T_{c(p,i)} = 1 - \frac{|P_{c,i} - r_{c,i}|}{Z_{\text{MAX}} - Z_{min}}$$

Equation 2.6 - Trust of target user c for p for a specific item i.

$$\text{WT}_{(c,i)} = T_{(c,i)} \left[X + Y . \frac{\sum_{q=1}^{m} W_q}{\sum_{q=1}^{r} W_q} \right]$$

Equation 2.7 - Context weighted Trust value

The dataset used in the research consisted of 2296 Movie ratings. The model considered context variables such as time, season, location, weather etc to ultimately calculate a context weighted trust for a searching user c has for content i and to then subsequently filter the returned recommendations based upon a threshold. The changing values of context parameters factors in the dynamic nature of trust.

2.3 Reputation Models

The trustworthiness of Wikipedia authors was determined using a Content-Driven Reputation Model which calculates an author's reputation (Adler & de Alfaro, 2007). It was determined that authors with low reputation had a higher probability of their edits being undone and visa-versa. An algorithm predicting reputation points "could be used to flag new contributions from low-reputation authors, or it could be used to allow only authors with high reputation to controversial or critical pages." (Adler & de Alfaro, 2007).

The research of Melnikov et al. (Melnikov, Lee, Rivera, Mazzara, & Longo, 2018) introduces a novel Dynamic Interaction-based Reputation Models (DIBRM) to modelling trust in online communities. This model is built around the concept that human trust is dynamic and considers, when calculating trust, the following factors: forgetting, cumulative, and activity period. The more interaction that occurs between members, in the short term, the more that trust increases. Here a user reputation at a point in time, a variable directly related to trust, is calculated using the formulae shown as Equation 2.8, Equation 2.9 and Equation 2.10. An additional historical reputation was calculated as an aggregate sum of previous user reputations up to a point in time.

$$I_n = I_{bn} + I_{cn}$$

Equation 2.8 - Interaction

$$I_{cn} = I_{bn} * \alpha * (1 - \frac{1}{A_n + 1})$$

Equation 2.9 - Cumulative Interaction

$$I_{cn} = I_{bn} * \alpha * (1 - \frac{1}{A_n + 1})$$

Equation 2.10 - Cumulative Interaction

$$\Delta_n = [\frac{t_n - t_{n-1}}{t_a}]$$

Equation 2.11 - Number of Periods between successive interactions

$$T_n = T_{n-1} * \beta^{\Delta_n} + I_n, \beta \in [0, 1]$$

Equation 2.12 – DIBRM Reputation

Stack Overflow datasets for user activity for four years from 15-Sep-2008 to 14-Sep-2012 (i.e., 15k users) were downloaded where post and comment activity data was used to run DIBRM models for different sets of input parameters. In addition, a rule-based model using voting data was built and run, to mimic Stack Overflow's own rule-based reputation system. DIBRM model efficiency was determined by comparing the calculated reputations by both models using the formulae shown as Equation 3.1 and Equation 2.14.

$$\mu_D = 1 - \frac{1}{N^2} * \sum_{i=1}^{N} (\frac{1}{D} * \sum_{j=1}^{D} |R_{Sij} - R_{Dij}|)$$

Equation 2.13 - DIBRM Reputation Model efficiency

$$\mu_H = 1 - \frac{1}{N^2} * \sum_{i=1}^{N} (\frac{1}{D} * \sum_{j=1}^{D} |R_{Sij} - R_{Hij}|)$$

Equation 2.14 - DIBRM Historical Reputation Model efficiency

According to the research the DIBRM historical reputation gave better results displaying 88% similarly when compared with Stack Overflow's own rules-based reputation system. In addition, it was shown that evaluation results are resistant to changes in factors (Activity period, Forgetting and Cumulative) and that the model is suitable for use in various other environments and communities. The research of (Yashkina, et al., 2019) further utilized the DIBRM model (Melnikov, Lee, Rivera, Mazzara, & Longo, 2018) however in this instance datasets from the Reddit and Math Overflow online social communities were utilized to evaluate the models. Data for a total of 4793 users was collected from Math Overflow over a three-month period whereas from Reddit data was collected for a period of three months only. The study reports that the DIBRM model mimics this reputation models for both Reddit and Math Overflow with a good degree of accuracy.

2.4 Machine Learning Trust Models

The following research papers although not directly associated with Computational Trust models were additionally assessed as all seek to improve the quality of data in online community websites and hence build trustworthiness in online communities. All machine learning models discussed use supervised learning classification or regression techniques.

De La Calzada et al. evaluated the quality of Wikipedia articles utilizing a step-two classification process i.e., firstly classifying the articles into either stabilized or controversial articles and then determining their quality classification (De la Calzada & Dekhtyar, 2010). Adaji et al. also built ML classifiers to predict the churn of SO expert respondents using features related to community activity including "the time between consecutive answer posts" etc (Adaji & Vassileva, 2015). Expert users are classified as

either “Churner” or “Loyal”. Experts are an asset to any community and prediction of possible churners could allow CQA community owners to target these uses with incentives to stay and thus increase the quality and trustworthiness of the community. Baltadzhieva et al. trained multiple linear regression models to predict Stack Overflow question score using author and question features (Baltadzhieva & Chrupała, 2015). In theory this predicted score value could be provided as feedback to the questioner to assist them to compose questions with improved quality and to receive a faster response. Higher quality data in online communities ultimately increases trust. Choetkiertikul et al. trained Random Forest (RF) classifier models to predict the best candidates to answer given SO questions (Choetkiertikul, Avery, Dam, Tran, & Ghose, 2015). This could be used to route questions to user groups who are willing and have the knowledge to answer them. Two prediction approaches are investigated here: 1) Feature based and 2) Social network based. This would decrease the number of unanswered questions, answer times and increase quality and trustworthiness. Alharthi et al. also built Machine Learning (ML) regression models to predict the question scores on SO (Alharthi, Outioua, & Baysal, 2016). The list of predictor variables included answer counts, accepted answer score, view counts, favourite counts, code length, comment counts and tag numbers. Lin et al. utilized Natural Language Processing (NLP) and ML techniques to predict the best answer for questions labelled “Python” on SO (Lin, Lin, & Schaedler, 2018). NLP techniques such as term frequency-inverse document frequency (tf-idf) and Latent Semantic Analysis (LSA) were utilized during feature engineering and applied to ML models such as RF and XGBoost to train classifiers to predict the best answer. Elalfy et al. predict best answers to SO questions using a hybrid model. Two modules are used in combination, where the first module is used to predict the best answers using content features model whereas the second one uses non-content features. Both were then combined in one hybrid model to determine the best prediction result (Elalfy, Gad, & Ismail, 2018).

2.5 Summary

From the literature review it can be concluded that computational models of trust need to factor in the context of the interaction and the dynamic nature of trust. There is no recognized or standard method for calculating the trustworthiness of online community information. Furthermore, there is no standard method for evaluation of models built to

calculate trust or reputation. Historically difficulties are encountered capturing, storing, and processing the large data volumes pertaining to the online CQA communities. For example, the "content-driven reputation" model of Adler et al. used English Wikipedia articles up to February 2007 only (Adler & de Alfaro, 2007). Zhang utilized a small SO dataset to train models for predicting duplicate questions (Zhang, Lo, Xia, & Sun, 2015). The SO dataset utilized in the research by Alharthi et al. mentions "we filtered out any question that does not have an accepted answer" and the final dataset included "12,077 questions with creation date between August 2008 and March 2009" only (Alharthi, Outioua, & Baysal, 2016). The imbalance of datasets used when training classification algorithms is not addressed by the research of (Adaji & Vassileva, 2015; Elalfy, Gad, & Ismail, 2018). The research by (Baltadzhieva & Chrupała, 2015; Choetkiertikul, Avery, Dam, Tran, & Ghose, 2015; Lin, Lin, & Schaedler, 2018) does not provide details of the hyperparameters used for the machine learning models, thus hindering further research reproduction. The research by (Baltadzhieva & Chrupała, 2015; Elalfy, Gad, & Ismail, 2018) does not mention whether the same hardware was utilized to build or evaluate the different machine learning models utilized. The method used by (Melnikov, Lee, Rivera, Mazzara, & Longo, 2018; Yashkina, et al., 2019) to determine the efficiency of the Dynamic Interaction-Based Reputation Models (DIBRM), as detailed in 3.10.2, has the following gaps, rendering the efficiency of the DIBRM model inconclusive.

- The member reputation and historical reputation calculated by the rule based and DIBRM models are not comparable, without at minimum using scaling, both models calculate reputation differently, are on entirely different scales and are not comparable. The rule-based value is calculated based purely upon a member's peer voting whereas member reputation and historical reputation are calculated from member posts and comments.
- The algorithm used to calculate efficiency will converge to 100% efficiency as the volume of members in a community increase.

Hence the motivation of this research is to accurately determine the extent with which rule based and DIBRM model member calculated reputations compare using statistical correlation testing methods (i.e., Pearson and Spearman). A complete COA dataset is used and full hardware details, the entire code set and all parameters necessary for further research are provided.

An additional gap relates to the calculation dynamic reputation in the context of the interaction i.e., the topic. Trust between individuals relates to the context (i.e., the topic) of the interaction.

Research Question

To what extent do models, built utilizing dynamic interaction or temporal factors, approximate subjective voting of users within the Stack Overflow community?

3 DESIGN AND IMPLEMENTATION

This chapter details the design, implementation and statistical analysis performed to determine if a correlation exists, and to what extent, between both rules-based and dynamic interaction reputation models (i.e., DIBRM) in the content of the Stack Overflow community. The results of the correlation hypothesis testing with answer the research question.

Research Hypothesis

Null hypothesis H_0: There is no correlation between models built using dynamic interactive algorithms and the rules-based Stack Overflow reputation model.

Alternate hypothesis H_1: If a model is built using a dynamic interactive algorithm based on cumulative, temporal and inactivity factors, THEN a correlation exists with the rules-based Stack Overflow reputation model with statistically significant results ($p < .05$).

The overall data flow design for the research project is detailed in Figure 3.1 below.

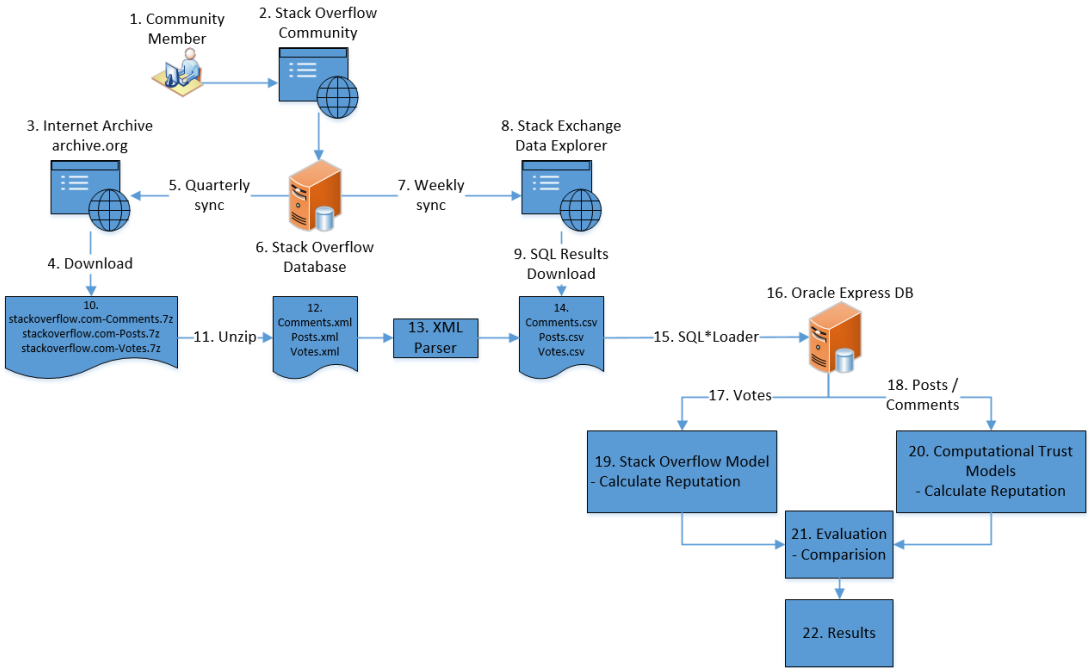


Figure 3.1 - Overall Research Design Architecture

The processing at each node of the flow is detailed in Table 3.1 below.

Node ID	Description	Processing Detail
1	Community member accessing Stack Overflow from personal laptop.	Web browser allowing navigation to Stack Overflow community URL.
2	Stack Overflow web servers.	Web servers delivering web pages to user browser.
3	Internet Archive website.	Storage of Stack Overflow data dumps.
4	Data Dump Download.	Downloading data dump files to desktop.
5	Quarterly data dump.	Stack Exchange providing quarterly data dumps.
6	Stack Overflow backend database.	Microsoft SQL Server database[4].
7	Weekly data sync.	Weekly data is synced to the Stack Exchange Data Explorer.
8	SEDE.	Tool allowing the execution of arbitrary SQL queries against data from the various question and answer sites in the Stack Exchange network[5].
9	CSV file format downloads.	Downloading SQL query results data to CSV format from SEDE.
10	Downloaded compressed datafiles.	At rest downloaded compressed datafiles.
11	7-Zip decompression utility.	Execution of 7-Zip decompression utility.
12	XML datafiles.	At rest XML datafiles.
13	Python XML parsers.	Execution of Python XML parsers.
14	CSV datafiles.	At rest CSV datafiles.
15	Oracle SQL*Loader.	Executing Oracle SQL*Loader to upload CSV file data into database[6].
16	Oracle database.	Oracle Express Database.
17	Votes data feed.	Votes data feed to Stack Overflow Reputation Model algorithm.
18	Post & comments data feed.	Post & comments data feed to Stack Overflow Computation Trust algorithm.
19	Stack Overflow rules-based PL/SQL procedure.	PL/SQL procedure implemented to mimic the rules-based Stack Overflow reputation model .
20	Computation trust PL/SQL procedure.	Computation trust PL/SQL procedure model implemented to calculate user reputation.
21	Comparison of rules based and computational trust models.	Comparison of rules based and Computational Trust Models
22	Results.	Publication of model comparison results.

Table 3.1 - Node Detail Listing

The CRISP-DM (CRoss Industry Standard Process for Data Mining) (Shearer, 2000) process for Data Mining is utilized to assess computation trust models for the Stack Overflow community.

[4] https://stackoverflow.blog/2008/09/21/what-was-stack-overflow-built-with; Date Accessed: 12-Jun-2022

[5] https://data.stackexchange.com/help; Date Accessed: 12-Jun-2022

[6] https://docs.oracle.com/cd/B19306_01/server.102/b14215/ldr_concepts.htm; Date Accessed: 12-Jun-2022

3.1 Data Availability

Stack Exchange provides the publicly available data via the following two methods [7]:

1) "Data Dumps" which are updated approximately quarterly on archive.org website, shown as nodes 3 and 5 of Table 3.1.
2) The SEDE tool updated weekly, shown as nodes 7 and 8 of Table 3.1.

Only a subset of the Stack Overflow data entities is available via the "Data Dumps", however all are available via the SEDE tool, as shown in columns "Available via SEDE?" and "Available via Data Dump?" in Table 6.1. Additionally, the data entities deemed relevant for building computational trust models are identify in Table 6.1 "Required for Research" column. The Stack Overflow reputation value within the community is calculated purely based upon votes received (i.e., vote entity), whereas computational trust algorithms calculate reputation based upon interactions within the community (i.e., post and comment entities). The data descriptor for each of Stack Overflow features community required for the research project is shown in Table 6.2.

3.2 Data Collection

The data volumes, for relevant entries, present in the Stack Overflow database at the time of writing this dissertation is shown in Table 3.2. These overall data volumes are determined used SEDE tool and SQL, see Appendix section 6.8.1.

Entity	Record Volume
Votes	231,441,846
VoteTypes	15
Comments	85,467,182
PostTypes	8
Users	17,922,426
Posts	56,264,788

Table 3.2 - Overall Stack Overflow Data Volumes

[7] https://meta.stackexchange.com/questions/2677/database-schema-documentation-for-the-public-data-dump-and-sede/326361#326361; Date Accessed: 12-Jun-2022

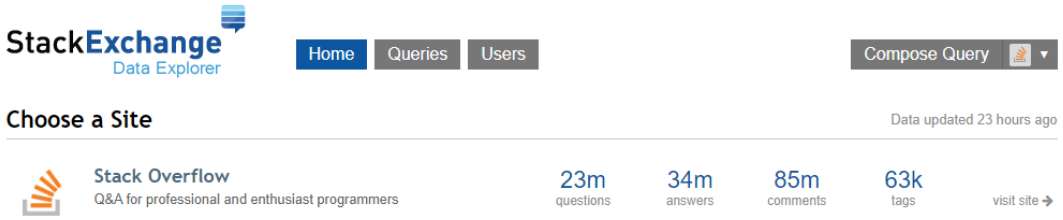


Figure 3.2 – Stack Overflow Data Volumes (website)

The volumes shown in Table 3.2 sync with those shown public on the Stack Exchange website [8], see Figure 3.2. R is chosen as the tool for data exploration and due to its one million record limitation, see section 1.5, a sample of the first 263 Stack Overflow registered members and their associated data is used for data analysis.

The SEDE tool is used to download the Stack Overflow community data, to six CSV files. As provided in Appendix section 6.8.2, SQL queries are written to query comment, post, post type, user, vote and vote type table data. Due to the SEDE 50k record limitation per SQL query, called out section 1.5, multiple queries (12 in all) are required to extract the 614k vote records.

3.3 Data Import

- All six CSV files are loaded into individual data frames using the R read.csv function, see Table 3.3.

CSV Files	Data Frame	Record Volume
Comments.csv	df_comments	48610
Posts.csv	df_posts	33328
PostTypes.csv	df_posttypes	9
Users.csv	df_users	237
Votes.csv	df_votes	614627
VoteTypes.csv	df_votetypes	16

Table 3.3 - Imported Data To R Data frame mapping

[8] https://data.stackexchange.com; Date Accessed: 12-Jun-2022

3.4 Data Quality

All data frames are checked for missing values. The ParentId missing values are expected since only posts of type question will have this value populated. In addition, BountyAmount missing values are also expected since these are only populated for votes of type 8 and 9 i.e., BountyStart and BountyClose. No data imputation is required in either case.

3.5 Data Understanding

To accurately model and design the database schema, see section 3.5.2, required for data storage a data understanding exercise was undertaken in conjunction with design phase where descriptive statistics and exploratory visualization are created.

3.5.1 Descriptive Statistics Categorical Data

See Table 3.4 to Table 3.7 below, for the descriptive statistics pertaining to the df_comments, df_posts, and df_users categorical features.

Id		UserId		PostId		CreationDate	
10	1	267	19982	22675886	16	Min.	2008-08-01
20	1	13	3868	5046373	15	1st Qu.	2010-01-26
22	1	67	1522	31146020	15	Median	2012-02-26
52	1	91	1118	31562791	15	Mean	2013-06-13
88	1	29	885	57312560	15	3rd Qu.	2016-05-25
162	1	157	655	6921194	14	Max.	2022-06-01
(Other)	48609	(Other)	20585	(Other)	48525		

Table 3.4 - df_comments variable stats by frequency

As shown in Table 3.4, the community member with Id of 267 with a volume of 19982 has made the most comments. It is also seen that PostId 22675886, with 16 comments, has had the largest volume of comments. The comments are seen to range from 01-Aug-2008 to 01-Jun-2022.

Id		PostTypeId		ParentId		UserId		CreationDate	
4	1	1	7117	1644	13	267	3192	Min.	2008-07-31
6	1	2	26083	373449	11	13	2140	1st Qu.	2008-12-18
7	1	4	50	5323	10	67	1163	Median	2009-11-20
9	1	5	47	1329	9	91	1023	Mean	2010-11-03
11	1	6	2	2658	9	234	720	3rd Qu.	2011-09-01
12	1			(Other)	26031	116	697	Max.	2022-06-01
(Other)	33293			NA's	7216	(Other)	24364		

Table 3.5 - df_posts variable stats by frequency

As shown in Table 3.5, the community member Id of 267 with 3192 posts also has the highest volume of posts. It is also seen that the largest volume of posts, 26083 records, are answer records (PostTypeId = 2). This can be also seen Table 3.6 where Post Type of 2 have 78% of the records. The posts are seen to range from 31-July-2008 to 26-May-2022.

1	2	3	4	5
21.373	78.329	0.150	0.141	0.006

Table 3.6 - Post Type by percentage

Id		CreationDate	
1	1	Min.	2008-07-31
2	1	1st Qu.	2008-08-01
3	1	Median	2008-08-02
4	1	Mean	2008-08-02
5	1	3rd Qu.	2008-08-03
8	1	Max.	2008-08-04
(Other)	230		

Table 3.7 - df_users variable stats by frequency

As shown in Table 3.8, records for 236 members are present who initially registered between from 31-July-2008 and 04-Aug-2008

Id		UserId		VotetypeId		PostId		CreationDate	
4	1	267	39177	2	521466	549	12343	Min.	2008-07-31
5	1	116	32361	5	73320	46155	6660	1st Qu.	2012-03-30
6	1	91	23772	1	8839	38578	6587	Median	2015-01-02
7	1	13	22118	3	8568	67699	6531	Mean	2015-01-21
9	1	136	17571	16	1549	57483	6239	3rd Qu.	2017-11-14
10	1	67	14511	15	254	237104	570905	Max.	2022-05-29
(Other)	614621	(Other)	465117	(Other)	631	(Other)			

Table 3.8 - df_votes variable stats by frequency

As shown in Table 3.8, the community member Id of 267 with 39177 votes has the highest volume of votes (both up and down votes). It is also seen that the largest volume of votes, 521466 records, are up votes records (VoteTypeId = 2). This can be also seen in Table 3.9 where Vote Type of 2 have 85% of the records. The votes are seen to range from 31-July-2008 to 29-May-2022.

1	2	3	5	6	8	9	10	11	15	16
1.438	84.843	1.394	11.929	0.004	0.034	0.023	0.015	0.026	0.041	0.252

Table 3.9 - Vote Type by percentage

3.5.2 Descriptive Statistics Continuous Data

See Table 3.10 for the descriptive statistics pertaining to the df_comments, df_users and df_votes continuous features. The reputation variable is seen to be multimodal with a range of 366289, where the lowest user reputation of 56.

Dataframe	Variable	vars	n	mean	sd	median	trimmed	mad	mode	min	max	range	Norm skew	Norm kurtosis	se	IQR	Q0.25	Q0.75
df_comments	Score	5	48609	0.816474	5.258934	0	0.288951	0	0	0	655	655	5270.771	256461.8	0.023853	1	0	1
df_users	Reputation	3	236	21521.92	39102.66	9461	13905.64	11875.63	731 1028 2221 2682 5105	56	366345	366289	33.10196	114.2403	2545.367	23574	2757	26331
df_votes	BountyAmount	6	614627	0.07594	4.787883	0	0	0	0	0	550	550	27639.61	1347357	0.006107	0	0	0

Table 3.10 – Continuous Data Descriptive Statistics

3.5.3 Visualizations Continuous Data

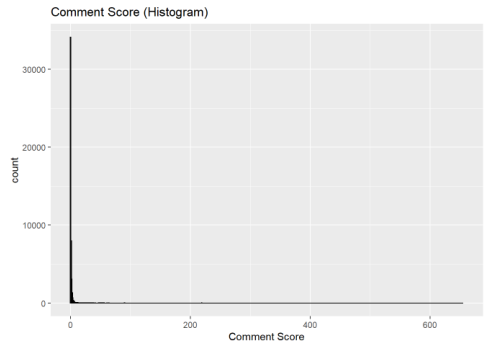


Figure 3.3 – Score Feature histogram

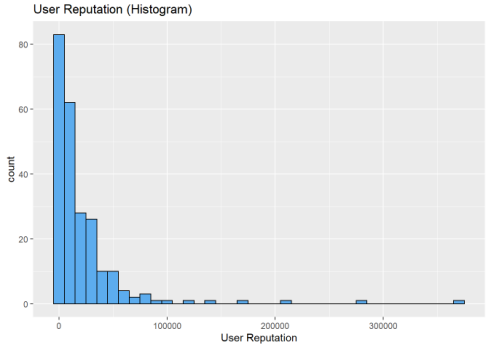


Figure 3.4 – Reputation Feature histogram

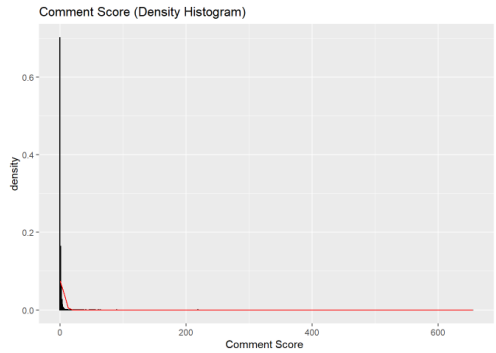


Figure 3.5 - Score density histogram

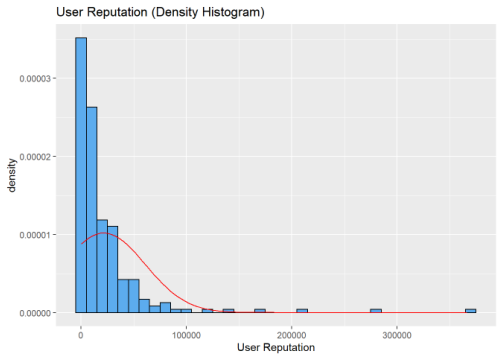


Figure 3.6 - Reputation density histogram

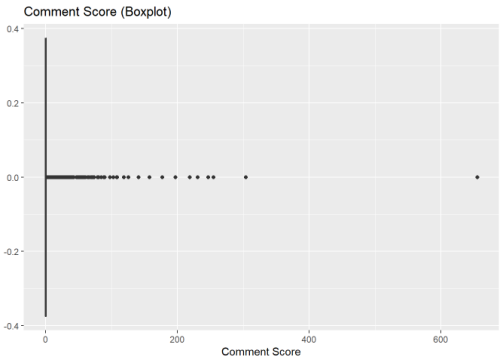


Figure 3.7 - Score Feature boxplot

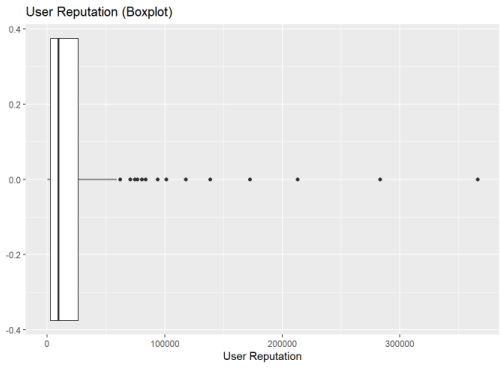


Figure 3.8 – Reputation Feature boxplot

As shown in Figure 3.5, the score feature is displaying a positively skewed distribution, this is also obvious from the outliers present in the boxplot of Figure 3.7. Summary

statistics calculated and shown in Table 3.10 confirm this since both the mode and median are less than the mean (Mean = 0.816, Median = 0 and Mode = 0).

As shown in Figure 3.4 , the reputation feature is displaying positive skewness and further analysis using a boxplot, see Figure 3.8, identified outliers. Summary statistics calculated and shown in Table 3.10 confirm this since both the mode and median are less than the mean (Mean = 21521.92, Median = 9461 and Mode = 731, 1028, 2221, 2682, 5105).

3.5.4 Visualizations Categorical Data

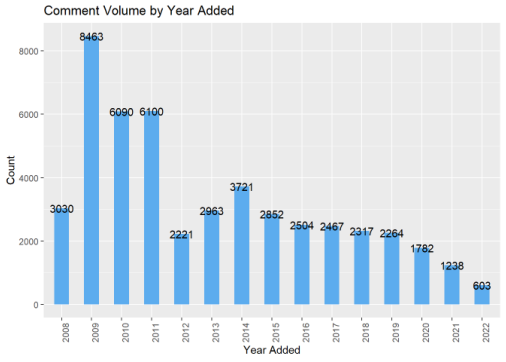


Figure 3.9 – Comment Year bar chart

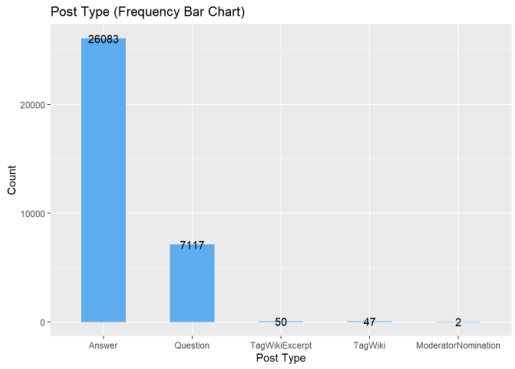


Figure 3.10 – PostType frequency bar chart

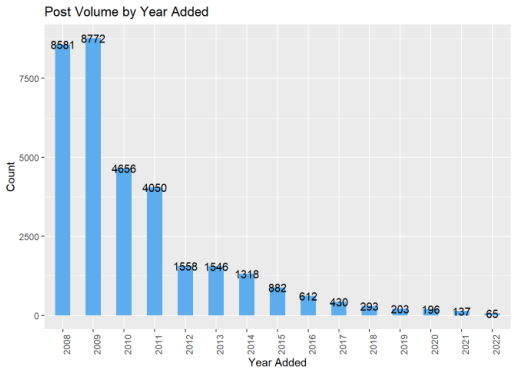


Figure 3.11 - Post Year bar chart

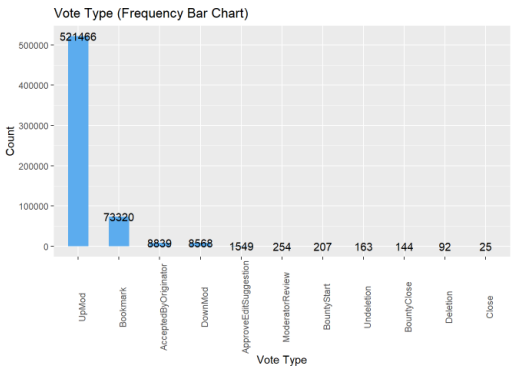


Figure 3.12 – VoteType frequency bar chart

As shown in Figure 3.9, the volume of comments, associated with the sample 263 user posts is threading downward year-on-year since 2014, in line with the post volume decreasing over the same period, see Figure 3.11. 2009 has the largest comment volume,

the year after Stack Overflow was launched[9]. As shown in Figure 3.10, the post type feature frequency distribution bar chart is ordered by post volume decreasing. This shows Answer and Question as the top two rated by content volume, accounting for 78.33% and 21.37% of the overall posts respectively, see Table 3.6. As shown in Figure 3.12, the vote type feature frequency distribution bar chart is also ordered by vote volume decreasing. This shows UpMod and Bookmarks as the top two rated by vote volume, accounting for 84.84% and 11.93% of the overall votes respectively.

3.6 Database Schema

The database schema shown in Figure 6.1 is built using Oracle SQL Developer Data Modeler[10] to store the Stack Overflow posts, comments, users and votes data required for reputation and Computational Trust calculations. DDL code is written to create the schema is provided in Appendix section 6.4.1. The data mapping from the attributes collected to the local data table storage is provided in Appendix section 6.2, see "Table Name" and "Table Column" columns.

3.7 Database Installation

A prebuilt Linux Virtual machine containing an Oracle database is downloaded from the Oracle Technology Network[11] and installed on the desktop. The steps involved in this process included:

1) As shown in Figure 6.2, Oracle VM VirtualBox is downloaded and installed on desktop system[12].
2) As shown in Figure 6.3, the Oracle Developer VM is downloaded and installed on the desktop.

---

[9] https://en.wikipedia.org/wiki/Stack_Overflow; Date Accessed: 12-Jun-2022

[10] https://www.oracle.com/ie/database/technologies/appdev/datamodeler.html; Date Accessed 12-Jun-2022

[11] https://www.oracle.com/database/technologies/databaseappdev-vm.html Date Accessed: 12-Jun-2022

[12] https://www.oracle.com/virtualization/technologies/vm/downloads/virtualbox-downloads.htm; Date Accessed: 12-Jun-2022

- The downloaded ova file is imported into Virtual Box[13] and is started up.

3.8 Quarterly Data Dumps

3.8.1 Data Dump Files

At the time of writing this research the latest Stack Overflow data dumps available on the archive.org were dated 07-March-2022[14]. See Figure 6.4 for the data dump files and sizes available in zipped format. As shown in nodes 4 and 10 of Figure 3.1, the files highlighted in Figure 6.4, totalling 24.5GB, is downloaded to home desktop on with broadband speed ~60Mbps taking in total approx. 1 hour. See Figure 6.5 for the downloaded files. As shown in nodes 11 and 12 of Figure 3.1, using 7-Zip[15] the downloaded compressed files when decompressed produce the following XML files, see Figure 6.6.

3.8.2 Ubuntu for Windows

Due the large XML file sizes windows had difficult counting file lines and parsing subset of lines; hence Windows Subsystem for Linux (WSL) is configured to allow Ubuntu to run on windows desktop[16]. See Figure 6.7 displaying the Ubuntu for Windows terminal.

3.8.3 XML Parsers

XML parsers are written using python to extract relevant XML tags from the XML files and spool the data to CSV files[17]. CSV files were used for ease of loading into the Oracle

[13] https://techgoeasy.com/pre-built-oracle-database-learning-testing-using-oracle-developer-vm; Date Accessed: 12-Jun-2022

[14] https://archive.org/details/stackexchange_20220307; Date Accessed: 12-Jun-2022

[15] https://www.7-zip.org/download.html; Date Accessed: 12-Jun-2022

[16] https://ubuntu.com/tutorials/install-ubuntu-on-wsl2-on-windows-10#1-overview; Date Accessed: 12-Jun-2022

[17] https://www.heatonresearch.com/2017/03/03/python-basic-wikipedia-parsing.html; Date Accessed: 12-Jun-2022

database. The python code for each of these parsers is provided in Appendix section 6.6. The volume of records parsed exceeds 360 million, taking approximately 2 hours to parse on home desktop, see Figure 6.8. The number of records written to the respective CSV files, see Figure 6.9. An additional one record is noticed in the CSV file due to the first header record. Table 6.3 displays a summary of the parsing execution volumes and timings per entity.

3.8.4 Database Loading

Oracle SQL*Loader[18] is used to upload CSV file data into the Oracle database. See Appendix section 6.7 for the code written to upload the data. A trial execution of SQL*Loader during the design phase produced the data loader log files seen in Figure 6.10, Figure 6.11 and Figure 6.13 respectively. The first header record is skipped by all data loads hence the CSV files have an additional record.

Table 6.4 displays a summary of the data loading execution volumes and timings for reach respective loads. NOTE: The votes data load is incomplete, see section 3.8.5 for a detailed explanation.

3.8.5 Data Dump Issues

As shown in Figure 6.12 the initial trial upload of votes data had to be abandoned, approximately 33% through, due to insufficient database storage resource on the virtual machine. The large data volumes involved caused not only storage issues but performance issues for SQL queries on the database on the virtual machine and hence a new approach is sought. This issue is called out as a limitation, see section 1.5. A new approach devised is to utilize post, vote and comment datasets pertaining to a sample of Stack Overflow members only. This data is extracted from Stack Overflow using the SEDE tool, see section 3.9.

[18] https://docs.oracle.com/cd/B19306_01/server.102/b14215/part_ldr.htm;
Date Accessed: 12-Jun-2022

3.9 Stack Exchange Data Explorer

The SEDE tool allows the execution of arbitrary SQL queries against data from the various question and answer sites in the Stack Exchange network[19]. Figure 6.14 shows the SEDE tool SQL query execution window together with the Download CSV button highlighted in red circle. Data is collected using the method described in section 3.2 and uploaded to the database using Oracle SQL*Loader method as described in section 3.8.4. The SQL*Loader log files for each data load are seen in Figure 6.15 to Figure 6.20 respectively. The first header record is skipped by all data loads hence the CSV files have an additional record. Table 6.5 displays a summary of the data loading execution volumes and timings for reach respective load.

3.10 Modelling

3.10.1 Rules Based Model

A mathematical model is built to recreate the rules-based Stack Overflow reputation algorithm. The algorithm primarily determines member reputation based upon votes cast by peers in the community. Members "who consistently provide useful content accrue reputation and are granted more privileges on the site" [20].

The following rules are modelled by the algorithm: [21]

Rule 1 - Members gain reputation points when a:

- question is voted up: +10
- answer is voted up: +10
- article is voted up: +10
- answer is marked "accepted": +15 (+2 to acceptor)
- suggested edit is accepted: +2 (up to +1000 total per user)
- bounty awarded to your answer: + full bounty amount

[19] https://data.stackexchange.com/help; Date Accessed: 12-Jun-2022

[20] https://stackoverflow.com/help/why-vote; Date Accessed: 12-Jun-2022

[21] https://stackoverflow.com/help/whats-reputation; Date Accessed: 12-Jun-2022

- one of your answers is awarded a bounty automatically: + half of the bounty amount (see more details about how bounties work)
- site association bonus: +100 on each site (awarded a maximum of one time per site)

Rule 2 - members lose reputation points when:

- a member's question is voted down: −2
- a member's answer is voted down: −2
- a member's article is downvoted: -2
- a member votes down an answer: −1
- a member votes downvote an article: -1
- a member places a bounty on a question: − full bounty amount
- a member post receives 6 spam or offensive flags: −100

Rule 3 - All members start with one reputation point, and reputation can never drop below 1

Rule 4 - A member can earn a maximum of 200 reputation per day from the combination of upvotes, downvotes and suggested edits

This model is implemented using Oracle PL/SQL and the code is available in Appendix section 6.10.1.

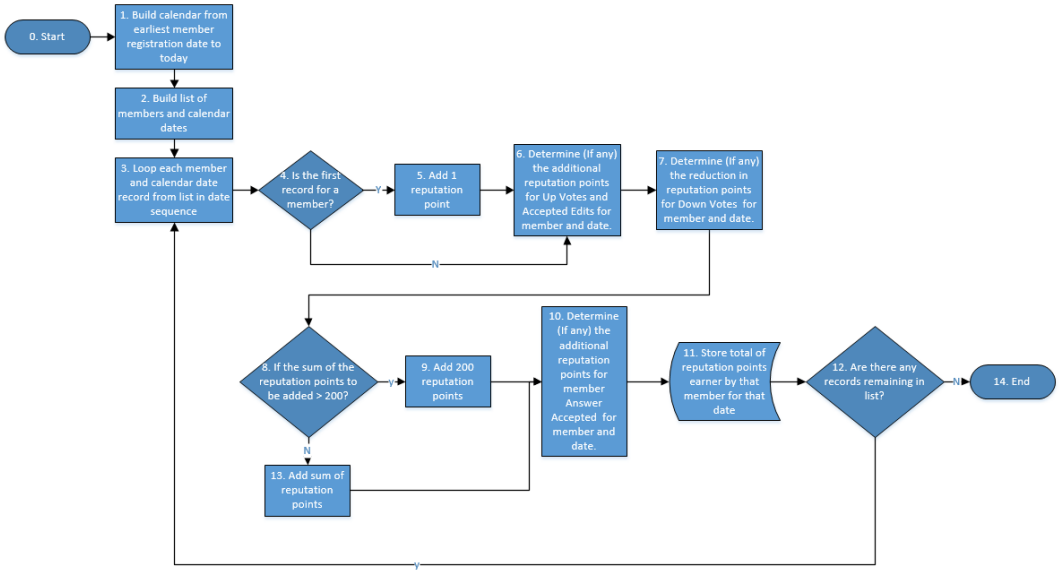


Figure 3.13 - Rules-Based Reputation Algorithm Flowchart

The processing flow for the rules-based algorithm is shown in Figure 3.13. See Table 3.11 for the processing that occurs at each note.

Node ID	Processing Detail
0	Start of processing.
1	Build calendar from earliest member registration date to today.
2	Build a list of all members by calendar date.
3	Loop through each record in member and calendar date sequence.
4, 5	Implement Rule 3 - All members start with one reputation point.
6	Implement Rule 1 - Up Votes & Accepted Edits.
7	Implement Rule 7 - Down Votes.
8, 9, 13	Implement Rule 4 - earn a maximum of 200 reputation per day from the combination of upvotes, downvotes and suggested edits.
10	Implement Rule 1 - Accepted Answers.
11	Storage of calculated member reputation points earner/lost for that day.
12	Check it there are remaining member and calendar date records to process.
14	End of processing.

Table 3.11 - Rules-Based Processing Node Detail

The rules-based value for the reputation of a user *i* on day *j* is defined as R_{Sij}.

3.10.2 DIBRM Model

Background

The Dynamic Interaction-Based Reputation (DIBRM) (Melnikov, Lee, Rivera, Mazzara, & Longo, 2018) introduces a novel approach to modelling trust in online communities. This model is built around the concept that human trust is dynamic and is primarily determined based around the level of interactivity between individuals. The more interaction that occurs between members, in the short term, the more that trust increases, e.g., if a mechanic serviced your car in the past, and did a good job, your trust with them would increase and hence if you required further work your trust level would indicate that this interaction would also be successful. The model implements the

concept of trust increasing using a variable called "cumulative factor". The model additionally factors in the notion that trust degrades overtime and hence if members have no interaction for long periods of time, their trust begins to decrease e.g., if you have not used a mechanic's services for some years your trust level would decrease. The model implements the concept of trust decreasing using a variable called "forgetting factor". The model calculates a member's reputation (as opposed to trust) however reputation is a core ingredient for building trust.

Model Description

As shown by Equation 3.1, interactions between individuals at a point in time are modelled as I_n where n ϵ 0…N is the index of a specific interaction and N is the total interactions.

$$I_n = I_{bn} + I_{cn}$$

Equation 3.1 - Interaction

Interactions have a basic value I_{bn}. For example, in Stack Overflow, an interaction could be asking, responding to or commenting on a question. Each interaction type has a contribution to the member's reputation value and is defined as the basic value.

$$I_{cn} = I_{bn} * \alpha * (1 - \frac{1}{A_n + 1})$$

Equation 3.2 - Cumulative Interaction

As shown by Equation 3.2, Term I_{cn} captures the cumulative part of the interaction. The weight of the cumulative part is defined as α and determines its maximum value for I_{cn}. A_n is the total number of sequential activities of the member. As shown in Figure 3.14, for α = 1 and I_{bn} = 2, I_{cn} can have a maximum value of 2, i.e., (I_{bn}*α), for A_n ϵ 1…5.

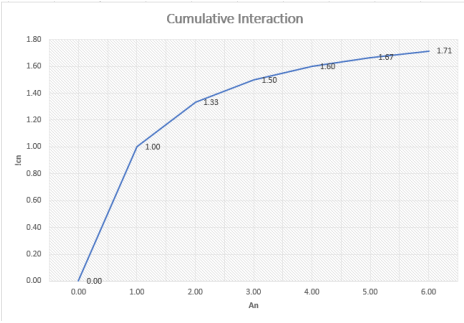


Figure 3.14 - Cumulative Interaction v Activity Scatterplot

$$\Delta_n = [\frac{t_n - t_{n-1}}{t_a}]$$

Equation 3.3 - Number of Periods between successive interactions

The frequency of interaction is modelled as the number of periods between the last two interactions and is defined as Δ_n where t_a is the typical period between interactions and t_n and t_{n-1} are the date and times of the last two interactions respectively.

$$T_n = T_{n-1} * \beta^{\Delta n} + I_n, \beta \in [0, 1]$$

Equation 3.4 - Reputation

The final equation modelling reputation of a member at a point in time is shown in Equation 3.4, where β is the forgetting factor. For Stack Overflow reputations for user *i* at a point in time *j* is called R_{Dij}, where $R_{Dij} = T_n$. The historical reputation R_{Hij} for Stack Overflow is defined as the cumulative sum of a member's reputation, an aggregate of the R_{Sij} value over interactions, see Equation 3.5.

$$R_{Hij} = \sum_{i=1}^{N} (R_{Dij})$$

Equation 3.5 - Historical Reputation

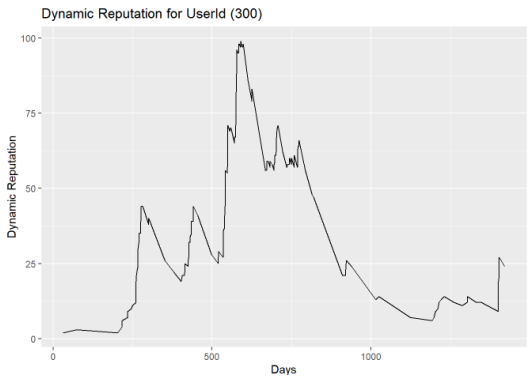


Figure 3.15 - Dynamic Reputation profile for UserId 300

The dynamic reputation of Stack Overflow user id 300 for the first 1400 days is shown in Figure 3.15. This historical reputation value will approximate the area below this curve.

The processing flow for the DIBRM algorithm is shown in Figure 3.16. See Table 3.11 for the processing that occurs at each note.

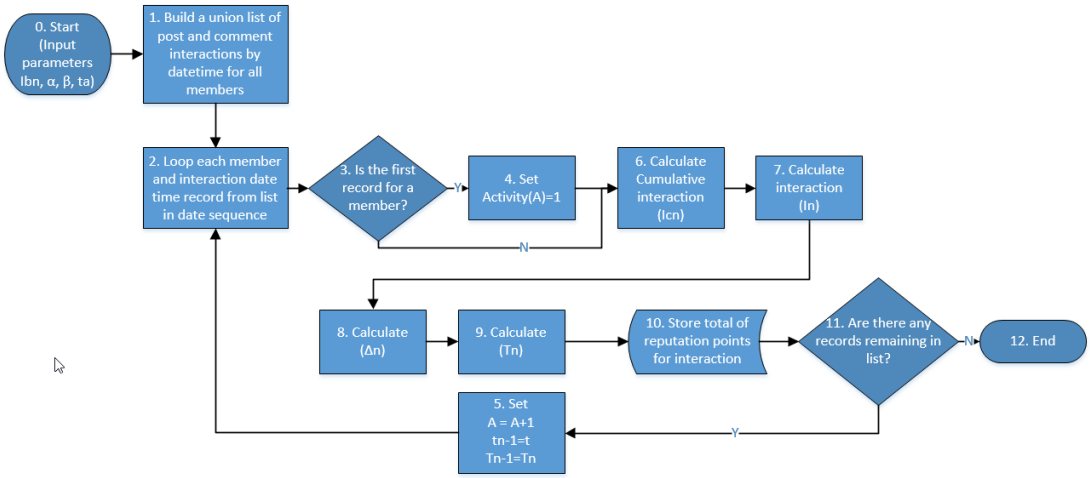


Figure 3.16 - DIBRM Reputation Algorithm Flowchart

Node ID	Processing Detail
0	Start of processing, setting input parameters, I_{bn}, α, β and t_a .
1	Build a union list of all post and comment interaction by date & time for all members.
2	Loop through list in member and interaction date and time sequence.
3, 4	Determine it this is the first interaction for member and if so, set A_n = 1.
5	Set all previous variable values.
6	Calculate cumulative interaction component, I_{cn}.
7	Calculate interaction, I_n
8	Calculate number of periods between last two interactions, Δ_n.
9	Calculate reputation, R_{Dij}.
10	Storage of calculated member reputation points.
11	Check it there are remaining member and calendar date records to process.
12	End of processing.

Figure 3.17 - DIBRM Processing Node Detail

Metric of approximation

The overall efficiency of the DIBRM model is determined by comparing the DIBRM reputations with the rules-based reputation value (R_{Sij}) for members on a particular day.

Two equations are used to calculate efficiencies. Equation 3.6 uses R_{Dij} to compare against the R_{Sij} whereas Equation 3.7 using R_{Hij}.

$$\mu_D = 1 - \frac{1}{N^2} * \sum_{i=1}^{N} (\frac{1}{D} * \sum_{j=1}^{D} |R_{Sij} - R_{Dij}|)$$

Equation 3.6 - DIBRM Reputation Model efficiency

$$\mu_H = 1 - \frac{1}{N^2} * \sum_{i=1}^{N} (\frac{1}{D} * \sum_{j=1}^{D} |R_{Sij} - R_{Hij}|)$$

Equation 3.7 - DIBRM Historical Reputation Model efficiency

N is defined as the number of users, where D is defined as the number of days between first and last dates.

Gaps Found in the Research

The method used by (Melnikov, Lee, Rivera, Mazzara, & Longo, 2018) and (Yashkina, et al., 2019) to determine the Dynamic Interaction-Based Reputation Models (DIBRM) model efficiency, as detailed above, has the following gaps, rendering the efficiency of the DIBRM model inconclusive:

- The rules-based reputation value R_{Sij} is not comparable with either of the DIBRM calculated R_{Dij} and R_{Hij} values as both models calculate reputation differently, are on entirely different scales and are not comparable. The R_{Sij} value is calculated based purely upon a member's peer voting whereas R_{Dij} and R_{Hij} are calculated from member posts and comments. At minimum if efficiency was to be determined using the method both variables would first require scaling,
- For simplicity if the result of averaging the difference of reputations over the total days (D) and total member volume (N) is identified by $AVG\Delta$ then Equation 3.7 becomes Equation 3.8 and hence the larger N becomes the more the algorithm's efficiency converges to 100%.

$$\mu_H = 1 - \frac{AVG\Delta}{N}$$

Equation 3.8 - Simplified Efficiency Calculation

where,

$$AVG\Delta = \frac{1}{N} * \sum_{i=1}^{N} (\frac{1}{D} * \sum_{j=1}^{D} |R_{Sij} - R_{Dij}|)$$

3.10.3 DIBRM Topic Model

The DIBRM model is extended with the introduction of trust between individuals related to the context (i.e., the topic) of the interaction. This model is implemented using Oracle PL/SQL and the code is available in Appendix section 6.10.4.

The dynamic reputation of Stack Overflow user id 300 for topic 4 topics volume for the first 1400 days is shown in Figure 3.18.

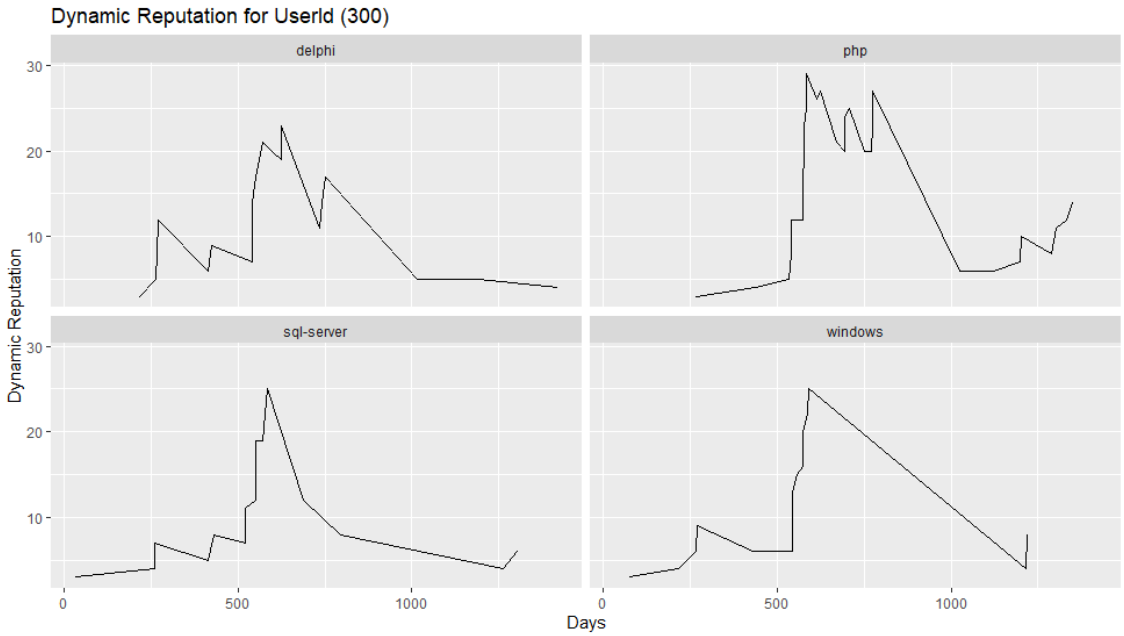


Figure 3.18 -Dynamic Topic Reputation profiles for UserId 300

Model Description

The formulas for the model are identical to those described in section 3.10.2 with the following exception. Interactions between individuals at a point in time on a topic are modelled as I_n, where n ϵ 0…N is the index of a specific interaction for that topic and N is the total interaction on that topic.

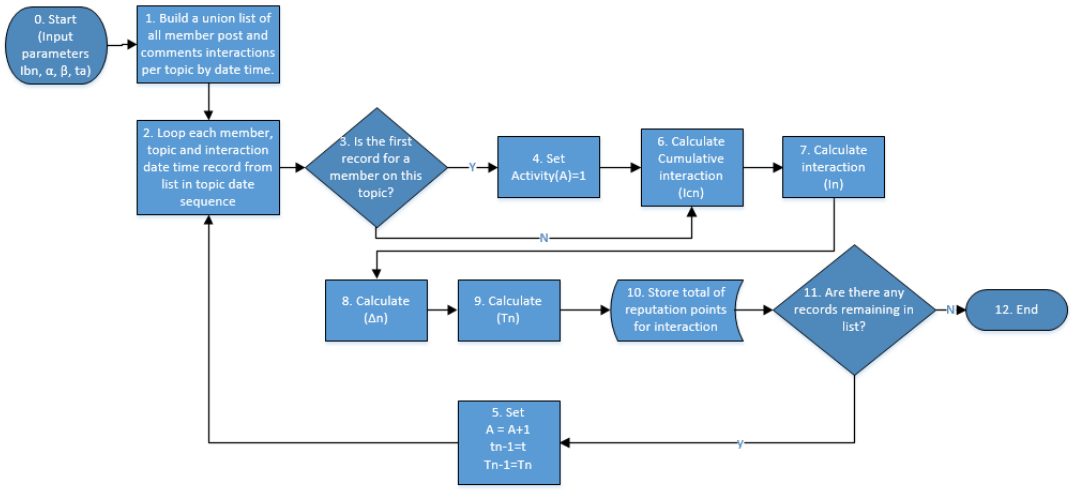


Figure 3.19 - DIBRM Topic Reputation Algorithm Flowchart

Node ID	Processing Detail
0	Start of processing, setting input parameters, I_{bn}, α, β and t_a .
1	Build a union list of all post and comment interaction per topic by date & time for all members.
2	Loop through list in member, topic and interaction date and time sequence.
3, 4	Determine it this is the first interaction for member on this topic and if so, set A_n = 1.
5	Set all previous variable values.
6	Calculate cumulative interaction component, I_{cn}
7	Calculate interaction, I_n
8	Calculate number of periods between last two interactions, Δ_n.
9	Calculate reputation, R_{Dij}.
10	Storage of calculated member reputation points.
11	Check it there are remaining member, topic and calendar date records to process.
12	End of processing.

Figure 3.20 - DIBRM Topic Processing Node Detail

3.11 Evaluation

3.11.1 Hypotheses Testing

Null hypothesis H_0: There is no correlation between models built using dynamic interactive algorithms and the rules-based Stack Overflow reputation model.

Alternate hypothesis H_1: If a model is built using a dynamic interactive algorithm, THEN a correlation exists with the rules-based Stack Overflow reputation model with statistically significant results ($p < .05$).

The cut-off p-value (probability value) (i.e., specified Significance level α) set for this research domain is 0.05 ($\alpha = 0.05$) hence with a 95% level of confidence statistically significant results are found if ($p < 0.05$) and hence the null hypotheses can be rejected. For example,

If p-value < α

- statistically significant result.
- evidence to reject the null hypothesis in favor of the alternate.
- Convention reports the p-values as $p < 0.05$
- Very small values of p (i.e., $p < 0.001$) are reported as $p < 0.001$

If p-value > α

- Not statistically significant result.
- No evidence to reject the null hypothesis

The hypothesis is tested using correlation statistical measure. Prior choosing the correlation test each scale feature is analysed to determine whether it conforms to the normal distribution or if the data can be considered to follow the normal distribution. When quantifying skew and kurtosis the following tests are used to determine if the data is a good fit for the normal distribution

- Shapiro-Wilks Test (sample size =<50)
- Kolmogorov-Smirnov Test (sample size > 50)

However, if these tests determine that data is not normally distributed the percentage of standardized skew and kurtosis scores that fall within an acceptable range (or heuristic) can then be calculated as shown in Equation 3.9.

Standardized score = value / std.error

Equation 3.9 - Standardized Score

If the standardized score for skewness and kurtosis lies between ± 2 (1.96 rounded) (George & Mallery, 2002) then this it is still acceptable to consider the data to follow a normal univariate distribution.

Quantification of the strength and direction of the relationship between the two variables i.e., the rules-based reputation and DIBRM reputation is determined using either the Pearson Correlation (Field, Miles, & Field, 2012) (for Parametric/Normal Distribution) or Spearman Rank Order Correlation (Field, Miles, & Field, 2012) / Kendall's Tau (for Non-Parametric/non-Normal Distribution) (Field, Miles, & Field, 2012). A correlation coefficient, r, is calculated to quantify the direction, a covariance calculated to quantify the strength and a statistical significance value. The correlation p-value is the probability value indicating whether the correlation results are statistically significant or not. If the p-value is less than the significance level ($p <= \alpha$, where $\alpha = 0.05$) and the correlation coefficient is significantly different from zero then the null hypothesis (H_0) is rejected, and the alternative hypothesis (H_1) is accepted. If the p-value is greater than the significance level ($p > 0.05$) them there is no-evident to reject the null hypothesis.

Correlation coefficient (r)	Description
-1	Strong negative correlation
0	No correlation
+1	Strong positive correlation

Table 3.12 - Correlation Coefficient

In addition to the outcome of the hypothesis test, the number of member reputation records (i.e., the degrees of freedom), the correlation coefficient, r, and the p-value are reported. Using the correlation coefficient, the magnitude of the strength and the direction of the relationship is commented on using heuristics (Cohen, 1998), see below.

- ±.1 = small/weak
- ±.3 = medium/moderate
- ±.5 = large/strong

3.11.2 Strength and Limitations of Design

Strengths

The strength of the design results from the use of PL/SQL stored database procedures to implement the models hence no further integration of data is required in order execute the models. For example, integration to R or Python to run models is not required. If a larger Oracle database environment is acquired, migration of the existing schema and models would require no rewrite. Integration of data from quarterly dump files is achievable with scheduled batch jobs monitoring for dump file timestamp updates.

Limitations

The limitations of the data storage and performance issues discussed in section 3.8.5, limits the models to utilize sample records for approx. 300 members only, taken using the SEDE tool which is limited to 50k records per query. Another limitation is that a subset of the rules are implemented to calculate Stack Overflow reputation in the model defined in section 3.10.1. For example, reputation point calculation including Bounty Amounts or site association. Using R_{Sij} (Stack Overflow rules-based reputation) as the ground truth for hypothesis testing may prove incorrect as this reputation calculation is entirely different to the dynamic model calculation and hence may not be suitable for comparison and hypothesis testing. A further limitation regards the sample sizes (or randomness) of records potentially not representative of the population.

4 RESULTS AND EVALUATION

This chapter provides all the details surrounding the complete set of tests executed for hypothesis testing and to ultimately answer the research question detailed in section 1.2.

The main high-level steps involved in conducting the hypotheses testing are as follows:

1. Data collection.
2. Model execution.
3. Data inspection (Bias, missing data, patterns).
4. Generate descriptive statistics.
5. Generate visuals (histograms and scatterplots).
6. Decide on normality.
7. Choose the correct correlation test.
8. Report the results of the correlation test.
9. Reject or accept the Null hypothesis H_0.

Test Id	Method	Reputation Model (x)	Reputation Model (y)
1	Pearson	SO	Rules-based
2	Spearman	DIBRM	Rules-based
3	Spearman	DIBRM topic	Rules-based

Table 4.1- Correlation Tests

See Table 4.1 for the correlation tests run to determine if there is a statistically significant relationship between the member reputation of Stack Overflow's own system and those calculated by the rule based, DIBRM and DIBRM Topic models.
The data collection was executed for a sample of the first 236 Stack Overflow registered members on Stack Overflow, downloaded and importing into the Oracle database schema, see Table 4.2 for the actual data volumes.

Comments	PostTypes	Posts	Users	VoteTypes	Votes
48615	8	33299	236	15	614873

Table 4.2 - Imported Data

The rules based, DIBRM and DIBRM Topic models were executed in the database using the following DIBRM model parameters (α = 1, I_{bn} = 2, t_a = 1, β = 0.99) (Melnikov, Lee, Rivera, Mazzara, & Longo, 2018).

The output for all models were downloaded to CSV files and imported into R studio for analysis and correlation testing.

4.1 Exploratory Correlations

Initially exploratory visualizations were plotted to assess potential relationships between features not directly related to the building of models. The relationships between the average volume of comments, posts (by a member) and votes (for a member) and member reputation were explored.

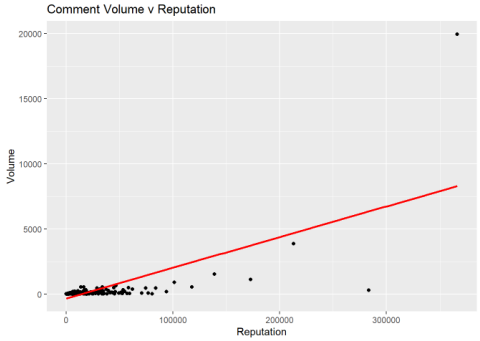


Figure 4.1 – Comment Vol. v Reputation Scatterplot

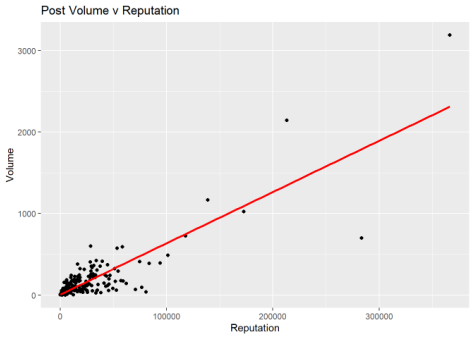


Figure 4.2 - Post Volume v Reputation Scatterplot

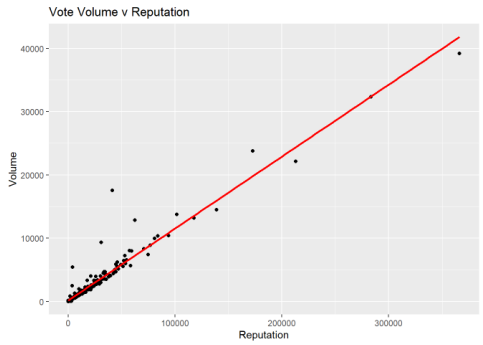


Figure 4.3 - Vote Volume v Reputation Scatterplot

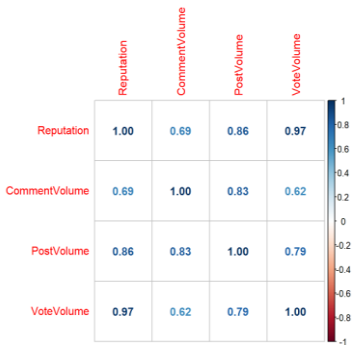


Figure 4.4 – Correlation Matrix

As shown in Figure 4.1, Figure 4.2 and Figure 4.3, all scatterplots display a positive correlation for the average comment, post and vote volumes per member versus user

reputation i.e., as the average volume increases the user reputation likewise increases. Figure 4.4 displays the correlation matrix of the average volumes per user for comment, post and vote plus the user reputation. Vote volume and reputation features have very strong correlation (r = 0.97). Post volume and reputation features have strong correlation (r = 0.86).

NOTE: Using the correlation coefficient, the magnitude of the strength and the direction of the relationship is commented on using heuristics (Cohen, 1998), see below.

- ±.1 = small/weak
- ±.3 = medium/moderate
- ±.5 = large/strong

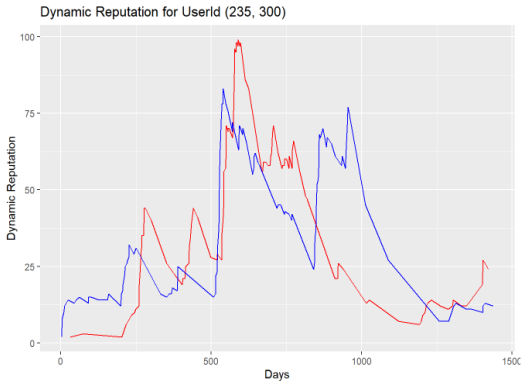


Figure 4.5 - DIBRM Reproduced Model

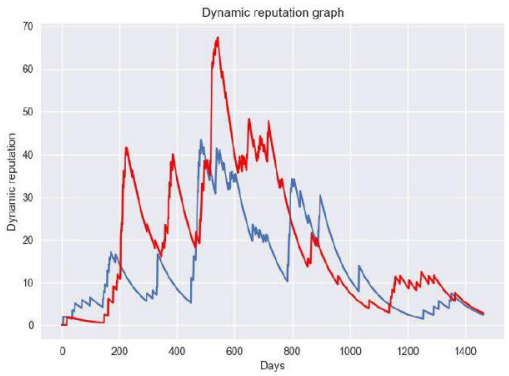


Figure 4.6 - DIBRM Original Model (Melnikov, Lee, Rivera, Mazzara, & Longo, 2018)

In order to validate that the DIBRM model constructed for this research was accurately reproducing the output of the original DIBRM model (Melnikov, Lee, Rivera, Mazzara, & Longo, 2018) reputations for two Stack Overflow members (235 and 300) for their first for 1500 days were plotted and compared. Figure 4.5 shown the DIBRM reproduced model whereas Figure 4.6 is from original research. Visually one can see that both visualizations have similar profiles however scaling is slightly different, most likely due to varying input parameters at model run-time.

4.2 Correlation Test 1 – Stack Overflow In-house v Rule-Based Reputation Models

To determine if there is a statistically significant correlation between Stack Overflows own member reputation and that of the rule-based model. Specifically, the comparison made was between the member reputation values of Stack Overflow itself, on the date of data collection (i.e., 01-Jun-2022), and the summation of the rule-based daily calculated reputations for each member from their registration up to (and including) the data collection date.

Descriptive statistics

Feature	n	mean	sd	median	trimmed	mad	min	max	range	Norm. skew	Norm. kurtosis	se	IQR	Q0.25	Q0.75
actrep	236	21521.92	39102.66	9461	13905.64	11875.63	56	366345	366289	33.102	114.24	2545.367	23574	2757	26331
synrep	236	22493.89	40824.61	9336.5	14278.01	11787.41	1	375562	375561	31.426	103.83	2657.456	23699.5	2701	26400.5

Table 4.3 - Descriptive Statistics

As shown in Table 4.3, the actrep and synrep features refer to the Stack Overflow's own member reputation system, and that of the rules-based model respectively. It is seen that actrep values range from 56 to 366345, whereas synrep range from 1 to 375562.

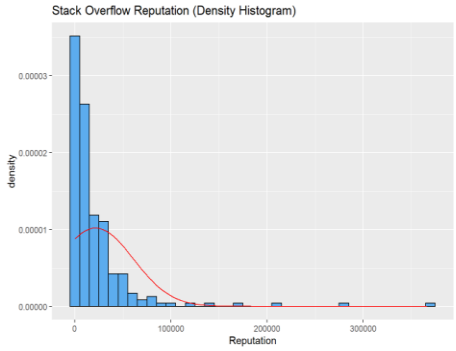


Figure 4.7 - Stack Overflow Reputation (Histogram)

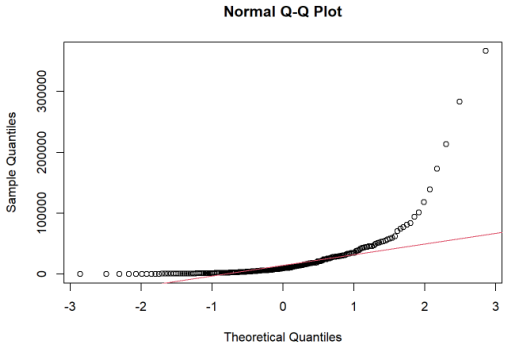


Figure 4.8 - Stack Overflow Reputation (Q-Q Plot)

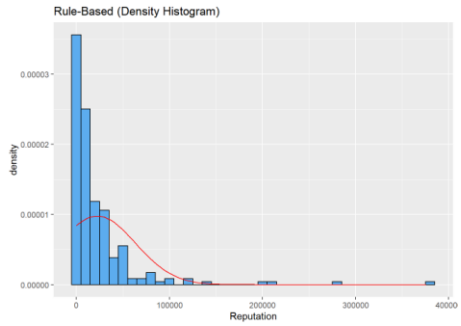


Figure 4.9 - Rules-Based Reputation (Histogram)

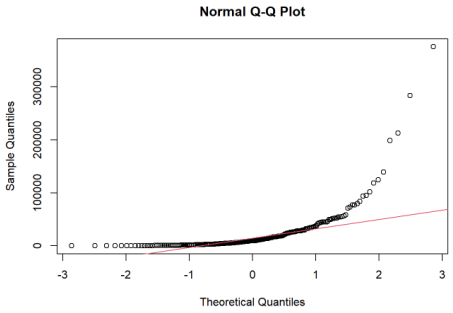


Figure 4.10- Rules-Based Reputation (Q-Q Plot)

As shown in Figure 4.7 and Figure 4.9, both histograms are displaying as positively skewed. Both Q-Q plots Figure 4.8 and Figure 4.10 are also displaying skewed distributions. Visually both these distributions are identical since the rules-based model implements Stack Overflow own in house rule-based reputation system.

Feature	% Standardized Scores < -1.96	% Standardized Scores > 1.96	% Standardized Scores < -3.29	% Standardized Scores > 3.29
actrep	0	2.9661	0	1.694
synrep	0	2.9661	0	1.694

Table 4.4 - Percent of standardized scores outside the acceptable range

The Stack Overflow reputation feature (actrep) was assessed for normality. Visual inspection of the histogram and Q-Q plot, see Figure 4.7 identified some issues with skewness and kurtosis. The standardized scores for skewness (33.102) and kurtosis

(114.24) were both outside the acceptable range, proposed by (West, Curran, & Finch, 1995). However as 99.03% of standardized scores, see Table 4.4, for reputation fall within the bounds of +/- 1.96, the data can be considered to approximate a normal distribution as outlined by (Field, Miles, & Field, 2012).

The rules-based model reputation feature (synrep) was assessed for normality. Visual inspection of the histogram and Q-Q plot, see Figure 4.9 identified some issues with skewness and kurtosis. The standardized scores for skewness (31.426) and kurtosis (103.83) were both outside the acceptable range, proposed by (West, Curran, & Finch, 1995). However as 99.03% of standardized scores, see Table 4.4, for reputation fall within the bounds of +/- 1.96, the data can be considered to approximate a normal distribution as outlined by (Field, Miles, & Field, 2012).

Since both variables both variables were found to approximate the normal distribution a Pearson correlation test was chosen.

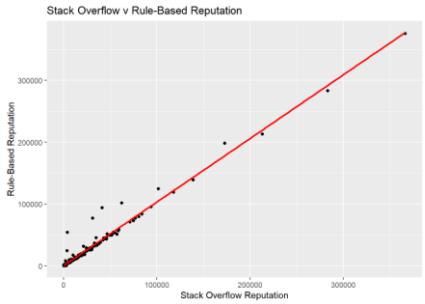


Figure 4.11 - Scatterplot of Stack Overflow and Rules-Based reputation

```
##
##  Pearson's product-moment correlation
##
## data:  df_actVsyn$actrep and df_actVsyn$synrep
## t = 90.931, df = 234, p-value < 0.00000000000000022
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9821225 0.9892648
## sample estimates:
##       cor
## 0.9861434
```

Figure 4.12 - Pearson Correlation results

As shown in Figure 4.11, a positive correlation between the Stack Overflow and the rules -based model is seen. As shown in Figure 4.12, the output provides Pearson's correlation co-efficient (0.986), the degrees of freedom (234) and the p-value. The p-value = 2.2e-16 (very small) i.e., $p < 0.001$ and hence the results are statistically signification.

The relationship between Stack Overflow reputation and rules-based reputation was investigated using a Pearson correlation (Field, Miles, & Field, 2012). A positive correlation was found ($r = 0.986$, $n = 234$, $p < .001$). Cohen's effect size (Cohen, 1998) indicated a strong effect size (0.986).

```
 Paired t-test

data:  df_actVsyn$synrep and df_actVsyn$actrep
t = 2.1733, df = 235, p-value = 0.03076
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
   90.85333 1853.07040
sample estimates:
mean difference
       971.9619
```

Figure 4.13 - Paired t-test

```
## d     |         95% CI
## ------------------------
## 0.28  | [0.03, 0.54]
```

Figure 4.14 - Effect Size

As shown in Figure 4.13 and Figure 4.14 a paired samples t-test was used to determine if there is a statistically significant difference between the mean reputations calculated by both reputation models.

A paired-samples t-test was conducted to evaluate Stack Overflow reputation and rule-based reputation. There was a statistically significant difference between the Stack Overflow reputations (M=21521.92, SD=39102.66) and the rule-based reputations (M=22493.89, SD=40824.61), t (235) = 2.173, p<.05). The mean increase in reputations was 971.96 with a 95% confidence interval ranging from 90.85 to 1853.07. Cohen's d also indicated a small effect size (0.28).

4.3 Correlation Test 2 - Rule-Based v DIBRM Model Reputation Models

To determine if there is a statistically significant correlation between the rules-based reputation and that of the DIBRM model. Specifically, the comparison made was between the rule-based daily calculated member reputations and the maximum daily DIBRM calculated member reputations, from member registration date until the date of data collection (i.e., 01-Jun-2022).

Descriptive statistics

Feature	n	mean	sd	median	trimmed	mad	min	max	range	Norm. skew	Norm. kurtosis	se	IQR	Q0.25	Q0.75
actrep	30823	32620.55	71905.23	4925	12602.83	6391.489	-1	375562	375563	220.57	325.096	409.5652	16843	1442	18285
synrep	30823	192.6548	316.4019	59	109.8031	68.1996	2	1716	1714	164.563	159.044	1.802194	150	21	171

Table 4.5 - Descriptive Statistics

As shown in Table 4.5, the actrep and synrep features, in this instance, refer to the reputation value calculated by the rule based and DIBRM models respectively. Notice that the volume of records has drastically increased (n = 30823) over those of the

previous test shown in Table 4.5 due the fact that both models calculate reputations per member per day (as opposed to per member as seen in 4.2). It is seen that the actrep values range from -1 to 375562, whereas synrep range from 2 to 1716.

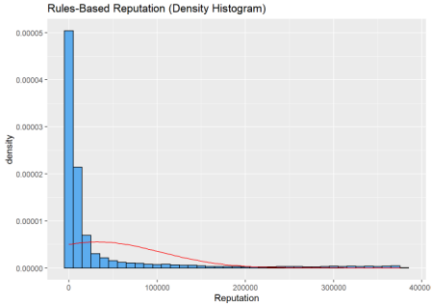


Figure 4.15 - Rules-Based Reputation (Histogram)

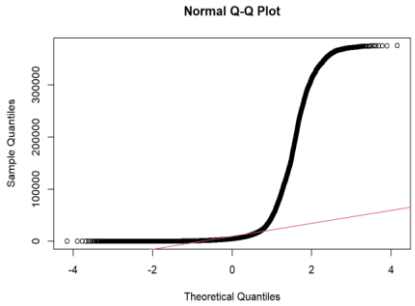


Figure 4.16 - Rules-Based Reputation (Q-Q Plot)

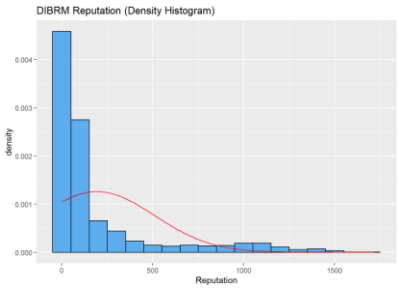


Figure 4.17 - DIBRM Reputation (Histogram)

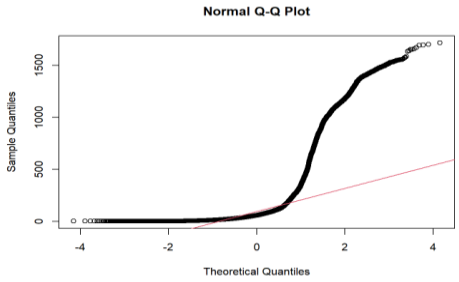


Figure 4.18 - DIBRM Reputation (Q-Q Plot)

As shown in Figure 4.15 and Figure 4.17, both histograms are displaying as positively skewed. Both Q-Q plots Figure 4.16 and Figure 4.18 are also displaying skewed distributions.

Feature	% Standardized Scores < -1.96	% Standardized Scores > 1.96	% Standardized Scores < -3.29	% Standardized Scores > 3.29
actrep	0	6.306	0	3.572
synrep	0	8.451	0	1.742

Table 4.6 - Percent of standardized scores outside the acceptable range

The rules-based model reputation feature (actrep) was assessed for normality. Visual inspection of the histogram and Q-Q plot, see Figure 4.15 identified some issues with

skewness and kurtosis. The standardized scores for skewness (220.57) and kurtosis (325.096) were both outside the acceptable range, proposed by (West, Curran, & Finch, 1995). As 96.43% of standardized scores, see Table 4.6, for reputation fall outside the bounds of +/- 3.29, the data cannot be considered to approximate a normal distribution as outlined by (Field, Miles, & Field, 2012).

The DIBRM model reputation feature (synrep) was assessed for normality. Visual inspection of the histogram and Q-Q plot, see Figure 4.17 identified some issues with skewness and kurtosis. The standardized scores for skewness (164.563) and kurtosis (159.044) were both outside the acceptable range, proposed by (West, Curran, & Finch, 1995). As 98.26% of standardized scores, see Table 4.6, for reputation fall outside the bounds of +/- 3.29, the data cannot be considered to approximate a normal distribution as outlined by (Field, Miles, & Field, 2012).

Since neither variable were found to approximate a normal distribution the Spearman rank correlation test was chosen.

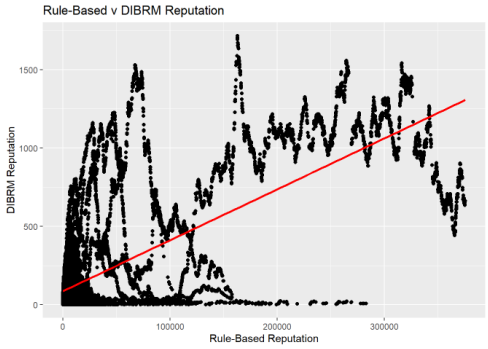


Figure 4.19 - Scatterplot of DIBRM versus Rules-Based reputation

```
Spearman's rank correlation rho

data:  df_actVsyn$actrep and df_actVsyn$synrep
S = 2480203342934, p-value < 0.00000000000000022
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.4918244
```

Figure 4.20 – Spearman Rank Correlation results

As shown in Figure 4.19, a positive correlation between the DIBRM and the rule-based model was seen. As shown in Figure 4.20, the output provides Spearman's correlation co-efficient (0.492) and the p-value. The p-value = 2.2e-16 (very small) i.e., $p < 0.001$ and hence the results are statistically signification.

The relationship between DIBRM reputation and rules-based reputation was investigated using a Spearman rank correlation (Field, Miles, & Field, 2012). A positive

correlation was found (ρ = 0.492, n = 30823, p < .001). Cohen's effect size (Cohen, 1998) indicated a moderate effect size (0.986).

4.4 Correlation Test 3 - Rule-Based v DIBRM Topic Model Reputation Models

To determine if there is a statistically significant correlation between the rules-based topic reputation and that of the DIBRM topic model. Specifically, the comparison made was between the rule-based daily calculated member reputations and the maximum daily DIBRM calculated member reputations for the topic for which the member has the highest volume of posts (i.e., their primary topic), from member registration date until the date of data collection (i.e., 01-Jun-2022).

Descriptive statistics

Feature	n	mean	sd	median	trimmed	mad	min	max	range	Norm. skew	Norm. kurtosis	se	IQR	Q0.25	Q0.75
actrep	10516	25866.21	43375.35	3278	15533.36	4628.677	-3	156655	156658	73.816	35.82	422.978	24240.75	723.75	24964.5
synrep	10516	194.2335	271.5187	49	139.9482	63.7518	2	1145	1143	61.013	17.354	2.647735	263.25	14	277.25

Table 4.7 - Descriptive Statistics

As shown in Table 4.7, the actrep and synrep features, in this instance, refer to the reputation values calculated by the rule-based topic and DIBRM topic models respectively. Notice that the volume of records has decreased (n = 10516) over those shown in Table 4.5, due to the fact that both models here calculate reputations per member per day per primary topic (as opposed to per member per day as per 4.3). It is seen that the actrep values range from -3 to 156655, whereas synrep range from 2 to 1145.

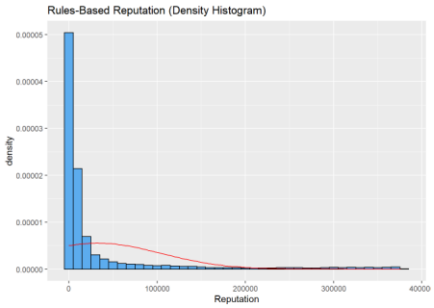


Figure 4.21 - Rules-Based Topic Reputation (Histogram)

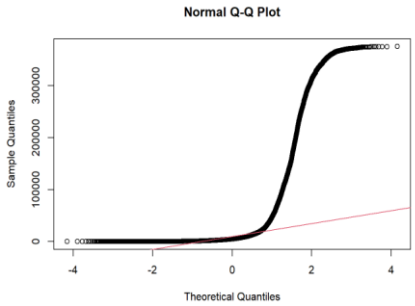


Figure 4.22 - Rules-Based Topic Reputation (Q-Q Plot)

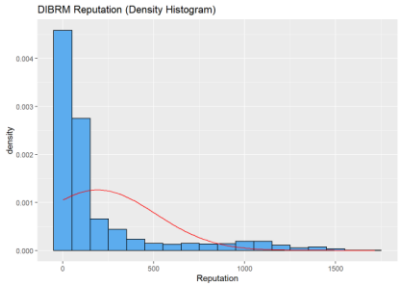


Figure 4.23 - DIBRM Topic Reputation (Histogram)

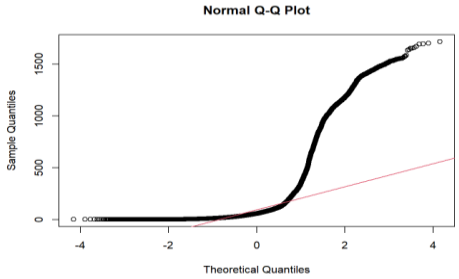


Figure 4.24 - DIBRM Topic Reputation (Q-Q Plot)

As shown in Figure 4.21 and Figure 4.23, both histograms are displaying as positively skewed. Both Q-Q plots Figure 4.22 and Figure 4.24Figure 4.18 are also displaying skewed distributions.

Feature	% Standardized Scores < -1.96	% Standardized Scores > 1.96	% Standardized Scores < -3.29	% Standardized Scores > 3.29
actrep	0	9.614	0	0
synrep	0	7.807	0	0.133

Table 4.8 - Percent of standardized scores outside the acceptable range

The rules-based topic model reputation feature (actrep) was assessed for normality. Visual inspection of the histogram and Q-Q plot, see Figure 4.23 identified some issues with skewness and kurtosis. The standardized scores for skewness (73.816) and kurtosis (35.82) were both outside the acceptable range, proposed by (West, Curran, & Finch, 1995). As 90.37% of standardized scores, see Table 4.8Table 4.6, for reputation fall

outside the bounds of +/- 1.96, the data cannot be considered to approximate a normal distribution as outlined by (Field, Miles, & Field, 2012).

The DIBRM topic model reputation feature (synrep) was assessed for normality. Visual inspection of the histogram and Q-Q plot, see Figure 4.23 identified some issues with skewness and kurtosis. The standardized scores for skewness (61.013) and kurtosis (17.354) were both outside the acceptable range, proposed by (West, Curran, & Finch, 1995). As 92.19% of standardized scores, see Table 4.8, for reputation fall outside the bounds of +/- 1.96, the data cannot be considered to approximate a normal distribution as outlined by (Field, Miles, & Field, 2012).

Since neither variable were found to approximate a normal distribution the Spearman rank correlation test was chosen.

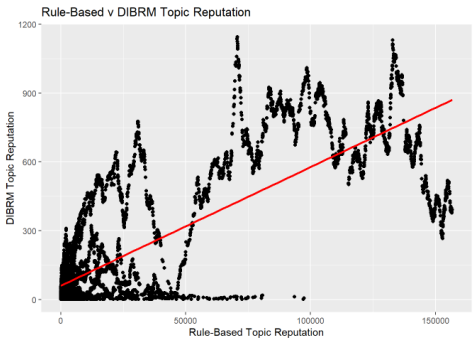


Figure 4.25 - Scatterplot of DIBRM versus Rules-Based topic reputations

```
Spearman's rank correlation rho

data:  df_actVsyn$actrep and df_actVsyn$synrep
S = 49533993506, p-value < 0.00000000000000022
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.7444341
```

Figure 4.26 – Spearman Rank Correlation results

As shown in Figure 4.25, a positive correlation between the DIBRM and the rule-based model was seen. As shown in Figure 4.26, the output provides Spearman's correlation co-efficient (0.744) and the p-value. The p-value = 2.2e-16 (very small) i.e., $p < 0.001$ and hence the results are statistically signification.

The relationship between DIBRM topic reputation and rules-based topic reputation was investigated using a Spearman rank correlation (Field, Miles, & Field, 2012). A positive correlation was found ($\rho = 0.744$, $n = 10516$, $p < .001$). Cohen's effect size (Cohen, 1998) indicated a moderate effect size (0.744).

4.5 Correlation Results Summary

From section 4.2 the results of the correlation testing of the relationship between Stack Overflow in-house reputation model and the Rule-Based Reputation model are.

- The relationship between Stack Overflow reputation and rules-based reputation was investigated using a Pearson correlation (Field, Miles, & Field, 2012). A positive correlation was found ($r = 0.986$, $n = 234$, $p < .001$). Cohen's effect size (Cohen, 1998) indicated a strong effect size (0.986).
- A paired-samples t-test was conducted to evaluate Stack Overflow reputation and rule-based reputation. There was a statistically significant difference between the Stack Overflow reputations (M=21521.92, SD=39102.66) and the rule-based reputations (M=22493.89, SD=40824.61), $t(235) = 2.173$, $p<.05$). The mean increase in reputations was 971.96 with a 95% confidence interval ranging from 90.85 to 1853.07. Cohen's d also indicated a small effect size (0.28).

From section 4.3 the results of the correlation testing of the relationship between DIBRM model and the Rule-Based reputation model are:

- The relationship between DIBRM reputation and rules-based reputation was investigated using a Spearman rank correlation (Field, Miles, & Field, 2012). A positive correlation was found ($\rho = 0.492$, $n = 30823$, $p < .001$). Cohen's effect size (Cohen, 1998) indicated a moderate effect size (0.986).

From section 4.4 the results of the correlation testing of the relationship between DIBRM model and the Rule-Based reputation model are:
The results of the correlation testing, conducted in section 4.4:

- The relationship between DIBRM topic reputation and rules-based topic reputation was investigated using a Spearman rank correlation (Field, Miles, & Field, 2012). A positive correlation was found ($\rho = 0.744$, $n = 10516$, $p < .001$). Cohen's effect size (Cohen, 1998) indicated a moderate effect size (0.744).

4.6 Hypothesis Testing Outcome

Research Hypothesis

Null hypothesis H_0: There is no correlation between models built using dynamic interactive algorithms and the rules-based Stack Overflow reputation model.

Alternate hypothesis H_1: If a model is built using a dynamic interactive algorithm, THEN a correlation exists with the rules-based Stack Overflow reputation model with statistically significant results ($p < .05$).

Test Id	Method	Reputation Model (x)	Reputation Model (y)	Degrees of Freedom(n)	Correlation Coefficient (r, ρ)	p-value (p)	Direction	Cohen's Effect Size heuristic
1	Pearson	SO	Rules-based	234	0.986	2.2e-16	Positive	strong
2	Spearman	DIBRM	Rules-based	30823	0.492	2.2e-16	Positive	moderate
3	Spearman	DIBRM topic	Rules-based	10516	0.744	2.2e-16	Positive	moderate

Table 4.9 - Correlation Result Summary

See Table 4.9 for the summary results for all three correlation tests. Test Id's 2 and 3 conclude there is statistically evidence ($p < 0.05$) to reject the null hypothesis H_0 and the in favour of the alternative hypothesis H_1. Hence the null hypothesis H_0 is rejected and the answer the research question "To what extent do models, built utilizing dynamic interaction or temporal factors, approximate subjective voting of users within the Stack Overflow community?" as follows: Models built using dynamic interaction or temporal factors do approximate the subjective voting of users within the Stack Overflow community to a "moderate" extent (Cohen, 1998).

The results do have limitations regarding whether the sample of members taken are truly representative of the population.

5 CONCLUSION

This chapter revisits the objectives of this research, the key findings, the contribution to the body of knowledge and recommended further work.

5.1 Research Overview

The objective of this research was to determine the extent with which rule based and DIBRM model member calculated reputations compare. The research was carried out using four sequential general objectives:

1) Design an experiment to determine the extent of the relationship between rules-based and dynamic interaction-based models.

2) Implement the design using the following tools and programming languages - Oracle Database, Oracle SQL*Loader, SQL, Oracle Procedural Language for SQL PL/SQL, Python and R.

3) Executing the experiments and choosing appropriate correlation test.

4) Critically analyse the findings and answer the research question.

5.2 Problem Definition

This research addresses the gap found in previous research utilizing the DIBRM model to calculate member reputation in online communities; whereby it was inconclusive if there was a relationship between subjective voting-based reputation and dynamic temporal reputation models.

5.3 Design/Experimentation, Evaluation & Results

This research designed, implemented and correlation tested the rule based versus DIBRM reputation models to determine if a relationship existed and if so to what extent. Sample data was collected from the Stack Overflow database, loaded into a local database where rule based and DIBRM models were built, run and outputs compared under various input parameter scenario. It was concluded that a moderate relationship does exist between these models. Strengths and limitations of the design were discussed with a view to recommending future work and research.

5.4 Contributions and Impact

The significance of the research is to add to the body of knowledge in the area of Computational Trust and to conclude that dynamic and temporal models can indeed produce results comparative to that of subjective vote-based systems. It is important that comparable alternative reputation models are developed for online communities since purely assessing reputation based upon member votes has potential for abuse. For example, online communities generally associate a value to member reputation, by providing increased privileges, access etc., this in turn potentially incentivises members to try improving their own reputation by gaming the system. This could occur by members creating fake profiles to vote up their own posts or down others, or to talk up themselves or down others in chatter. By implementing dynamic temporal reputation-based systems to determine reputation in an online community only members who truly interact and engage with the community on an ongoing basis (via posting and commenting) can improve their reputation. This is a more equitable from of reputation and is less open to abuse.

An additional gap which was addressed by this research relates to the determination that the calculation of dynamic reputation models in the context of the interaction also have a moderate relationship with rules-based models. Context is important when determining member reputation. For example, a member in Stack Overflow who currently has a high reputation value may indeed be a guru in java, but this reputation does not necessarily transfer to sql-server. If a member's reputation was context based, it would build a greater sense of trust within the community, as members would be able determine the ranked experts in particular specialties.

5.5 Future Work

There are many different avenues of research and possible future interesting engagements that could be spawned from this research and to further build computation trust models for online communities.

Increase Sample Sizes

It would be valuable to acquire some larger database storage resources, upload the full Stack Overflow Data Dumps and execute the DIBRM models to determine if results

found in this research are representative of the population. The current code set was built with scalability in mind so executing for larger datasets should not be an issue.

Other Communities

Apply the DIBRM models to new public online community data and access the results for comparison with their current model. Possibility of acquiring a corporate sponsor to implement the dynamic temporal reputation system on their corporate SaaS community platform and to compare model results with current reputation systems. Corporate community moderators generally have large knowledge of their domain of members and could quite easily determine the accuracy and value of the model.

Novel Models based on Computational Trust

It would of interest to start new research to design and implement a novel mathematical model to calculate trust, as opposed to reputation, for online communities. Trust in this instant would be a personal (or private) value a member holds regarding another member on a topic. For example, member A asks a question on a particular topic and member B responds with an answer accepted by member A, this increases the trust value member A has for B in the context of that topic. Additionally, member A can accept recommendations to increase the trust value they hold for another member only from those members A already has trust value.

Explainable Layer

Possible further work would be to take the dynamic model outputs and using explainable artificial intelligence (XAI) methods to add an explainable layer (Vilone & Longo, 2021) and to perform an analysis of convergent and face validity (Rizzo & Longo, 2018).

BIBLIOGRAPHY

Adaji, I., & Vassileva, J. (2015). Predicting churn of expert respondents in social networks using data mining techniques: A case study of stack overflow. *Proceedings - 2015 IEEE 14th International Conference on Machine Learning and Applications, ICMLA 2015*. doi:10.1109/ICMLA.2015.120

Adler, B., & de Alfaro, L. (2007). A Content-Driven Reputation System for the Wikipedia. *Proceedings of the 16th International Conference on World Wide Web - WWW '07* (pp. 261–270). Association for Computing Machinery. doi:10.1145/1242572.1242608

Alharthi, H., Outioua, D., & Baysal, O. (2016). Predicting Questions' Scores on Stack Overflow. *2016 IEEE/ACM 3rd International Workshop on CrowdSourcing in Software Engineering (CSI-SE)*, (pp. 1-7). doi:10.1109/CSI-SE.2016.009

Baltadzhieva, A., & Chrupała, G. (2015). Predicting the Quality of Questions on Stackoverflow. *Proceedings of Recent Advances in Natural Language Processing*, (pp. 32-40). Retrieved from https://aclanthology.org/R15-1005.pdf

Braga, D., Niemann, M., Hellingrath, B., & Neto, F. (2018). Survey on Computational Trust and Reputation Models. *ACM Computing Surveys, 51*(5), 1–40. doi:10.1145/3236008

Choetkiertikul, M., Avery, D., Dam, H., Tran, T., & Ghose, A. (2015). Who Will Answer My Question on Stack Overflow? *2015 24th Australasian Software Engineering Conference* (pp. 155-164). IEEE. doi:10.1109/ASWEC.2015.28

Choi, E. (2013). Motivations and Expectations for Asking Questions Within Online Q&A. *Bull. IEEE Tech. Comm. Digit. Libr., 9*. Retrieved from https://www.semanticscholar.org/paper/Motivations-and-Expectations-for-Asking-Questions-Choi/11c9fa83068e17db94187233e74224b146fda197

Cohen, J. (1998). *Statistical Power Analysis for the Behavioral Sciences.* Lawrence Erlbaum Associates.

De la Calzada, G., & Dekhtyar, A. (2010). On measuring the quality of Wikipedia articles. *Proceedings of the 4th workshop on Information Credibility*. Association for Computing Machinery. doi:10.1145/1772938.1772943

Dondio, P., & Longo, L. (2011). Trust-Based Techniques for Collective Intelligence in Social Search Systems. In *Next Generation Data Technologies for Collective*

Computational Intelligence (Vol. 352, pp. 113-135). Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-20344-2_5

Dondio, P., & Longo, L. (2014, August). Computing Trust as a Form of Presumptive Reasoning. *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT). 2*, pp. 274-281. IEEE. doi:10.1109/WI-IAT.2014.108

Dutta, P., & Kumaravel, A. (2016, March 18). A Novel Approach to Trust based Identification of Leaders in Social Networks. *Indian Journal of Science and Technology*. doi:10.17485/ijst/2016/v9i10/85317

Elalfy, D., Gad, W., & Ismail, R. (2018). A hybrid model to predict best answers in question answering communities. *Egyptian Informatics Journal, 19*(1), 21-31. doi:10.1016/j.eij.2017.06.002

Field, A., Miles, J., & Field, Z. (2012). *Discovering Statistics Using R.* SAGE Publications Ltd.

Gambhir, M., Doja, M., & Moinuddin. (2014). Action-based trust computation algorithm for Online Social Network. *2014 Fourth International Conference on Advanced Computing & Communication Technologies*, (pp. 451-458}). doi:10.1109/ACCT.2014.89}

George, D., & Mallery, P. (2002). *SPSS for Windows Step by Step: A Simple Guide and Reference.*

Hamdi, S., Bouzeghoub, A., Gancarski, A., & Yahia, S. (2013). Trust Inference Computation for Online Social Networks. *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, (pp. 210-217). doi:10.1109/TrustCom.2013.240

Lin, J.-W., Lin, T.-C., & Schaedler, P. (2018). Predicting the Best Answers for Questions on Stack Overflow. Retrieved November 28, 2021, from http://castman.net/static/file/paper/cs295-nlp-2018w.pdf

Longo, L., & Barrett, S. (2009, January). A context-aware approach based on self-organizing maps to study web-users' tendencies from their behaviour. *Proceedings of the 1st International Workshop on Context-Aware Middleware and Services: affiliated with the 4th International Conference on Communication System Software and Middleware (COMSWARE 2009). 385*, pp. 12-17. ACM. doi:10.1145/1554233.1554237

Longo, L., Barrett, S., & Dondio, P. (2009, October). Information foraging theory as a form of collective intelligence for social search. *International Conference on Computational Collective Intelligence, 5796*, pp. 63-74. doi:10.1007/978-3-642-04441-0_5

Longo, L., Barrett, S., & Dondio, P. (2009, March). Toward Social Search - From Explicit to Implicit Collaboration to Predict Users' Interests. *WEBIST 2009 - Proceedings of the Fifth International Conference on Web Information Systems and Technologies, Lisbon, Portugal, March 23-26, 2009*, (pp. 693-696). Retrieved September 30, 2022

Longo, L., Dondio, P., & Barrett, S. (2007, September). Temporal Factors to evaluate trustworthiness of virtual identities. *Third International Conference on Security and Privacy in Communications Networks and the Workshops. SecureComm* (pp. 11-19). IEEE. doi:10.1109/seccom.2007.4550300

Longo, L., Dondio, P., & Barrett, S. (2010, January). Enhancing Social Search: A Computational Collective Intelligence Model of Behavioural Traits, Trust and Time. *T. Computational Collective Intelligence, 2*, 46-69. doi:10.1007/978-3-642-17155-0_3

Longo, L., Pierpaolo, D., Riccardo, B., Barrett, S., & Butterfield, A. (2009, November). Enabling Adaptation in Trust Computations. *2009 Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns* (pp. 701-706). IEEE. doi:10.1109/ComputationWorld.2009.70

Marsh, S. (1994, April). *Formalising Trust as a Computational Concept.* Retrieved June 06, 2022, from https://dspace.stir.ac.uk/handle/1893/2010#.YqYgxqjMKUk

Melnikov, A., Lee, J., Rivera, V., Mazzara, M., & Longo, L. (2018, May). Towards Dynamic Interaction-Based Reputation Models. *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)* (pp. 422-428). IEEE. doi:10.1109/AINA.2018.00070

Raban, D., & Harper, F. (2008, December 03). Motivations for Answering Questions Online Abstract. Retrieved from https://www.researchgate.net/publication/241053908_Motivations_for_Answering_Questions_Online_Abstract

Rizzo, L., & Longo, L. (2018, November). Inferential Models of Mental Workload with Defeasible Argumentation and Non-monotonic Fuzzy Reasoning: a

Comparative Study. *Proceedings of the 2nd Workshop on Advances In Argumentation In Artificial Intelligence, 2296*, pp. 11-26. Trento, Italy.

Shearer, C. (2000). The CRISP-DM model: the new blueprint for data mining. *Journal of data warehousing, 5*(4), 13-22.

Singh, M., & Yi Chin, T. (2016). Hybrid Multi-faceted Computational Trust Model for Online Social Network (OSN). *The Science and Information Organization, 7*(6).

Tomáš, T., & Samek, J. (2013, May). Trust evaluation on Facebook using multiple contexts. *CEUR Workshop Proceedings*, 10.

Vilone, G., & Longo, L. (2021, August). Classification of Explainable Artificial Intelligence Methods through Their Output Formats. *Machine Learning and Knowledge Extraction, 3,* 615-661. doi:10.3390/make3030032

West, S. G., Curran, P. J., & Finch, J. F. (1995). *Structural equation models with nonnormal variables: Problems and remedies.* (In R. H. Hoyle (Ed.), Structural equation modeling: Concepts, issues, and applications ed.). Newbery Park, CA: Sage Publications, Inc.

Yashkina, E., Pinigin, A., Lee, J., Mazzara, M., Adekotujo, A., Zubair, A., & Longo, L. (2019, March). Expressing Trust with Temporal Frequency of User Interaction in Online Communities. *33rd International Conference on Advanced Information Networking and Applications* (pp. 1133-1146). Springer, Cham. doi:10.1007/978-3-030-15032-7_95

Zhang, Y., Lo, D., Xia, X., & Sun, J. (2015). Multi-Factor Duplicate Question Detection in Stack Overflow. *Journal of Computer Science and Technology, 30*(5). doi:10.1007/s11390-015-1576-4

6 APPENDIX

6.1 Data Descriptor

Entity	Feature Name	Datatype	Nullable? (YES/NO)?	Length/precision
Comments	Id (PK)	int	NO	10
Comments	PostId	int	NO	10
Comments	Score	int	NO	10
Comments	Text	nvarchar	NO	600
Comments	CreationDate	datetime	NO	3
Comments	UserDisplayName	nvarchar	YES	40
Comments	UserId	int	YES	10
Comments	ContentLicense	varchar	NO	12
Posts	Id (PK)	int	NO	10
Posts	PostTypeId	tinyint	NO	3
Posts	AcceptedAnswerId	int	YES	10
Posts	ParentId	int	YES	10
Posts	CreationDate	datetime	NO	3
Posts	DeletionDate	datetime	YES	3
Posts	Score	int	NO	10
Posts	ViewCount	int	YES	10
Posts	Body	nvarchar	YES	-1
Posts	OwnerUserId	int	YES	10
Posts	OwnerDisplayName	nvarchar	YES	40
Posts	LastEditorUserId	int	YES	10
Posts	LastEditorDisplayName	nvarchar	YES	40
Posts	LastEditDate	datetime	YES	3
Posts	LastActivityDate	datetime	YES	3
Posts	Title	nvarchar	YES	250
Posts	Tags	nvarchar	YES	250
Posts	AnswerCount	int	YES	10

Posts	CommentCount	int	YES	10
Posts	FavoriteCount	int	YES	10
Posts	ClosedDate	datetime	YES	3
Posts	CommunityOwnedDate	datetime	YES	3
Posts	ContentLicense	varchar	NO	12
PostTypes	Id (PK)	tinyint	NO	3
PostTypes	Name	nvarchar	NO	50
Users	Id (PK)	int	NO	10
Users	Reputation	int	NO	10
Users	CreationDate	datetime	NO	3
Users	DisplayName	nvarchar	YES	40
Users	LastAccessDate	datetime	NO	3
Users	WebsiteUrl	nvarchar	YES	200
Users	Location	nvarchar	YES	100
Users	AboutMe	nvarchar	YES	-1
Users	Views	int	NO	10
Users	UpVotes	int	NO	10
Users	DownVotes	int	NO	10
Users	ProfileImageUrl	nvarchar	YES	200
Users	EmailHash	varchar	YES	32
Users	AccountId	int	YES	10
Votes	Id (PK)	int	NO	10
Votes	PostId	int	NO	10
Votes	VoteTypeId	tinyint	NO	3
Votes	UserId	int	YES	10
Votes	CreationDate	datetime	YES	3
Votes	BountyAmount	int	YES	10
VoteTypes	Id (PK)	tinyint	NO	3
VoteTypes	Name	nvarchar	NO	50

6.2 Entity to Table column Mapping

Entity	Attribute	Description	Table Name	Table Column	Measurement Level
Comments	Id	Comment unique id.	SO_COMMENTS	ID	Nominal
Comments	UserId	Community user who submitted the comment. NOTE: Absent if user has been deleted.	SO_COMMENTS	USERID	Nominal
Comments	PostId	Identifying the post record that this comment relates.	SO_COMMENTS	POSTID	Nominal
Comments	CreationDate	Date when the Comment was created.	SO_COMMENTS	CREATIONDATE	Ordinal
Comments	Score	Score of the Comment. Calculated based upon upvotes minus downvotes.	SO_COMMENTS	SCORE	Interval
Posts	Id	Post unique id.	SO_POSTS	ID	Nominal
Posts	CreationDate	Date when the Post was created.	SO_POSTS	CREATIONDATE	Ordinal
Posts	PostTypeId	Id identifying the Post Type.	SO_POSTS	POSTTYPEID	Nominal
Posts	ParentId	The parent SO_POSTS Question record, only present for Answer records i.e., when PostTypeId = 2	SO_POSTS	PARENTID	Nominal
Posts	OwnerUserId	The community user who created Post	SO_POSTS	USERID	Nominal
PostTypes	Id	Post Type unique Id.	SO_POSTTYPES	ID	Nominal
PostTypes	Name	Post Type description.	SO_POSTTYPES	NAME	Nominal
Users	Id	Community User unique id.	SO_USERS	ID	Nominal
Users	CreationDate	Community member registration date.	SO_USERS	CREATIONDATE	Nominal
Users	Reputation	Reputation of Community member.	SO_USERS	REPUTATION	Ordinal
Votes	Id	Vote unique Id	SO_VOTES	ID	Nominal
Posts	OwnerUserId	Identifies the community user who create the the Post that this vote pertains.	SO_VOTES	USERID	Nominal
Votes	VoteTypeId	Id identifying the Vote Type. The foreign key from VoteTypes table	SO_VOTES	VOTETYPEID	Nominal

Votes	PostId	Identifying the post record that this Vote relates.	SO_VOTES	POSTID	Nominal
Votes	CreationDate	Date when the Vote was cast.	SO_VOTES	CREATIONDATE	Ordinal
Votes	BountyAmount	Bounty Amount present only if VoteTypeId in (8,9)	SO_VOTES	BOUNTYAMOUNT	Ratio
VoteTypes	Id	Vote Type unique Id.	SO_VOTETYPES	ID	Nominal
VoteTypes	Name	Vote Type description.	SO_VOTETYPES	NAME	Nominal

6.3 R Code

```
---
title: "MSC Dissertation - R Markup"
author: "Patrick ONeill (D20124902)"
date:   "06-Jun-2022"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo=TRUE, message=FALSE,warning=FALSE)
```

Install Relevant Packages

```{r}
# Specify the relevant packages
needed_packages <- c("ggplot2", "sqldf", "reshape2", "maps",
"stringr","lubridate","dplyr","psych", "scales", "corrgram",
"Hmisc","semTools","effectsize")

# Extract not installed packages
not_installed <- needed_packages[!(needed_packages %in% installed.packages()[ ,
"Package"])]
# Install not installed packages
if(length(not_installed)) install.packages(not_installed)

library(ggplot2)   #For creating histograms with more detail than plot
library(sqldf)
library(reshape2)
library(maps)
library(stringr)
library(lubridate)
library(dplyr)     #For data frame wrangling
library(psych)
library(scales)
library(corrgram)
```
```

```
library(Hmisc)
library(semTools)
library(effectsize) #To calculate effect size for t-test
```

1 Importing Data

```{r}
options(scipen=999)
# Import the downloaded CSV files
df_comments <- read.csv( 'C:\\pj\\Proj\\Comments.csv' , na.strings = c("","NA"), sep=
',' , header=T )
df_posts <- read.csv( 'C:\\pj\\Proj\\Posts.csv' , na.strings = c("","NA"), sep= ',' ,
header=T )
df_posttypes <- read.csv( 'C:\\pj\\Proj\\PostTypes.csv' , na.strings = c("","NA"), sep=
',' , header=T )
df_users <- read.csv( 'C:\\pj\\Proj\\Users.csv' , na.strings = c("","NA"), sep= ',' ,
header=T )
df_votes <- read.csv( 'C:\\pj\\Proj\\Votes.csv' , na.strings = c("","NA"), sep= ',' ,
header=T )
df_votetypes <- read.csv( 'C:\\pj\\Proj\\VoteTypes.csv' , na.strings = c("","NA"), sep=
',' , header=T )

#names(df_comments)
#str(df_comments)

# Convert categorical variables to Factors
df_comments$Id <- as.factor(df_comments$Id)
df_comments$UserId <- as.factor(df_comments$UserId)
df_comments$PostId  <- as.factor(df_comments$PostId)
df_posts$Id <- as.factor(df_posts$Id)
df_posts$PostTypeId <- as.factor(df_posts$PostTypeId)
df_posts$ParentId <- as.factor(df_posts$ParentId)
df_posts$UserId <- as.factor(df_posts$UserId)
df_posttypes$Id <- as.factor(df_posttypes$Id)
df_posttypes$Name <- as.factor(df_posttypes$Name)
df_users$Id <- as.factor(df_users$Id)
df_votes$Id <- as.factor(df_votes$Id)
df_votes$UserId <- as.factor(df_votes$UserId)
df_votes$VotetypeId  <- as.factor(df_votes$VotetypeId)
df_votes$PostId  <- as.factor(df_votes$PostId)
df_votetypes$Id <- as.factor(df_votetypes$Id)
df_votetypes$Name <- as.factor(df_votetypes$Name)

# Convert to date
df_comments$CreationDate <- ymd_hms(df_comments$CreationDate)
df_posts$CreationDate <- ymd_hms(df_posts$CreationDate)
df_users$CreationDate <- ymd_hms(df_users$CreationDate)
df_votes$CreationDate <- ymd_hms(df_votes$CreationDate)
# Convert from POSIXct to Date
df_comments$CreationDate <- as.Date(df_comments$CreationDate)
```

```
df_posts$CreationDate <- as.Date(df_posts$CreationDate)
df_users$CreationDate <- as.Date(df_users$CreationDate)
df_votes$CreationDate <- as.Date(df_votes$CreationDate)
```

2 Function to calculate mode

```{r}
mode <- function(invar) {
temp <- table(invar)
names(temp)[temp == max(temp)]
}
```

3 Missing Values

```{r}
# Check for missing values in df_comments data frame
allMissing <- is.na(df_comments)
counts <- colSums(allMissing)
counts[counts > 0]

# Check for missing values in df_posts data frame
allMissing <- is.na(df_posts)
counts <- colSums(allMissing)
counts[counts > 0]

# Check for missing values in df_posttypes data frame
allMissing <- is.na(df_posttypes)
counts <- colSums(allMissing)
counts[counts > 0]

# Check for missing values in df_users data frame
allMissing <- is.na(df_users)
counts <- colSums(allMissing)
counts[counts > 0]

# Check for missing values in df_votes data frame
allMissing <- is.na(df_votes)
counts <- colSums(allMissing)
counts[counts > 0]

# Check for missing values in df_votetypes data frame
allMissing <- is.na(df_votetypes)
counts <- colSums(allMissing)
counts[counts > 0]

# Replace missing values with 0 for numerical variable
df_votes$BountyAmount[is.na(df_votes$BountyAmount)] <- 0

# Recheck for missing values in df_votes data frame
```

```
allMissing <- is.na(df_votes)
counts <- colSums(allMissing)
counts[counts > 0]
```

4 Feature Engineering

```{r}
```

5 Descriptive Statistics

```{r}
# Produce continuous feature descriptive stats
df_comments$dummy <- 0
data.frame(psych::describe(df_comments, IQR=TRUE, quant=c(.25,.75),omit=T))
df_comments$dummy <- NULL
prop.table(table(df_comments$Score))*100
mode(df_comments$Score)

# Testing for Normality
# Skew
tpskew<-semTools::skew(df_comments$Score)
normskew <- tpskew[1]/tpskew[2]
normskew
# Kurtosis
tpkurt <- semTools::kurtosis(df_comments$Score)
normkurt <- tpkurt[1]/tpkurt[2]
normkurt
# Kolmogorov-Smirnov test for normality
ks.test(df_comments$Score, "pnorm", mean=mean(df_comments$Score),
sd=sd(df_users$Reputationdf_comments$Score))
# shapiro.test(df_comments$Score) used for small samples < 50
# Calculate the percent of Reputation standardized scores outside the acceptable range
of 1.96
zScore <- abs(scale(df_comments$Score))
FSA::perc(as.numeric(zScore), -1.96, "lt")
FSA::perc(as.numeric(zScore), 1.96, "gt")
# Calculate the percent of Reputation standardized scores outside the acceptable range
of 3.29
zScore <- abs(scale(df_comments$Score))
FSA::perc(as.numeric(zScore), -3.29, "lt")
FSA::perc(as.numeric(zScore), 3.29, "gt")

df_users$dummy <- 0
data.frame(psych::describe(df_users, IQR=TRUE, quant=c(.25,.75),omit=T))
df_users$dummy <- NULL
prop.table(table(df_users$Reputation))*100
mode(df_users$Reputation)

# Testing for Normality
```

```
# Skew
tpskew<-semTools::skew(df_users$Reputation)
normskew <- tpskew[1]/tpskew[2]
normskew
# Kurtosis
tpkurt <- semTools::kurtosis(df_users$Reputation)
normkurt <- tpkurt[1]/tpkurt[2]
normkurt
# Kolmogorov-Smirnov test for normality
ks.test(df_users$Reputation, "pnorm", mean=mean(df_users$Reputation),
sd=sd(df_users$Reputation))
# shapiro.test(df_users$Reputation) used for small samples < 50
# Calculate the percent of Reputation standardized scores outside the acceptable range
of 1.96
zReputation <- abs(scale(df_users$Reputation))
FSA::perc(as.numeric(zReputation), -1.96, "lt")
FSA::perc(as.numeric(zReputation), 1.96, "gt")
# Calculate the percent of Reputation standardized scores outside the acceptable range
of 3.29
zReputation <- abs(scale(df_users$Reputation))
FSA::perc(as.numeric(zReputation), -3.29, "lt")
FSA::perc(as.numeric(zReputation), 3.29, "gt")

df_votes$dummy <- 0
data.frame(psych::describe(df_votes, IQR=TRUE, quant=c(.25,.75),omit=T))
df_votes$dummy <- NULL
mode(df_votes$BountyAmount)

# Testing for Normality
# Skew
tpskew<-semTools::skew(df_votes$BountyAmount)
normskew <- tpskew[1]/tpskew[2]
normskew
# Kurtosis
tpkurt <- semTools::kurtosis(df_votes$BountyAmount)
normkurt <- tpkurt[1]/tpkurt[2]
normkurt
# Kolmogorov-Smirnov test for normality
ks.test(df_votes$BountyAmount, "pnorm", mean=mean(df_votes$BountyAmount),
sd=sd(df_votes$BountyAmount))
# shapiro.test(df_votes$BountyAmount) used for small samples < 50
# Calculate the percent of Reputation standardized scores outside the acceptable range
of 1.96
zVotes <- abs(scale(df_votes$BountyAmount))
FSA::perc(as.numeric(zVotes), -1.96, "lt")
FSA::perc(as.numeric(zVotes), 1.96, "gt")
# Calculate the percent of Reputation standardized scores outside the acceptable range
of 3.29
zVotes <- abs(scale(df_votes$BountyAmount))
FSA::perc(as.numeric(zVotes), -3.29, "lt")
FSA::perc(as.numeric(zVotes), 3.29, "gt")
```

```
# Produce categorical feature stats
catvars <- c("Id","UserId","PostId","CreationDate")
summary(df_comments[,catvars])

catvars <- c("Id","PostTypeId","ParentId","UserId", "CreationDate")
summary(df_posts[,catvars])
prop.table(table(df_posts$PostTypeId))*100

catvars <- c("Id","Name")
summary(df_posttypes[,catvars])

catvars <- c("Id", "CreationDate")
summary(df_users[,catvars])

catvars <- c("Id","UserId", "VotetypeId","PostId","CreationDate")
summary(df_votes[,catvars])
prop.table(table(df_votes$VotetypeId))*100

catvars <- c("Id","Name")
summary(df_votetypes[,catvars])
```

6 Exploratory Visualizations

```{r}
# Continuous feature visualizations

# Figure 1 - Score histogram
plt1 <- sqldf("select Score as Score, Id as Id from df_comments")
ggplot(data=plt1, aes(x=Score)) +
  labs(title="Comment Score (Histogram)", x = "Comment Score") +
    geom_histogram(binwidth=1, colour="black", aes(y = ..count..), fill = "steelblue2")

# Figure 2 - Reputation histogram
plt2 <- sqldf("select Reputation as Reputation, Id as Id from df_users")
ggplot(data=plt2, aes(x=Reputation)) +
  labs(title="User Reputation (Histogram)", x = "User Reputation") +
    geom_histogram(binwidth=10000, colour="black", aes(y = ..count..), fill =
"steelblue2")

# Figure 3 - Score density histogram
ggplot(data = plt1, aes(x = Score)) +
  labs(title="Comment Score (Density Histogram)", x = "Comment Score") +
    geom_histogram(binwidth=1, colour="black", aes(y = ..density..), fill =
"steelblue2") +
    stat_function(fun=dnorm, color="red",
                  args=list(mean=mean(plt1$Score,na.rm=TRUE),
                            sd=sd(plt1$Score,na.rm=TRUE)))
```

```
# Figure 4 - Reputation histogram
ggplot(data = plt2, aes(x = Reputation)) +
  labs(title="User Reputation (Density Histogram)", x = "User Reputation") +
    geom_histogram(binwidth=10000, colour="black", aes(y = ..density..), fill =
"steelblue2") +
    stat_function(fun=dnorm, color="red",
                  args=list(mean=mean(plt2$Reputation,na.rm=TRUE),
                            sd=sd(plt2$Reputation,na.rm=TRUE)))

# Figure 5 - Score boxplot
ggplot(plt1, aes(x = Score)) +
  labs(title="Comment Score (Boxplot)", x="Comment Score") + geom_boxplot()

# Figure 6 - Reputation boxplot
ggplot(plt2, aes(x = Reputation)) +
  labs(title="User Reputation (Boxplot)", x="User Reputation") + geom_boxplot()

# Categorical Feature Visualizations

# Figure 9 - Comment Volume by Year Added
plt3 <- sqldf("SELECT strftime('%Y', CreationDate * 3600 * 24, 'unixepoch') as
added_year,
                      count(Id) as count
              FROM    df_comments
                    GROUP BY strftime('%Y', CreationDate * 3600 * 24, 'unixepoch')")

ggplot(data=plt3, aes(x = added_year, y = count)) +
  labs(title='Comment Volume by Year Added', x='Year Added', y = 'Count') +
    geom_bar(stat="identity", fill="steelblue2", width = 0.5) +
      geom_text(aes(label = count)) +
        theme(axis.text.x = element_text(angle = 90))

# Figure 7 - Post Type Frequency Bar chart
plt4 <- sqldf("select t2.Name as PostType, count(t1.Id) as count from df_posts t1
               INNER JOIN df_posttypes t2 on t1.PostTypeId = t2.Id
               group by t2.Name")
ggplot(data=plt4, aes(x = reorder(PostType, - count), y = count)) +
  labs(title='Post Type (Frequency Bar Chart)', x='Post Type', y = 'Count') +
    geom_bar(stat="identity", fill="steelblue2", width = 0.5) +
      geom_text(aes(label = count))

# Figure 9 - Post Volume by Year Added
plt5 <- sqldf("SELECT strftime('%Y', CreationDate * 3600 * 24, 'unixepoch') as
added_year,
                      count(Id) as count
              FROM    df_posts
                    GROUP BY strftime('%Y', CreationDate * 3600 * 24, 'unixepoch')")

ggplot(data=plt5, aes(x = added_year, y = count)) +
  labs(title='Post Volume by Year Added', x='Year Added', y = 'Count') +
    geom_bar(stat="identity", fill="steelblue2", width = 0.5) +
```

```
      geom_text(aes(label = count)) +
        theme(axis.text.x = element_text(angle = 90))

# Figure 9 - User Volume by Year Added
plt6 <- sqldf("SELECT strftime('%Y', CreationDate * 3600 * 24, 'unixepoch') as
added_year,
                      count(Id) as count
              FROM    df_users
                    GROUP BY strftime('%Y', CreationDate * 3600 * 24, 'unixepoch')")

ggplot(data=plt6, aes(x = added_year, y = count)) +
  labs(title='User Volume by Year Added', x='Year Added', y = 'Count') +
    geom_bar(stat="identity", fill="steelblue2", width = 0.5) +
      geom_text(aes(label = count)) +
        theme(axis.text.x = element_text(angle = 90))

# Figure 8 - Vote Type Frequency Bar chart
plt7 <- sqldf("select t2.Name as Votetype, count(t1.Id) as count from df_votes t1
               INNER JOIN df_votetypes t2 on t1.VoteTypeId = t2.Id
               group by t2.Name")
ggplot(data=plt7, aes(x = reorder(Votetype, - count), y = count)) +
  labs(title='Vote Type (Frequency Bar Chart)', x='Vote Type', y = 'Count') +
    geom_bar(stat="identity", fill="steelblue2", width = 0.5) +
      geom_text(aes(label = count)) +
        theme(axis.text.x = element_text(angle = 90))

```

7 Correlation

```{r}
plt8 <- sqldf("SELECT t1.Reputation as Reputation, count(t2.Id) / count(distinct t1.Id)
as Volume
              FROM    df_users t1
                              INNER JOIN df_comments t2 on t1.Id = t2.UserId
                    GROUP BY t1.Reputation")

   ggplot(data = plt8, aes(x = Reputation, y = Volume)) +
     geom_point() +
       geom_smooth(formula = y ~ x, method = "lm", colour = "Red", se = F) +
        labs(title='Comment Volume v Reputation', x = 'Reputation', y = 'Volume')

# Average Post Volume v Reputation
plt9 <- sqldf("SELECT t1.Reputation as Reputation, count(t2.Id) / count(distinct t1.Id)
as Volume
              FROM    df_users t1
                              INNER JOIN df_posts t2 on t1.Id = t2.UserId
                    GROUP BY t1.Reputation")

   ggplot(data = plt9, aes(x = Reputation, y = Volume)) +
     geom_point() +
```

```
       geom_smooth(formula = y ~ x, method = "lm", colour = "Red", se = F) +
        labs(title='Post Volume v Reputation', x = 'Reputation', y = 'Volume')

# Average Vote Volume v Reputation
plt10 <- sqldf("SELECT t1.Reputation as Reputation, count(t2.Id) / count(distinct
t1.Id) as Volume
                FROM    df_users t1
                                INNER JOIN df_votes t2 on t1.Id = t2.UserId
                     GROUP BY t1.Reputation")

   ggplot(data = plt10, aes(x = Reputation, y = Volume)) +
     geom_point() +
       geom_smooth(formula = y ~ x, method = "lm", colour = "Red", se = F) +
        labs(title='Vote Volume v Reputation', x = 'Reputation', y = 'Volume')

# Join up the data to produce Correlation matrix
plt11 <- sqldf("SELECT t1.Reputation as Reputation,
                       t1.Volume as CommentVolume,
                       t2.Volume as PostVolume,
                       t3.Volume as VoteVolume
                FROM    plt8 t1
                                INNER JOIN plt9 t2 on t1.Reputation = t2.Reputation
                                INNER JOIN plt10 t3 on t1.Reputation = t3.Reputation
                     GROUP BY t1.Reputation")

raqData <- plt11[,c(1,2,3,4)]
raqMatrix<-cor(raqData)
round(raqMatrix, 2)
corrplot::corrplot(raqMatrix, method="number")
``````

---
title: "MSC Dissertation - R Markup"
author: "Patrick ONeill (D20124902)"
date: "06-Jun-2022"
output: html_document

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo=TRUE, message=FALSE,warning=FALSE)
```

Install Relevant Packages

```{r}
# Specify the relevant packages
needed_packages <- c("ggplot2", "sqldf", "reshape2", "maps",
"stringr","lubridate","dplyr","psych", "scales", "corrgram",
"Hmisc","semTools","effectsize","rstatix","tidyverse","ggpubr")

# Extract not installed packages
```

```
not_installed <- needed_packages[!(needed_packages %in% installed.packages()[ ,
"Package"])]
# Install not installed packages
if(length(not_installed)) install.packages(not_installed)

library(ggplot2)   #For creating histograms with more detail than plot
library(sqldf)
library(reshape2)
library(maps)
library(stringr)
library(lubridate)
library(dplyr)     #For data frame wrangling
library(psych)
library(scales)
library(corrgram)
library(Hmisc)
library(semTools)
library(effectsize) #To calculate effect size for t-test
library(rstatix)
library(tidyverse)
library(ggpubr)
```

1 Importing Data

```{r}
options(scipen=999)
```

7 Correlation

```{r}
# Import the downloaded CSV files
# select * from SO_HIST_TRUST_STACKOVERFLOW_V
# ORDER BY 1,2
# select * from SO_HIST_TRUST_STACKOVERFLOW_T_V
# ORDER BY 1,2
# select * from SO_HIST_TRUSTMAXPERDAY_DIBRM_V
# ORDER BY 1,2
# select * from SO_HIST_TRUSTMAXPERDAY_DIBRM_T_V
# ORDER BY 1,2

df_users <- read.csv( 'C:\\pj\\Proj\\Users.csv' , na.strings = c("","NA"), sep= ',' ,
header=T )
df_rules <- read.csv( 'C:\\pj\\Proj\\RulesBased.csv' , na.strings = c("","NA"), sep=
',' , header=T )
df_rulest <- read.csv( 'C:\\pj\\Proj\\RulesTBased.csv' , na.strings = c("","NA"), sep=
',' , header=T )
df_dibrm <- read.csv( 'C:\\pj\\Proj\\DIBRMBased.csv' , na.strings = c("","NA"), sep=
',' , header=T )
```

```
df_dibrmt <- read.csv( 'C:\\pj\\Proj\\DIBRMTBased.csv' , na.strings = c("","NA"), sep=
',' , header=T )

count(df_rules)
count(df_rulest)
count(df_dibrm)
count(df_dibrmt)
df_users$Id <- as.factor(df_users$Id)

# Convert to date
df_users$CreationDate <- ymd_hms(df_users$CreationDate)
# Convert from POSIXct to Date
df_users$CreationDate <- as.Date(df_users$CreationDate)
df_rules$CALDATE <-  as.Date(df_rules$CALDATE, format =  "%d/%m/%Y")
df_rulest$CALDATE <- as.Date(df_rules$CALDATE, format =  "%d/%m/%Y")
df_dibrm$CALDATE <-  as.Date(df_dibrm$CALDATE, format =  "%d/%m/%Y")
df_dibrmt$CALDATE <- as.Date(df_dibrmt$CALDATE, format = "%d/%m/%Y")

# Convert categorical variables to Factors
df_rulest$TOPIC <- as.factor(df_rulest$TOPIC)
df_dibrmt$TOPIC <- as.factor(df_dibrmt$TOPIC)

str(df_rules)
str(df_rulest)
str(df_dibrm)
str(df_dibrmt)

```

2 Function to calculate mode

```{r}
mode <- function(invar) {
temp <- table(invar)
names(temp)[temp == max(temp)]
}
```

```{r}
# Plot DIBRM for userid 300 for first 300 days
pltred <- sqldf("SELECT DAYNUM,TRUST as reputation
       FROM df_dibrm t1
            WHERE USERID=300
            AND daynum <= 1500
       ORDER BY DAYNUM")

pltblue <- sqldf("SELECT DAYNUM,TRUST as reputation

       FROM df_dibrm t1
            WHERE USERID=235
            AND daynum <= 1500
```

```
       ORDER BY DAYNUM")

ggplot() +
          geom_line(data = pltred , aes(x = DAYNUM, y = reputation), color = "red") +
          geom_line(data = pltblue , aes(x = DAYNUM, y = reputation), color = "blue")
+
          labs(title='Dynamic Reputation for UserId (235, 300)', x = 'Days', y =
'Dynamic Reputation')


# Plot DIBRM Topic for userid 300 for first 1500 days for top 4 topics
plt15t5 <- sqldf("SELECT USERID, TOPIC, COUNT(t1.DAYNUM) as count
                  FROM df_dibrmt t1
                       WHERE t1.USERID=300
                       AND t1.daynum <= 1500
                  GROUP BY USERID, TOPIC
                  ORDER BY count desc limit 4")

plt15t <- sqldf("SELECT t1.DAYNUM,t1.TOPIC,t1.TRUST as reputation
       FROM df_dibrmt t1
       INNER JOIN plt15t5 t2 on t1.USERID = t2.USERID
       AND T1.TOPIC=t2.TOPIC
            WHERE t1.USERID=300
            AND t1.daynum <= 1500
       ORDER BY t1.TOPIC, t1.DAYNUM")

ggplot(data = plt15t, aes(x = DAYNUM, y = reputation)) +
   geom_line() +
        labs(title='Dynamic Reputation for UserId (300)', x = 'Days', y = 'Dynamic
Reputation') +
            facet_wrap(. ~ TOPIC)


```


```{r}
## SO Rules-based v Modelled Rule-based
# Join up the data actual v synthesised
# NOTE: the max of the CUMTRUST would work here also if they didn;t have negative trust
df_actVsyn <- sqldf("SELECT t1.Id,
                           t1.Reputation as actrep,
                                                                SUM(t2.TRUST) as
synrep
                    FROM df_users t1
                       INNER JOIN df_rules t2 on (t1.id = t2.USERID)
                    GROUP BY t1.Id, t1.Reputation")

# Descriptive stats of actual
data.frame(psych::describe(df_actVsyn, IQR=TRUE, quant=c(.25,.75),omit=T))
mode(df_actVsyn$actrep)
```

```
mode(df_actVsyn$synrep)
# Test for normalization of actual

# Plot Histogram of actual
ggplot(data=df_actVsyn, aes(x=actrep)) +
  labs(title="Stock Overflow Reputation (Histogram)", x = "Reputation") +
    geom_histogram(binwidth=10000, colour="black", aes(y = ..count..), fill =
"steelblue2")

ggplot(data = df_actVsyn, aes(x = actrep)) +
  labs(title="Stack Overflow Reputation (Density Histogram)", x = "Reputation") +
    geom_histogram(binwidth=10000, colour="black", aes(y = ..density..), fill =
"steelblue2") +
    stat_function(fun=dnorm, color="red",
                  args=list(mean=mean(df_actVsyn$actrep,na.rm=TRUE),
                            sd=sd(df_actVsyn$actrep,na.rm=TRUE)))


qqnorm(df_actVsyn$actrep)
qqline(df_actVsyn$actrep, col=2) #show a line on the plot

# Skew
tpskew<-semTools::skew(df_actVsyn$actrep)
normskew <- tpskew[1]/tpskew[2]
normskew
# Kurtosis
tpkurt <- semTools::kurtosis(df_actVsyn$actrep)
normkurt <- tpkurt[1]/tpkurt[2]
normkurt
# Kolmogorov-Smirnov test for normality
ks.test(df_actVsyn$actrep,"pnorm",exact=FALSE,  mean=mean(df_actVsyn$actrep),
sd=sd(df_actVsyn$actrep))

# Calculate the percent of Reputation standardized scores outside the acceptable range
of 1.96
zReputation <- abs(scale(df_actVsyn$actrep))
FSA::perc(as.numeric(zReputation), -1.96, "lt")
FSA::perc(as.numeric(zReputation), 1.96, "gt")
# Calculate the percent of Reputation standardized scores outside the acceptable range
of 3.29
zReputation <- abs(scale(df_actVsyn$actrep))
FSA::perc(as.numeric(zReputation), -3.29, "lt")
FSA::perc(as.numeric(zReputation), 3.29, "gt")

# Test for normalization of Synthetic SO Reputation
# Plot Histogram
ggplot(data=df_actVsyn, aes(x=synrep)) +
  labs(title="Rule-Based Reputation (Histogram)", x = "Reputation") +
    geom_histogram(binwidth=10000, colour="black", aes(y = ..count..), fill =
"steelblue2")
```

```
ggplot(data = df_actVsyn, aes(x = synrep)) +
  labs(title="Rule-Based (Density Histogram)", x = "Reputation") +
    geom_histogram(binwidth=10000, colour="black", aes(y = ..density..), fill =
"steelblue2") +
    stat_function(fun=dnorm, color="red",
                  args=list(mean=mean(df_actVsyn$synrep,na.rm=TRUE),
                            sd=sd(df_actVsyn$synrep,na.rm=TRUE)))

qqnorm(df_actVsyn$synrep)
qqline(df_actVsyn$synrep, col=2) #show a line on the plot

# Skew
tpskew<-semTools::skew(df_actVsyn$synrep)
normskew <- tpskew[1]/tpskew[2]
normskew
# Kurtosis
tpkurt <- semTools::kurtosis(df_actVsyn$synrep)
normkurt <- tpkurt[1]/tpkurt[2]
normkurt
# Kolmogorov-Smirnov test for normality
ks.test(df_actVsyn$synrep,"pnorm",exact=FALSE,  mean=mean(df_actVsyn$synrep),
sd=sd(df_actVsyn$synrep))

zReputation <- abs(scale(df_actVsyn$synrep))
FSA::perc(as.numeric(zReputation), -1.96, "lt")
FSA::perc(as.numeric(zReputation), 1.96, "gt")
# Calculate the percent of Reputation standardized scores outside the acceptable range
of 3.29
zReputation <- abs(scale(df_actVsyn$synrep))
FSA::perc(as.numeric(zReputation), -3.29, "lt")
FSA::perc(as.numeric(zReputation), 3.29, "gt")
# dibrm historical reputation not normal

# Scatterplot of variables
 ggplot(data = df_actVsyn, aes(x = actrep, y = synrep)) +
     geom_point() +
       geom_smooth(formula = y ~ x, method = "lm", colour = "Red", se = F) +
        labs(title='Stack Overflow v Rule-Based Reputation', x = 'Stack Overflow
Reputation', y = 'Rule-Based Reputation')

#Pearson Correlation
stats::cor.test(df_actVsyn$actrep, df_actVsyn$synrep, method='pearson')
res <- stats::cor.test(df_actVsyn$actrep, df_actVsyn$synrep, method='pearson')
#Calculate Cohen's d
effcd=round((2*res$statistic)/sqrt(res$parameter),2)
effcd
#Using function from effectsize package
effectsize::t_to_d(t = res$statistic, res$parameter)

# paired t-test
stats::t.test(df_actVsyn$synrep,df_actVsyn$actrep, paired = TRUE)
```

```
res <- stats::t.test(df_actVsyn$synrep,df_actVsyn$actrep, paired = TRUE)
effcd=round((2*res$statistic)/sqrt(res$parameter),2)
effcd
#Using function from effectsize package
effectsize::t_to_d(t = res$statistic, res$parameter)
```

```{r}
##  Rules-based v DIBRM Model
#df_actVsyn <- sqldf("SELECT t1.USERID,
#                            t1.DAYNUM,
#                                                sum(t1.CUMTRUST) as actrep,
#                                                sum(t2.CUMTRUST) as synrep
#                                    FROM df_rules t1
#                                    INNER JOIN df_dibrm t2 on (t1.userid = t2.USERID
#                                    AND t1.daynum = t2.daynum)
#                                    GROUP BY t1.USERID, t1.DAYNUM")

# Compare the rules based value on that day (which is a cum value of the + and - of
each day)
# which the model max trust level on that day
df_actVsyn <- sqldf("SELECT t1.USERID,
                            t1.DAYNUM,
                                                                t1.CUMTRUST as
actrep,
                                                                t2.TRUST as synrep
                                                    FROM df_rules t1
                                                    INNER JOIN df_dibrm t2 on (t1.userid =
t2.USERID
                                                    AND t1.daynum = t2.daynum)")

# Descriptive stats of actual
data.frame(psych::describe(df_actVsyn, IQR=TRUE, quant=c(.25,.75),omit=T))
mode(df_actVsyn$actrep)
mode(df_actVsyn$synrep)
# Test for normalization of actual

# Plot Histogram of actual
ggplot(data=df_actVsyn, aes(x=actrep)) +
  labs(title="Rules-Based Reputation (Histogram)", x = "Reputation") +
    geom_histogram(binwidth=10000, colour="black", aes(y = ..count..), fill =
"steelblue2")

ggplot(data = df_actVsyn, aes(x = actrep)) +
  labs(title="Rules-Based Reputation (Density Histogram)", x = "Reputation") +
    geom_histogram(binwidth=10000, colour="black", aes(y = ..density..), fill =
"steelblue2") +
    stat_function(fun=dnorm, color="red",
                  args=list(mean=mean(df_actVsyn$actrep,na.rm=TRUE),
                            sd=sd(df_actVsyn$actrep,na.rm=TRUE)))
```

```
qqnorm(df_actVsyn$actrep)
qqline(df_actVsyn$actrep, col=2) #show a line on the plot

# Skew
tpskew<-semTools::skew(df_actVsyn$actrep)
normskew <- tpskew[1]/tpskew[2]
normskew
# Kurtosis
tpkurt <- semTools::kurtosis(df_actVsyn$actrep)
normkurt <- tpkurt[1]/tpkurt[2]
normkurt
# Kolmogorov-Smirnov test for normality
ks.test(df_actVsyn$actrep,"pnorm",exact=FALSE,  mean=mean(df_actVsyn$actrep),
sd=sd(df_actVsyn$actrep))

# Calculate the percent of Reputation standardized scores outside the acceptable range
of 1.96
zReputation <- abs(scale(df_actVsyn$actrep))
FSA::perc(as.numeric(zReputation), -1.96, "lt")
FSA::perc(as.numeric(zReputation), 1.96, "gt")
# Calculate the percent of Reputation standardized scores outside the acceptable range
of 3.29
zReputation <- abs(scale(df_actVsyn$actrep))
FSA::perc(as.numeric(zReputation), -3.29, "lt")
FSA::perc(as.numeric(zReputation), 3.29, "gt")

# Test for normalization of Synthetic SO Reputation
# Plot Histogram
ggplot(data=df_actVsyn, aes(x=synrep)) +
  labs(title="DIBRM Reputation (Histogram)", x = "Reputation") +
    geom_histogram(binwidth=100, colour="black", aes(y = ..count..), fill =
"steelblue2")

ggplot(data = df_actVsyn, aes(x = synrep)) +
  labs(title="DIBRM Reputation (Density Histogram)", x = "Reputation") +
    geom_histogram(binwidth=100, colour="black", aes(y = ..density..), fill =
"steelblue2") +
    stat_function(fun=dnorm, color="red",
                  args=list(mean=mean(df_actVsyn$synrep,na.rm=TRUE),
                            sd=sd(df_actVsyn$synrep,na.rm=TRUE)))

qqnorm(df_actVsyn$synrep)
qqline(df_actVsyn$synrep, col=2) #show a line on the plot

# Skew
tpskew<-semTools::skew(df_actVsyn$synrep)
normskew <- tpskew[1]/tpskew[2]
normskew
# Kurtosis
```

```
tpkurt <- semTools::kurtosis(df_actVsyn$synrep)
normkurt <- tpkurt[1]/tpkurt[2]
normkurt
# Kolmogorov-Smirnov test for normality
ks.test(df_actVsyn$synrep,"pnorm",exact=FALSE,  mean=mean(df_actVsyn$synrep),
sd=sd(df_actVsyn$synrep))

zReputation <- abs(scale(df_actVsyn$synrep))
FSA::perc(as.numeric(zReputation), -1.96, "lt")
FSA::perc(as.numeric(zReputation), 1.96, "gt")
# Calculate the percent of Reputation standardized scores outside the acceptable range
of 3.29
zReputation <- abs(scale(df_actVsyn$synrep))
FSA::perc(as.numeric(zReputation), -3.29, "lt")
FSA::perc(as.numeric(zReputation), 3.29, "gt")
# dibrm historical reputation not normal

# Scatterplot of variables
 ggplot(data = df_actVsyn, aes(x = actrep, y = synrep)) +
     geom_point() +
       geom_smooth(formula = y ~ x, method = "lm", colour = "Red", se = F) +
        labs(title='Rule-Based v DIBRM Reputation', x = 'Rule-Based Reputation', y =
'DIBRM Reputation')

#Pearson Correlation
stats::cor.test(df_actVsyn$actrep, df_actVsyn$synrep, method='pearson')
res <- stats::cor.test(df_actVsyn$actrep, df_actVsyn$synrep, method='pearson')
stats::cor.test(df_actVsyn$actrep, df_actVsyn$synrep, exact=FALSE,method='spearman')
#Calculate Cohen's d
effcd=round((2*res$statistic)/sqrt(res$parameter),2)
effcd
#Using function from effectsize package
effectsize::t_to_d(t = res$statistic, res$parameter)

# not normally distribute hence paired wilcox test
#stats::t.test(df_actVsyn$synrep,df_actVsyn$actrep, paired = TRUE)
#res <- stats::t.test(df_actVsyn$synrep,df_actVsyn$actrep, paired = TRUE)
#wilcox.test(df_actVsyn$synrep, df_actVsyn$actrep, paired = TRUE)
#res <- wilcox.test(df_actVsyn$synrep, df_actVsyn$actrep, paired = TRUE)
#res$p.value

#coin::wilcoxsign_test(df_actVsyn$synrep, df_actVsyn$actrep, paired = TRUE)

#reff<-rstatix::wilcox_effsize(synrep, actrep, data=df_actVsyn, paired=TRUE)
```

```{r}
##  Rules-based v DIBRM Topic Model
#df_actVsyn <- sqldf("SELECT t1.USERID,
#                            t1.topic,
#                            t1.DAYNUM,
```

```
#                                                 sum(t1.CUMTRUST) as actrep,
#                                                 sum(t2.CUMTRUST) as synrep
#                                       FROM df_rulest t1
#                                       INNER JOIN df_dibrmt t2 on (t1.userid =
t2.USERID
#                                       AND t1.daynum = t2.daynum
#                                       AND t1.topic=t2.topic)
#                                       GROUP BY t1.USERID, t1.topic, t1.DAYNUM")

# Compare the rules based topic primary topic value on that day (which is a cum value
of the + and - of each day)
# which the model max trust level on that day

df_actVsyn <- sqldf("SELECT t1.USERID,
                            t1.topic,
                            t1.DAYNUM,
                                                                    t1.CUMTRUST as
actrep,
                                                                    t2.TRUST as synrep
                                                    FROM df_rulest t1
                                                    INNER JOIN df_dibrmt t2 on (t1.userid
= t2.USERID
                                                    AND t1.daynum = t2.daynum
                                                    AND t1.topic=t2.topic)")


# Descriptive stats of actual
data.frame(psych::describe(df_actVsyn, IQR=TRUE, quant=c(.25,.75),omit=T))
mode(df_actVsyn$actrep)
mode(df_actVsyn$synrep)
# Test for normalization of actual

# Plot Histogram of actual
ggplot(data=df_actVsyn, aes(x=actrep)) +
  labs(title="Rules-Based Topic Reputation (Histogram)", x = "Reputation") +
    geom_histogram(binwidth=10000, colour="black", aes(y = ..count..), fill =
"steelblue2")

ggplot(data = df_actVsyn, aes(x = actrep)) +
  labs(title="Rules-Based Topic Reputation (Density Histogram)", x = "Reputation") +
    geom_histogram(binwidth=10000, colour="black", aes(y = ..density..), fill =
"steelblue2") +
    stat_function(fun=dnorm, color="red",
                  args=list(mean=mean(df_actVsyn$actrep,na.rm=TRUE),
                            sd=sd(df_actVsyn$actrep,na.rm=TRUE)))


qqnorm(df_actVsyn$actrep)
qqline(df_actVsyn$actrep, col=2) #show a line on the plot

# Skew
```

```
tpskew<-semTools::skew(df_actVsyn$actrep)
normskew <- tpskew[1]/tpskew[2]
normskew
# Kurtosis
tpkurt <- semTools::kurtosis(df_actVsyn$actrep)
normkurt <- tpkurt[1]/tpkurt[2]
normkurt
# Kolmogorov-Smirnov test for normality
ks.test(df_actVsyn$actrep,"pnorm",exact=FALSE,  mean=mean(df_actVsyn$actrep),
sd=sd(df_actVsyn$actrep))

# Calculate the percent of Reputation standardized scores outside the acceptable range
of 1.96
zReputation <- abs(scale(df_actVsyn$actrep))
FSA::perc(as.numeric(zReputation), -1.96, "lt")
FSA::perc(as.numeric(zReputation), 1.96, "gt")
# Calculate the percent of Reputation standardized scores outside the acceptable range
of 3.29
zReputation <- abs(scale(df_actVsyn$actrep))
FSA::perc(as.numeric(zReputation), -3.29, "lt")
FSA::perc(as.numeric(zReputation), 3.29, "gt")

# Test for normalization of Synthetic SO Reputation
# Plot Histogram
ggplot(data=df_actVsyn, aes(x=synrep)) +
  labs(title="DIBRM Reputation (Histogram)", x = "Reputation") +
    geom_histogram(binwidth=100, colour="black", aes(y = ..count..), fill =
"steelblue2")

ggplot(data = df_actVsyn, aes(x = synrep)) +
  labs(title="DIBRM Reputation (Density Histogram)", x = "Reputation") +
    geom_histogram(binwidth=100, colour="black", aes(y = ..density..), fill =
"steelblue2") +
    stat_function(fun=dnorm, color="red",
                  args=list(mean=mean(df_actVsyn$synrep,na.rm=TRUE),
                            sd=sd(df_actVsyn$synrep,na.rm=TRUE)))

qqnorm(df_actVsyn$synrep)
qqline(df_actVsyn$synrep, col=2) #show a line on the plot

# Skew
tpskew<-semTools::skew(df_actVsyn$synrep)
normskew <- tpskew[1]/tpskew[2]
normskew
# Kurtosis
tpkurt <- semTools::kurtosis(df_actVsyn$synrep)
normkurt <- tpkurt[1]/tpkurt[2]
normkurt
# Kolmogorov-Smirnov test for normality
ks.test(df_actVsyn$synrep,"pnorm",exact=FALSE,  mean=mean(df_actVsyn$synrep),
sd=sd(df_actVsyn$synrep))
```

```
zReputation <- abs(scale(df_actVsyn$synrep))
FSA::perc(as.numeric(zReputation), -1.96, "lt")
FSA::perc(as.numeric(zReputation), 1.96, "gt")
# Calculate the percent of Reputation standardized scores outside the acceptable range
of 3.29
zReputation <- abs(scale(df_actVsyn$synrep))
FSA::perc(as.numeric(zReputation), -3.29, "lt")
FSA::perc(as.numeric(zReputation), 3.29, "gt")
# dibrm historical reputation not normal

# Scatterplot of variables
 ggplot(data = df_actVsyn, aes(x = actrep, y = synrep)) +
     geom_point() +
       geom_smooth(formula = y ~ x, method = "lm", colour = "Red", se = F) +
        labs(title='Rule-Based v DIBRM Topic Reputation', x = 'Rule-Based Topic
Reputation', y = 'DIBRM Topic Reputation')

#Pearson Correlation
stats::cor.test(df_actVsyn$actrep, df_actVsyn$synrep, method='pearson')
res <- stats::cor.test(df_actVsyn$actrep, df_actVsyn$synrep, method='pearson')
stats::cor.test(df_actVsyn$actrep, df_actVsyn$synrep, exact=FALSE,method='spearman')
#Calculate Cohen's d
effcd=round((2*res$statistic)/sqrt(res$parameter),2)
effcd
#Using function from effectsize package
effectsize::t_to_d(t = res$statistic, res$parameter)
```

6.4 Database Schema

6.4.1 Table DDL

```
ALTER TABLE SO_CALENDAR
 DROP PRIMARY KEY CASCADE;

DROP TABLE SO_CALENDAR CASCADE CONSTRAINTS;

CREATE TABLE SO_CALENDAR
(
  CALDATE  DATE                                 NOT NULL
);

ALTER TABLE SO_COMMENTS
 DROP PRIMARY KEY CASCADE;

DROP TABLE SO_COMMENTS CASCADE CONSTRAINTS;

CREATE TABLE SO_COMMENTS
```

```
(
  ID            INTEGER                         NOT NULL,
  USERID        INTEGER,
  POSTID        INTEGER,
  CREATIONDATE  DATE,
  SCORE         INTEGER,
  TAGS          VARCHAR2(1000 CHAR)
);

ALTER TABLE SO_HIST_TRUST_DIBRM
 DROP PRIMARY KEY CASCADE;

DROP TABLE SO_HIST_TRUST_DIBRM CASCADE CONSTRAINTS;

CREATE TABLE SO_HIST_TRUST_DIBRM
(
  USERID          INTEGER                       NOT NULL,
  INDEXN          INTEGER                       NOT NULL,
  CALDATE         DATE                          NOT NULL,
  ICUMATN         NUMBER,
  IBASATN         NUMBER,
  ALPHA           NUMBER,
  ACTATN          INTEGER,
  IATN            NUMBER,
  DELTAATN        NUMBER,
  TIMEATN         DATE,
  TIMEATNMINUS1   DATE,
  ACTPERIOD       NUMBER,
  TRUSTATN        NUMBER,
  TRUSTATNMINUS1  NUMBER,
  BETA            NUMBER
);

ALTER TABLE SO_HIST_TRUST_DIBRM_T
 DROP PRIMARY KEY CASCADE;

DROP TABLE SO_HIST_TRUST_DIBRM_T CASCADE CONSTRAINTS;

CREATE TABLE SO_HIST_TRUST_DIBRM_T
(
  USERID          INTEGER                       NOT NULL,
  TOPIC           VARCHAR2(100 CHAR)            NOT NULL,
  INDEXN          INTEGER                       NOT NULL,
  CALDATE         DATE                          NOT NULL,
  ICUMATN         NUMBER,
  IBASATN         NUMBER,
  ALPHA           NUMBER,
  ACTATN          INTEGER,
  IATN            NUMBER,
  DELTAATN        NUMBER,
  TIMEATN         DATE,
```

```
  TIMEATNMINUS1   DATE,
  ACTPERIOD       NUMBER,
  TRUSTATN        NUMBER,
  TRUSTATNMINUS1  NUMBER,
  BETA            NUMBER
);

ALTER TABLE SO_HIST_TRUST_STACKOVERFLOW
 DROP PRIMARY KEY CASCADE;

DROP TABLE SO_HIST_TRUST_STACKOVERFLOW CASCADE CONSTRAINTS;

CREATE TABLE SO_HIST_TRUST_STACKOVERFLOW
(
  USERID   INTEGER                              NOT NULL,
  CALDATE  DATE                                 NOT NULL,
  TRUST    NUMBER                               NOT NULL
);

DROP TABLE SO_HIST_TRUST_STACKOVERFLOW_T CASCADE CONSTRAINTS;

CREATE TABLE SO_HIST_TRUST_STACKOVERFLOW_T
(
  USERID   INTEGER                              NOT NULL,
  CALDATE  DATE                                 NOT NULL,
  TRUST    NUMBER                               NOT NULL,
  TOPIC    VARCHAR2(100 CHAR)
);

ALTER TABLE SO_POSTS
 DROP PRIMARY KEY CASCADE;

DROP TABLE SO_POSTS CASCADE CONSTRAINTS;

CREATE TABLE SO_POSTS
(
  ID            INTEGER                         NOT NULL,
  CREATIONDATE  DATE,
  POSTTYPEID    INTEGER,
  PARENTID      INTEGER,
  USERID        INTEGER,
  TAGS          VARCHAR2(1000 CHAR)
);

ALTER TABLE SO_POSTTYPES
 DROP PRIMARY KEY CASCADE;

DROP TABLE SO_POSTTYPES CASCADE CONSTRAINTS;

CREATE TABLE SO_POSTTYPES
(
```

```
  ID    INTEGER                                 NOT NULL,
  NAME  VARCHAR2(100 CHAR)
);

ALTER TABLE SO_USERS
 DROP PRIMARY KEY CASCADE;

DROP TABLE SO_USERS CASCADE CONSTRAINTS;

CREATE TABLE SO_USERS
(
  ID            INTEGER                         NOT NULL,
  CREATIONDATE  DATE,
  REPUTATION    INTEGER,
  PRIM_TOPIC    VARCHAR2(100 CHAR)
);

ALTER TABLE SO_VOTES
 DROP PRIMARY KEY CASCADE;

DROP TABLE SO_VOTES CASCADE CONSTRAINTS;

CREATE TABLE SO_VOTES
(
  ID            INTEGER                         NOT NULL,
  USERID        INTEGER,
  VOTETYPEID    INTEGER,
  POSTID        INTEGER,
  CREATIONDATE  DATE,
  BOUNTYAMOUNT  INTEGER,
  TOPIC         VARCHAR2(100 CHAR)
);

ALTER TABLE SO_VOTETYPES
 DROP PRIMARY KEY CASCADE;

DROP TABLE SO_VOTETYPES CASCADE CONSTRAINTS;

CREATE TABLE SO_VOTETYPES
(
  ID               INTEGER                      NOT NULL,
  NAME             VARCHAR2(100 CHAR),
  REPUTATIONADDER  INTEGER
);

CREATE UNIQUE INDEX SO_CALENDAR_PK ON SO_CALENDAR
(CALDATE);

CREATE UNIQUE INDEX SO_COMMENTS_PK ON SO_COMMENTS
(ID);
```

```
CREATE UNIQUE INDEX SO_HIST_INTERACTION_DIBRM_PK ON SO_HIST_TRUST_DIBRM
(USERID, INDEXN);

CREATE UNIQUE INDEX SO_HIST_TRUST_DIBRM_T_PK ON SO_HIST_TRUST_DIBRM_T
(USERID, TOPIC, INDEXN);

CREATE UNIQUE INDEX SO_POSTS_PK ON SO_POSTS
(ID);

CREATE UNIQUE INDEX SO_POSTTYPES_PK ON SO_POSTTYPES
(ID);

CREATE UNIQUE INDEX SO_USERS_PK ON SO_USERS
(ID);

CREATE UNIQUE INDEX SO_USER_REPUTATION_PK ON SO_HIST_TRUST_STACKOVERFLOW
(USERID, CALDATE);

CREATE INDEX SO_VOTES_ID1 ON SO_VOTES
(USERID, VOTETYPEID, CREATIONDATE);

CREATE UNIQUE INDEX SO_VOTES_PK ON SO_VOTES
(ID);

CREATE UNIQUE INDEX SO_VOTETYPES_PK ON SO_VOTETYPES
(ID);

ALTER TABLE SO_CALENDAR ADD (
  CONSTRAINT SO_CALENDAR_PK
  PRIMARY KEY
  (CALDATE)
  USING INDEX SO_CALENDAR_PK
  ENABLE VALIDATE);

ALTER TABLE SO_COMMENTS ADD (
  CONSTRAINT SO_COMMENTS_PK
  PRIMARY KEY
  (ID)
  USING INDEX SO_COMMENTS_PK
  ENABLE VALIDATE);

ALTER TABLE SO_HIST_TRUST_DIBRM ADD (
  CONSTRAINT SO_HIST_INTERACTION_DIBRM_PK
  PRIMARY KEY
  (USERID, INDEXN)
  USING INDEX SO_HIST_INTERACTION_DIBRM_PK
  ENABLE VALIDATE);

ALTER TABLE SO_HIST_TRUST_DIBRM_T ADD (
  CONSTRAINT SO_HIST_TRUST_DIBRM_T_PK
  PRIMARY KEY
```

```
  (USERID, TOPIC, INDEXN)
  USING INDEX SO_HIST_TRUST_DIBRM_T_PK
  ENABLE VALIDATE);

ALTER TABLE SO_HIST_TRUST_STACKOVERFLOW ADD (
  CONSTRAINT SO_USER_REPUTATION_PK
  PRIMARY KEY
  (USERID, CALDATE)
  USING INDEX SO_USER_REPUTATION_PK
  ENABLE VALIDATE);

ALTER TABLE SO_POSTS ADD (
  CONSTRAINT SO_POSTS_PK
  PRIMARY KEY
  (ID)
  USING INDEX SO_POSTS_PK
  ENABLE VALIDATE);

ALTER TABLE SO_POSTTYPES ADD (
  CONSTRAINT SO_POSTTYPES_PK
  PRIMARY KEY
  (ID)
  USING INDEX SO_POSTTYPES_PK
  ENABLE VALIDATE);

ALTER TABLE SO_USERS ADD (
  CONSTRAINT SO_USERS_PK
  PRIMARY KEY
  (ID)
  USING INDEX SO_USERS_PK
  ENABLE VALIDATE);

ALTER TABLE SO_VOTES ADD (
  CONSTRAINT SO_VOTES_PK
  PRIMARY KEY
  (ID)
  USING INDEX SO_VOTES_PK
  ENABLE VALIDATE);

ALTER TABLE SO_VOTETYPES ADD (
  CONSTRAINT SO_VOTETYPES_PK
  PRIMARY KEY
  (ID)
  USING INDEX SO_VOTETYPES_PK
  ENABLE VALIDATE);
```

6.5 XML Files

6.5.1 Comments.xml

```
head -3 Comments.xml
<?xml version="1.0" encoding="utf-8"?>
<votes>
  <row Id="1" PostId="1" VoteTypeId="2" CreationDate="2008-07-31T00:00:00.000" />

tail -1 Comments.xml
</votes>

wc -l Comments.xml
83160603 Votes.xml
```

6.5.2 Posts.xml

```
head -3 Posts.xml
<?xml version="1.0" encoding="utf-8"?>
<posts>
  <row Id="4" PostTypeId="1" AcceptedAnswerId="7" CreationDate="2008-07-
31T21:42:52.667" Score="742" ViewCount="61738" Body="&lt;p&gt;I want to use a
&lt;code&gt;Track-Bar&lt;/code&gt; to change a &lt;code&gt;Form&lt;/code&gt;'s
opacity.&lt;/p&gt;&#xA;&lt;p&gt;This is my code:&lt;/p&gt;&#xA;&lt;pre
class=&quot;lang-cs prettyprint-override&quot;&gt;&lt;code&gt;decimal trans =
trackBar1.Value / 5000;&#xA;this.Opacity =
trans;&#xA;&lt;/code&gt;&lt;/pre&gt;&#xA;&lt;p&gt;When I build the application, it
gives the following error:&lt;/p&gt;&#xA;&lt;blockquote&gt;&#xA;&lt;pre
class=&quot;lang-none prettyprint-override&quot;&gt;&lt;code&gt;Cannot implicitly
convert type decimal to
double&#xA;&lt;/code&gt;&lt;/pre&gt;&#xA;&lt;/blockquote&gt;&#xA;&lt;p&gt;I have tried
using &lt;code&gt;trans&lt;/code&gt; and &lt;code&gt;double&lt;/code&gt;, but then the
&lt;code&gt;Control&lt;/code&gt; doesn't work. This code worked fine in a past VB.NET
project.&lt;/p&gt;&#xA;" OwnerUserId="8" LastEditorUserId="3072350"
LastEditorDisplayName="" LastEditDate="2021-02-26T03:31:15.027" LastActivityDate="2021-
11-15T21:15:29.713" Title="How to convert a Decimal to a Double in C#?"
Tags="&lt;c#&gt;&lt;floating-point&gt;&lt;type-
conversion&gt;&lt;double&gt;&lt;decimal&gt;" AnswerCount="12" CommentCount="3"
FavoriteCount="59" CommunityOwnedDate="2012-10-31T16:42:47.213" ContentLicense="CC BY-
SA 4.0" />

tail -1 Posts.xml
</posts>

wc -l Posts.xml
54741617 Posts.xml
```

6.5.3 Votes.xml

```
head -3 Votes.xml

<?xml version="1.0" encoding="utf-8"?>
<votes>
  <row Id="1" PostId="1" VoteTypeId="2" CreationDate="2008-07-31T00:00:00.000" />
<row Id="69393872" PostId="23858087" VoteTypeId="8" UserId="3166768"
CreationDate="2014-06-04T00:00:00.000" BountyAmount="100" />

tail -1 Votes.xml
</votes>

wc -l Votes.xml
222945520 Votes.xml
```

6.6 XML Parsers

6.6.1 Comments XML Parser

```
#!/usr/bin/env python
# coding: utf-8
# Filename: CommentXMLParser.py

import xml.etree.ElementTree as etree
import codecs
import csv
import time
import os

os.getcwd()
os.chdir('C:\\Users\\pjhome\\TUD\\Proj')

PATH_XML = 'C:\\Users\\pjhome\\TUD\\Proj\\'
FILENAME_XML = 'Comments.xml'
FILENAME_CSV = 'Comments.csv'

ENCODING = "utf-8"
def hms_string(sec_elapsed):
    h = int(sec_elapsed / (60 * 60))
    m = int((sec_elapsed % (60 * 60)) / 60)
    s = sec_elapsed % 60
    return "{}:{:>02}:{:>05.2f}".format(h, m, s)
def strip_tag_name(t):
    t = elem.tag
    idx = k = t.rfind("}")
    if idx != -1:
        t = t[idx + 1:]
    return t
```

```
pathXML = os.path.join(PATH_XML, FILENAME_XML)
pathCSV = os.path.join(PATH_XML, FILENAME_CSV)

totalCount = 0
title = None
start_time = time.time()

with codecs.open(pathCSV, "w", ENCODING) as CSVFH:
    CSVWriter = csv.writer(CSVFH, quoting=csv.QUOTE_MINIMAL)
    CSVWriter.writerow(['Id', 'UserId', 'PostId', 'CreationDate', 'Score'])

    for event, elem in etree.iterparse(pathXML, events=('start', 'end')):
        tname = strip_tag_name(elem.tag)
        if event == 'start':
            if tname == 'row':
                Id = elem.get('Id', '')
                UserId = elem.get('UserId', '')
                PostId = elem.get('PostId', '')
                CreationDate = elem.get('CreationDate', '')
                Score = elem.get('Score', '')
                totalCount += 1
                CSVWriter.writerow([Id, UserId, PostId, CreationDate, Score])
                elem.clear()
print(totalCount)
time_took = time.time() - start_time
print(f"Total runtime: {hms_string(time_took)}")
```

6.6.2 Posts XML Parser

```
#!/usr/bin/env python
# coding: utf-8
# Filename: PostXMLParser.py
import xml.etree.ElementTree as etree
import codecs
import csv
import time
import os

os.getcwd()
os.chdir('C:\\Users\\pjhome\\TUD\\Proj')

PATH_XML = 'C:\\Users\\pjhome\\TUD\\Proj\\'
FILENAME_XML = 'Posts.xml'
FILENAME_CSV = 'Posts.csv'

ENCODING = "utf-8"
def hms_string(sec_elapsed):
    h = int(sec_elapsed / (60 * 60))
    m = int((sec_elapsed % (60 * 60)) / 60)
    s = sec_elapsed % 60
    return "{}:{:>02}:{:>05.2f}".format(h, m, s)
```

```
def strip_tag_name(t):
    t = elem.tag
    idx = k = t.rfind("}")
    if idx != -1:
        t = t[idx + 1:]
    return t
pathXML = os.path.join(PATH_XML, FILENAME_XML)
pathCSV = os.path.join(PATH_XML, FILENAME_CSV)

totalCount = 0
title = None
start_time = time.time()

with codecs.open(pathCSV, "w", ENCODING) as CSVFH:
    CSVWriter = csv.writer(CSVFH, quoting=csv.QUOTE_MINIMAL)
    CSVWriter.writerow(['Id', 'CreationDate', 'PostTypeId', 'ParentId', 'UserId'])
    for event, elem in etree.iterparse(pathXML, events=('start', 'end')):
        tname = strip_tag_name(elem.tag)
        if event == 'start':
            if tname == 'row':
                Id = elem.get('Id', '')
                CreationDate = elem.get('CreationDate', '')
                PostTypeId = elem.get('PostTypeId', '')
                ParentId = elem.get('ParentId', '')
                OwnerUserId = elem.get('OwnerUserId', '')
                totalCount += 1
                CSVWriter.writerow([Id, CreationDate, PostTypeId, ParentId,
OwnerUserId])
                elem.clear()
print(totalCount)
time_took = time.time() - start_time
print(f"Total runtime: {hms_string(time_took)}")
```

6.6.3 Votes XML Parser

```
#!/usr/bin/env python
# coding: utf-8
# Filename: VoteXMLParser.py
import xml.etree.ElementTree as etree
import codecs
import csv
import time
import os

os.getcwd()
os.chdir('C:\\Users\\pjhome\\TUD\\Proj')

PATH_XML = 'C:\\Users\\pjhome\\TUD\\Proj\\'
FILENAME_XML = 'Votes.xml'
FILENAME_CSV = 'Votes.csv'
```

```
ENCODING = "utf-8"
def hms_string(sec_elapsed):
    h = int(sec_elapsed / (60 * 60))
    m = int((sec_elapsed % (60 * 60)) / 60)
    s = sec_elapsed % 60
    return "{}:{:>02}:{:>05.2f}".format(h, m, s)
def strip_tag_name(t):
    t = elem.tag
    idx = k = t.rfind("}")
    if idx != -1:
        t = t[idx + 1:]
    return t
pathXML = os.path.join(PATH_XML, FILENAME_XML)
pathCSV = os.path.join(PATH_XML, FILENAME_CSV)

totalCount = 0
title = None
start_time = time.time()

with codecs.open(pathCSV, "w", ENCODING) as CSVFH:
    CSVWriter = csv.writer(CSVFH, quoting=csv.QUOTE_MINIMAL)
    CSVWriter.writerow(['Id', 'UserId', 'VoteTypeId', 'PostId', 'CreationDate',
'BountyAmount'])
    for event, elem in etree.iterparse(pathXML, events=('start', 'end')):
        tname = strip_tag_name(elem.tag)
        if event == 'start':
            if tname == 'row':
                Id = elem.get('Id', '')
                UserId = elem.get('UserId', '')
                VoteTypeId = elem.get('VoteTypeId', '')
                PostId = elem.get('PostId', '')
                CreationDate = elem.get('CreationDate', '')
                BountyAmount = elem.get('BountyAmount', '')
                totalCount += 1
                CSVWriter.writerow([Id, UserId, VoteTypeId, PostId, CreationDate,
BountyAmount])
                elem.clear()
print(totalCount)
time_took = time.time() - start_time
print(f"Total runtime: {hms_string(time_took)}")
```

6.7 Oracle SQL*Loader Files

6.7.1 Control Files

Comments.ctl

```
OPTIONS ( SKIP=1, MULTITHREADING=TRUE)
LOAD DATA
```

```
INFILE 'C:\pj\Proj\Comments.csv'
BADFILE 'C:\pj\Proj\Comments.bad'
DISCARDFILE 'C:\pj\Proj\Comments.dsc'

INTO TABLE "SO_COMMENTS"
TRUNCATE
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
(ID,
USERID,
POSTID,
CREATIONDATE DATE "RRRR-MM-DD HH24:MI:SS",
SCORE)
```

Users.ctl

```
OPTIONS ( SKIP=1, MULTITHREADING=TRUE)
LOAD DATA
INFILE 'C:\pj\Proj\Users.csv'
BADFILE 'C:\pj\Proj\Users.bad'
DISCARDFILE 'C:\pj\Proj\Users.dsc'

INTO TABLE "SO_USERS"
TRUNCATE
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
(ID,
CREATIONDATE DATE "RRRR-MM-DD HH24:MI:SS",
REPUTATION)
```

Posts.ctl

```
OPTIONS ( SKIP=1, MULTITHREADING=TRUE)
LOAD DATA
INFILE 'C:\pj\Proj\Posts.csv'
BADFILE 'C:\pj\Proj\Posts.bad'
DISCARDFILE 'C:\pj\Proj\Posts.dsc'

INTO TABLE "SO_POSTS"
TRUNCATE
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
(ID,
CREATIONDATE DATE "RRRR-MM-DD HH24:MI:SS",
POSTTYPEID,
PARENTID,
USERID)
```

PostTypes.ctl

```
OPTIONS ( SKIP=1, MULTITHREADING=TRUE)
LOAD DATA
INFILE 'C:\pj\Proj\PostTypes.csv'
BADFILE 'C:\pj\Proj\PostTypes.bad'
DISCARDFILE 'C:\pj\Proj\PostTypes.dsc'

INTO TABLE "SO_POSTTYPES"
TRUNCATE
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
(ID,
NAME)
```

Votes.ctl

```
OPTIONS ( SKIP=1, MULTITHREADING=TRUE)
LOAD DATA
INFILE 'C:\pj\Proj\Votes.csv'
BADFILE 'C:\pj\Proj\Votes.bad'
DISCARDFILE 'C:\pj\Proj\Votes.dsc'

INTO TABLE "SO_VOTES"
TRUNCATE
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
(ID,
USERID,
VOTETYPEID,
POSTID,
CREATIONDATE DATE "RRRR-MM-DD HH24:MI:SS",
BOUNTYAMOUNT)
```

VoteTypes.ctl

```
OPTIONS ( SKIP=1, MULTITHREADING=TRUE)
LOAD DATA
INFILE 'C:\pj\Proj\VoteTypes.csv'
BADFILE 'C:\pj\Proj\VoteTypes.bad'
DISCARDFILE 'C:\pj\Proj\VoteTypes.dsc'

INTO TABLE "SO_VOTETYPES"
TRUNCATE
FIELDS TERMINATED BY ','
OPTIONALLY ENCLOSED BY '"' AND '"'
TRAILING NULLCOLS
(ID,
NAME)
```

6.7.2 Batch Files

Comments.bat

```
sqlldr CONTROL=Comments.ctl SILENT=feedback, header LOG=Comments.log BAD=Comments.bad
skip=1
```

Users.bat

```
sqlldr CONTROL=Users.ctl SILENT=feedback, header LOG=Users.log BAD=Users.bad skip=1
```

Posts.bat

```
sqlldr CONTROL=Posts.ctl SILENT=feedback, header LOG=Posts.log BAD=Posts.bad skip=1
```

PostTypes.bat

```
sqlldr CONTROL=PostTypes.ctl SILENT=feedback, header LOG=PostTypes.log
BAD=PostTypes.bad skip=1
```

Votes.bat

```
sqlldr system CONTROL=Votes.ctl SILENT=feedback, header LOG=Votes.log BAD=Votes.bad
skip=1
```

VoteTypes.bat

```
sqlldr CONTROL=VoteTypes.ctl SILENT=feedback, header LOG=VoteTypes.log
BAD=VoteTypes.bad skip=1
```

6.8 SEDE SQL Queries

6.8.1 Data Volumes SQL

```
SELECT 'Users' as Entity, count(*) as Volume from users
UNION ALL
SELECT 'Votes', count(*) from votes
UNION ALL
SELECT 'Posts', count(*) from posts
UNION ALL
SELECT 'Comments', count(*) from comments
UNION ALL
SELECT 'PostTypes', count(*) from posttypes
UNION ALL
SELECT 'VoteTypes', count(*) from votetypes;
```

6.8.2 Data Extraction SQL

```
--VoteTypes
SELECT Id, Name
```

```
FROM VoteTypes
ORDER BY 1;

--PostTypes
SELECT Id, Name
FROM PostTypes
ORDER BY 1;

--Users
  SELECT Id, CreationDate, Reputation
    FROM Users
   WHERE Id between 1 AND 300
ORDER BY Id;

-- Posts by the User
  SELECT Id,
         CreationDate,
         PostTypeId,
         ParentId,
         OwnerUserId AS UserId,
         Tags
    FROM Posts
   WHERE OwnerUserId between 1 AND 300
   AND posttypeid in (1,4,5,6)
   UNION
   SELECT Posts.Id,
         Posts.CreationDate,
         Posts.PostTypeId,
         Posts.ParentId,
         Posts.OwnerUserId AS UserId,
         par.Tags
    FROM Posts, Posts par
   WHERE     posts.parentId = par.Id
       AND Posts.posttypeid = 2
       AND Posts.OwnerUserId between 1 AND 300
ORDER BY OwnerUserId, Id;

-- Comments by the User
SELECT Comments.Id,
       Comments.UserId,
       Comments.PostId,
       Comments.CreationDate,
       Comments.Score,
       Posts.Tags
  FROM Comments, Posts
 WHERE     Comments.PostId = Posts.Id
       AND posttypeid in (1,4,5,6)
       AND Comments.UserId BETWEEN 1 AND 300
UNION
SELECT Comments.Id,
       Comments.UserId,
```

```
       Comments.PostId,
       Comments.CreationDate,
       Comments.Score,
       par.Tags
  FROM Comments, Posts, Posts par
 WHERE     Comments.PostId = Posts.Id
       AND Posts.parentid = par.id
       AND Posts.posttypeid = 2
       AND Comments.UserId BETWEEN 1 AND 300
ORDER BY UserId, Id;

-- Votes for the User
  SELECT Votes.Id,
         Posts.OwnerUserId AS UserId,
         Votes.VotetypeId,
         Votes.PostId,
         Votes.CreationDate,
         Votes.BountyAmount
    FROM Posts, Votes
   WHERE Posts.Id = Votes.PostId AND Posts.OwnerUserId between 1 AND 300
ORDER BY Posts.OwnerUserId, Votes.Id;
```

6.9 Database Views

6.9.1 Rules-based Model

```
CREATE OR REPLACE FORCE VIEW SO_HIST_INTERACTION_TOPIC_V
(USERID, CREATIONDATE, TOPIC, INTERACTIONTYPE)
AS

SELECT USERID,
       CREATIONDATE,
       NVL (
          REPLACE (REPLACE (SUBSTR (tags, 1, INSTR (tags, '>', 1)), '<'),
                   '>'),
          'NA')
          AS topic,
       'POST' AS interactiontype
  FROM so_posts
UNION
SELECT USERID,
       CREATIONDATE,
       NVL (
          REPLACE (REPLACE (SUBSTR (tags, 1, INSTR (tags, '>', 1)), '<'),
                   '>'),
          'NA')
          AS topic,
       'COMMENT' AS interactiontype
```

```
  FROM so_comments;


CREATE OR REPLACE FORCE VIEW SO_HIST_INTERACTION_TOPIC_V
(USERID, CREATIONDATE, TOPIC, INTERACTIONTYPE)
BEQUEATH DEFINER
AS

SELECT USERID,
       CREATIONDATE,
       NVL (
          REPLACE (REPLACE (SUBSTR (tags, 1, INSTR (tags, '>', 1)), '<'),
                   '>'),
          'NA')
          AS topic,
       'POST' AS interactiontype
  FROM so_posts
UNION
SELECT USERID,
       CREATIONDATE,
       NVL (
          REPLACE (REPLACE (SUBSTR (tags, 1, INSTR (tags, '>', 1)), '<'),
                   '>'),
          'NA')
          AS topic,
       'COMMENT' AS interactiontype
  FROM so_comments;
```

6.9.2 DIBRM Models

```
CREATE OR REPLACE FORCE VIEW SO_HIST_TRUST_DIBRM_V
(USERID, CALDATE, DAYNUM, TRUST, CUMTRUST)
BEQUEATH DEFINER
AS

SELECT a.userid,
       TRUNC (a.CalDate) AS caldate,
       TRUNC (a.CalDate) - TRUNC (b.CreationDate) AS daynum,
       ROUND (trustatn, 0) AS trust,
       SUM (ROUND (trustatn, 0)) OVER (PARTITION BY userid ORDER BY caldate)
          AS cumtrust
  FROM so_hist_trust_dibrm a, so_users b
 WHERE a.userid = b.id;

CREATE OR REPLACE FORCE VIEW SO_HIST_TRUST_DIBRM_T_V
(USERID, TOPIC, CALDATE, DAYNUM, TRUST,
 CUMTRUST)
BEQUEATH DEFINER
AS
```

```
SELECT a.userid,
       a.topic,
       TRUNC (a.CalDate) AS caldate,
       TRUNC (a.CalDate) - TRUNC (b.CreationDate) AS daynum,
       ROUND (trustatn, 0) AS trust,
       SUM (ROUND (trustatn, 0))
          OVER (PARTITION BY userid, topic ORDER BY caldate)
          AS cumtrust
  FROM so_hist_trust_dibrm_t a, so_users b
 WHERE a.userid = b.id;

CREATE OR REPLACE FORCE VIEW SO_HIST_TRUSTMAXPERDAY_DIBRM_V
(USERID, CALDATE, DAYNUM, TRUST, CUMTRUST)
BEQUEATH DEFINER
AS

SELECT a.userid,
       TRUNC (a.CalDate) AS caldate,
       TRUNC (a.CalDate) - TRUNC (b.CreationDate) AS daynum,
       ROUND (trustatn, 0) AS trust,
       SUM (ROUND (trustatn, 0)) OVER (PARTITION BY userid ORDER BY caldate)
          AS cumtrust
  FROM (  SELECT userid, TRUNC (caldate) AS caldate, MAX (trustatn) AS trustatn --Use
max trust per user per day
            FROM so_hist_trust_dibrm
        GROUP BY userid, TRUNC (caldate)) a,
       so_users b
 WHERE a.userid = b.id;
CREATE OR REPLACE FORCE VIEW SO_HIST_TRUSTMAXPERDAY_DIBRM_T_V
(USERID, TOPIC, CALDATE, DAYNUM, TRUST,
 CUMTRUST)
BEQUEATH DEFINER
AS

SELECT a.userid,
       a.topic,
       TRUNC (a.CalDate) AS caldate,
       TRUNC (a.CalDate) - TRUNC (b.CreationDate) AS daynum,
       ROUND (trustatn, 0) AS trust,
       SUM (ROUND (trustatn, 0))
          OVER (PARTITION BY userid, topic ORDER BY caldate)
          AS cumtrust
  FROM (  SELECT userid,
                 topic,
                 TRUNC (caldate) AS caldate,
                 MAX (trustatn) AS trustatn   --Use max trust per user per day
            FROM so_hist_trust_dibrm_t
        GROUP BY userid, topic, TRUNC (caldate)) a,
       so_users b
 WHERE a.userid = b.id;
```

6.10 PL/SQL Procedure Code

6.10.1 Rules based

```
CREATE OR REPLACE PROCEDURE SO_REPUTATION_PROC
AS
   CURSOR c1
   IS
        SELECT usr.id AS userid,
               TRUNC (usr.creationdate) AS UserCreationDate,
               TRUNC (cal.caldate) AS CalendarDate
          FROM so_users usr, so_calendar cal
         WHERE TRUNC (usr.creationdate) <= TRUNC (cal.caldate)
      ORDER BY id, caldate;

   --https://stackoverflow.com/help/whats-reputation

   loc_start_date         DATE;
   loc_num_votes          INTEGER;
   loc_olduserid          INTEGER;
   loc_day_upvote_rep     INTEGER;
   loc_day_dwvote_rep     INTEGER;
   loc_day_edit_rep       INTEGER;
   loc_day_comb_rep       INTEGER;
   loc_day_accepted_rep   INTEGER;
   loc_day_total_rep      INTEGER;
   loc_initial_rep        INTEGER;
BEGIN
   EXECUTE IMMEDIATE 'TRUNCATE TABLE so_calendar';

   EXECUTE IMMEDIATE 'TRUNCATE TABLE so_hist_trust_stackoverflow';

   -- Get the earliest date of all users
   SELECT TO_DATE (MIN (creationdate), 'DD-MON-RRRR')
     INTO loc_start_date
     FROM so_users;

   --Build calendar from earliest date to today
   --https://blogs.oracle.com/sql/post/how-to-generate-days-weeks-or-months-between-
two-dates-in-oracle-database
   INSERT INTO so_calendar (caldate)
          SELECT loc_start_date + LEVEL - 1
            FROM DUAL
      CONNECT BY LEVEL <= (SYSDATE - loc_start_date + 1);

   COMMIT;

   --Insert reputation points based upon rules
```

```
loc_olduserid := 0;

FOR c1_rec IN C1
LOOP
   BEGIN
      -- 1) All users start with one reputation point,
      IF c1_rec.userid != loc_olduserid
      THEN
         loc_initial_rep := 1;
      ELSE
         loc_initial_rep := 0;
      END IF;

      -- question is voted up: +10
      -- answer is voted up: +10
      -- article is voted up: +10
      SELECT NVL (COUNT (v.id), 0)
        INTO loc_num_votes
        FROM so_votes v, so_votetypes vt
       WHERE     v.votetypeid = vt.id
             AND vt.name = 'UpMod'
             AND creationdate = c1_rec.CalendarDate
             AND v.userid = c1_rec.userid;

      loc_day_upvote_rep := loc_num_votes * 10;

      --your question is voted down: -2
      --your answer is voted down: -2
      --your article is downvoted: -2
      SELECT NVL (COUNT (v.id), 0)
        INTO loc_num_votes
        FROM so_votes v, so_votetypes vt
       WHERE     v.votetypeid = vt.id
             AND vt.name = 'DownMod'
             AND creationdate = c1_rec.CalendarDate
             AND v.userid = c1_rec.userid;

      loc_day_dwvote_rep := loc_num_votes * -2;

      --suggested edit is accepted: +2 (up to +1000 total per user)
      SELECT NVL (COUNT (v.id), 0)
        INTO loc_num_votes
        FROM so_votes v, so_votetypes vt
       WHERE     v.votetypeid = vt.id
             AND vt.name = 'ApproveEditSuggestion'
             AND creationdate = c1_rec.CalendarDate
             AND v.userid = c1_rec.userid;

      loc_day_edit_rep := loc_num_votes * 10;
```

```
         --You can earn a maximum of 200 reputation per day from the combination of
upvotes, downvotes and suggested edits
         loc_day_comb_rep :=
            loc_day_upvote_rep + loc_day_dwvote_rep + loc_day_edit_rep;

         IF loc_day_comb_rep > 200
         THEN
            loc_day_comb_rep := 200;
         END IF;

         -- answer is marked "accepted": +15 (+2 to acceptor)
         SELECT NVL (COUNT (v.id), 0)
           INTO loc_num_votes
           FROM so_votes v, so_votetypes vt
          WHERE     v.votetypeid = vt.id
                AND vt.name = 'AcceptedByOriginator'
                AND creationdate = c1_rec.CalendarDate
                AND v.userid = c1_rec.userid;

         loc_day_accepted_rep := loc_num_votes * 15;

         loc_day_total_rep :=
            loc_initial_rep + loc_day_comb_rep + loc_day_accepted_rep;

         -- one of your posts receives 6 spam or offensive flags: -100
         INSERT INTO so_hist_trust_stackoverflow (userid, caldate,   --daynum,
                                                                  trust)
              VALUES (c1_rec.userid, c1_rec.CalendarDate, --c1_rec.CalendarDate -
c1_rec.UserCreationDate,
                                                       loc_day_total_rep);

         COMMIT;

         loc_olduserid := c1_rec.userid;
      END;
   END LOOP;
/* EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      NULL;
   WHEN OTHERS
   THEN
      ROLLBACK;
      DBMS_OUTPUT.put_line (
         'Error code ' || SQLCODE || ': ' || SUBSTR (SQLERRM, 1, 255)); */
END;
/
```

6.10.2 Rules based Topic

```
CREATE OR REPLACE PROCEDURE SO_PRIM_TOPIC_UPDATE_PROC
AS
   CURSOR c1
   IS
      SELECT id FROM so_users;

   loc_topic   VARCHAR2 (100);
   loc_vol     INT;
BEGIN
   UPDATE so_votes a
      SET topic =
             (SELECT NVL (
                        REPLACE (
                           REPLACE (
                              SUBSTR (b.tags, 1, INSTR (b.tags, '>', 1)),
                              '<'),
                           '>'),
                        'NA')
                FROM so_posts b
               WHERE a.postid = b.id)
    WHERE EXISTS
             (SELECT 'x'
                FROM so_posts b
               WHERE a.postid = b.id);

   COMMIT;

   FOR c1_rec IN c1
   LOOP
      BEGIN
              SELECT topic, COUNT (*) AS vol
                INTO loc_topic, loc_vol
                FROM so_votes
               WHERE userid = c1_rec.id
            GROUP BY topic
            ORDER BY vol DESC
         FETCH FIRST 1 ROWS ONLY;

         UPDATE so_users
            SET prim_topic = loc_topic
          WHERE id = c1_rec.id;

         COMMIT;
      EXCEPTION
         WHEN NO_DATA_FOUND
         THEN
            NULL;
      END;
   END LOOP;
```

```
END;
/


CREATE OR REPLACE PROCEDURE SO_REPUTATION_TOPIC_PROC
AS
   CURSOR c1
   IS
        SELECT usr.id AS userid,
               usr.prim_topic AS PrimTopic,
               TRUNC (usr.creationdate) AS UserCreationDate,
               TRUNC (cal.caldate) AS CalendarDate
          FROM so_users usr, so_calendar cal
         WHERE TRUNC (usr.creationdate) <= TRUNC (cal.caldate)
      ORDER BY usr.id, usr.prim_topic, cal.caldate;

   --https://stackoverflow.com/help/whats-reputation

   loc_start_date         DATE;
   loc_num_votes          INTEGER;
   loc_olduserid          INTEGER;
   loc_day_upvote_rep     INTEGER;
   loc_day_dwvote_rep     INTEGER;
   loc_day_edit_rep       INTEGER;
   loc_day_comb_rep       INTEGER;
   loc_day_accepted_rep   INTEGER;
   loc_day_total_rep      INTEGER;
   loc_initial_rep        INTEGER;
BEGIN
   EXECUTE IMMEDIATE 'TRUNCATE TABLE so_calendar';

   EXECUTE IMMEDIATE 'TRUNCATE TABLE so_hist_trust_stackoverflow_t';

   -- Get the earliest date of all users
   SELECT TO_DATE (MIN (creationdate), 'DD-MON-RRRR')
     INTO loc_start_date
     FROM so_users;

   --Build calendar from earliest date to today
   --https://blogs.oracle.com/sql/post/how-to-generate-days-weeks-or-months-between-
two-dates-in-oracle-database
   INSERT INTO so_calendar (caldate)
          SELECT loc_start_date + LEVEL - 1
            FROM DUAL
      CONNECT BY LEVEL <= (SYSDATE - loc_start_date + 1);

   COMMIT;

   --Insert reputation points based upon rules
   loc_olduserid := 0;
```

```
FOR c1_rec IN C1
LOOP
   BEGIN
      -- 1) All users start with one reputation point,
      IF c1_rec.userid != loc_olduserid
      THEN
         loc_initial_rep := 1;
      ELSE
         loc_initial_rep := 0;
      END IF;

      -- question is voted up: +10
      -- answer is voted up: +10
      -- article is voted up: +10
      SELECT NVL (COUNT (v.id), 0)
        INTO loc_num_votes
        FROM so_votes v, so_votetypes vt
       WHERE     v.votetypeid = vt.id
             AND vt.name = 'UpMod'
             AND v.topic = c1_rec.PrimTopic
             AND creationdate = c1_rec.CalendarDate
             AND v.userid = c1_rec.userid;

      loc_day_upvote_rep := loc_num_votes * 10;

      --your question is voted down: -2
      --your answer is voted down: -2
      --your article is downvoted: -2
      SELECT NVL (COUNT (v.id), 0)
        INTO loc_num_votes
        FROM so_votes v, so_votetypes vt
       WHERE     v.votetypeid = vt.id
             AND vt.name = 'DownMod'
             AND v.topic = c1_rec.PrimTopic
             AND creationdate = c1_rec.CalendarDate
             AND v.userid = c1_rec.userid;

      loc_day_dwvote_rep := loc_num_votes * -2;

      --suggested edit is accepted: +2 (up to +1000 total per user)
      SELECT NVL (COUNT (v.id), 0)
        INTO loc_num_votes
        FROM so_votes v, so_votetypes vt
       WHERE     v.votetypeid = vt.id
             AND vt.name = 'ApproveEditSuggestion'
             AND v.topic = c1_rec.PrimTopic
             AND creationdate = c1_rec.CalendarDate
             AND v.userid = c1_rec.userid;

      loc_day_edit_rep := loc_num_votes * 10;
```

```
         --You can earn a maximum of 200 reputation per day from the combination of
upvotes, downvotes and suggested edits
         loc_day_comb_rep :=
            loc_day_upvote_rep + loc_day_dwvote_rep + loc_day_edit_rep;

         IF loc_day_comb_rep > 200
         THEN
            loc_day_comb_rep := 200;
         END IF;

         -- answer is marked "accepted": +15 (+2 to acceptor)
         SELECT NVL (COUNT (v.id), 0)
           INTO loc_num_votes
           FROM so_votes v, so_votetypes vt
          WHERE     v.votetypeid = vt.id
                AND vt.name = 'AcceptedByOriginator'
                AND v.topic = c1_rec.PrimTopic
                AND creationdate = c1_rec.CalendarDate
                AND v.userid = c1_rec.userid;

         loc_day_accepted_rep := loc_num_votes * 15;

         loc_day_total_rep :=
            loc_initial_rep + loc_day_comb_rep + loc_day_accepted_rep;

         -- one of your posts receives 6 spam or offensive flags: -100
         INSERT INTO so_hist_trust_stackoverflow_t (userid,
                                                    caldate,
                                                    topic,
                                                    trust)
              VALUES (c1_rec.userid,
                      c1_rec.CalendarDate,
                      c1_rec.PrimTopic,
                      loc_day_total_rep);

         COMMIT;

         loc_olduserid := c1_rec.userid;
      END;
   END LOOP;
/* EXCEPTION
   WHEN NO_DATA_FOUND
   THEN
      NULL;
   WHEN OTHERS
   THEN
      ROLLBACK;
      DBMS_OUTPUT.put_line (
         'Error code ' || SQLCODE || ': ' || SUBSTR (SQLERRM, 1, 255)); */
END;
/
```

6.10.3 DIBRM Procedure Code

```
CREATE OR REPLACE PROCEDURE SO_DIBRM_PROC (
   in_IBasAtn     IN NUMBER DEFAULT 2,   -- Cumulative interaction value at n
   in_Alpha       IN NUMBER DEFAULT 1, -- Weight of the cumulative interaction value
chosen
   in_ActPeriod   IN NUMBER DEFAULT 1, -- Size of activity period chosen
   in_Beta        IN NUMBER DEFAULT 0.99)          -- Forgetting Factor chosen
AS
   ICumAtn           NUMBER;              -- Cumulative interaction value at n
   --  IBasAtn           INTEGER;               -- Basic interaction value chosen
   --  Alpha             NUMBER; -- Weight of the cumulative interaction value chosen
   ActAtn            INTEGER;     -- Total count of Activites at interaction n
   IAtn              NUMBER;                   -- Actual Interation value At n

   DeltaAtn          NUMBER; -- Number of periods between n and n-1 interactions
   TimeAtn           DATE;                    -- DateTime of the interaction n
   TimeAtnMinus1     DATE;                  -- DateTime of the interaction n-1
   --   in_ActPeriod         INTEGER;               -- Size of activity period chosen

   TrustAtn          NUMBER;      -- Calculated Trust of user at interaction n
   TrustAtnMinus1    NUMBER;    -- Calculated Trust of user at interaction n-1

   --  in_Beta              NUMBER;                       -- Forgetting Factor chosen

   UserIdAtnMinus1   INTEGER;

   CURSOR c1
   IS
        SELECT userid, creationdate
          FROM so_hist_interaction_v
      ORDER BY userid, creationdate;
BEGIN
   UserIdAtnMinus1 := 0;

   EXECUTE IMMEDIATE 'TRUNCATE TABLE so_hist_trust_dibrm';

   FOR c1_rec IN c1
   LOOP
      BEGIN
         IF c1_rec.userid != UserIdAtnMinus1
         THEN
            ActAtn := 1;                    -- Total number of activities is 1
            TimeAtnMinus1 := c1_rec.creationdate;    -- Time of n equal to n-1
            TrustAtnMinus1 := 0;                        --Trust at n-1 is zero
         END IF;

         ICumAtn := in_IBasAtn * in_Alpha * (1 - (1 / (ActAtn + 1)));
         IAtn := in_IBasAtn + ICumAtn;

         TimeAtn := c1_rec.creationdate;
```

```
   DeltaAtn := (TimeAtn - TimeAtnMinus1) / in_ActPeriod;

   /*   DeltaAtn := ROUND (DeltaAtn, 0);

      IF DeltaAtn < 1
      THEN
         DeltaAtn := 1;
      END IF; */

   TrustAtn := (TrustAtnMinus1 * POWER (in_Beta, DeltaAtn)) + IAtn;
   DBMS_OUTPUT.PUT_LINE (
      'TrustAtn = ' || c1_rec.userid || '-' || TrustAtn);

   INSERT INTO so_hist_trust_dibrm (userid,
                                            indexn,
                                            caldate,
                                            icumatn,
                                            IBasAtn,
                                            Alpha,
                                            actatn,
                                            iatn,
                                            deltaatn,
                                            timeatn,
                                            timeatnminus1,
                                            ActPeriod,
                                            trustatn,
                                            trustatnminus1,
                                            Beta)
        VALUES (c1_rec.userid,
                ActAtn,
                c1_rec.creationdate,
                icumatn,
                in_IBasAtn,
                in_Alpha,
                actatn,
                iatn,
                DeltaAtn,
                timeatn,
                timeatnminus1,
                in_ActPeriod,
                trustatn,
                trustatnminus1,
                in_Beta);

   COMMIT;

   ActAtn := ActAtn + 1;
   TimeAtnMinus1 := TimeAtn;
   TrustAtnMinus1 := TrustAtn;
   UserIdAtnMinus1 := c1_rec.userid;
END;
```

```
   END LOOP;
/* EXCEPTION
WHEN NO_DATA_FOUND
THEN
   NULL;
WHEN OTHERS
THEN
   ROLLBACK;
   DBMS_OUTPUT.put_line (
      'Error code ' || SQLCODE || ': ' || SUBSTR (SQLERRM, 1, 255)); */

END;
/
```

6.10.4 DIBRM Topic Procedure Code

```
CREATE OR REPLACE PROCEDURE SO_DIBRM_TOPIC_PROC (
   in_IBasAtn     IN NUMBER DEFAULT 2,    -- Cumulative interaction value at n
   in_Alpha       IN NUMBER DEFAULT 1, -- Weight of the cumulative interaction value
chosen
   in_ActPeriod   IN NUMBER DEFAULT 1,       -- Size of activity period chosen
   in_Beta        IN NUMBER DEFAULT 0.99)          -- Forgetting Factor chosen
AS
   ICumAtn           NUMBER;              -- Cumulative interaction value at n
   --  IBasAtn           INTEGER;               -- Basic interaction value chosen
   --  Alpha             NUMBER; -- Weight of the cumulative interaction value chosen
   ActAtn            INTEGER;     -- Total count of Activites at interaction n
   IAtn              NUMBER;                   -- Actual Interation value At n

   DeltaAtn          NUMBER; -- Number of periods between n and n-1 interactions
   TimeAtn           DATE;                    -- DateTime of the interaction n
   TimeAtnMinus1     DATE;                  -- DateTime of the interaction n-1
   --   in_ActPeriod         INTEGER;               -- Size of activity period chosen

   TrustAtn          NUMBER;      -- Calculated Trust of user at interaction n
   TrustAtnMinus1    NUMBER;    -- Calculated Trust of user at interaction n-1

   --  in_Beta             NUMBER;                       -- Forgetting Factor chosen

   TrustEntityMinus1   VARCHAR(200);

   CURSOR c1
   IS
        SELECT userid || '-' || topic AS TrustEntity,
               userid,
               topic,
               creationdate
          FROM so_hist_interaction_topic_v
      ORDER BY userid, topic, creationdate;
BEGIN
   TrustEntityMinus1 := 'XYZ';
```

```
EXECUTE IMMEDIATE 'TRUNCATE TABLE so_hist_trust_dibrm_t';

FOR c1_rec IN c1
LOOP
   BEGIN
      IF c1_rec.TrustEntity != TrustEntityMinus1
      THEN
         ActAtn := 1;                    -- Total number of activities is 1
         TimeAtnMinus1 := c1_rec.creationdate;    -- Time of n equal to n-1
         TrustAtnMinus1 := 0;                        --Trust at n-1 is zero
      END IF;

      ICumAtn := in_IBasAtn * in_Alpha * (1 - (1 / (ActAtn + 1)));
      IAtn := in_IBasAtn + ICumAtn;

      TimeAtn := c1_rec.creationdate;
      DeltaAtn := (TimeAtn - TimeAtnMinus1) / in_ActPeriod;

      /*   DeltaAtn := ROUND (DeltaAtn, 0);

         IF DeltaAtn < 1
         THEN
            DeltaAtn := 1;
         END IF; */

      TrustAtn := (TrustAtnMinus1 * POWER (in_Beta, DeltaAtn)) + IAtn;
      DBMS_OUTPUT.PUT_LINE (
         'TrustAtn = ' || c1_rec.userid || '-' || TrustAtn);

      INSERT INTO so_hist_trust_dibrm_t (userid,
                                         topic,
                                         indexn,
                                         caldate,
                                         icumatn,
                                         IBasAtn,
                                         Alpha,
                                         actatn,
                                         iatn,
                                         deltaatn,
                                         timeatn,
                                         timeatnminus1,
                                         ActPeriod,
                                         trustatn,
                                         trustatnminus1,
                                         Beta)
           VALUES (c1_rec.userid,
                   c1_rec.topic,
                   ActAtn,
                   c1_rec.creationdate,
                   icumatn,
```

```
                    in_IBasAtn,
                    in_Alpha,
                    actatn,
                    iatn,
                    DeltaAtn,
                    timeatn,
                    timeatnminus1,
                    in_ActPeriod,
                    trustatn,
                    trustatnminus1,
                    in_Beta);

         COMMIT;

         ActAtn := ActAtn + 1;
         TimeAtnMinus1 := TimeAtn;
         TrustAtnMinus1 := TrustAtn;
         TrustEntityMinus1 := c1_rec.TrustEntity;
      END;
   END LOOP;
/* EXCEPTION
WHEN NO_DATA_FOUND
THEN
   NULL;
WHEN OTHERS
THEN
   ROLLBACK;
   DBMS_OUTPUT.put_line (
      'Error code ' || SQLCODE || ': ' || SUBSTR (SQLERRM, 1, 255)); */

END;
/
```

6.11 Implementation Artifacts

Entity	Available via SEDE? (Y/N)	Available via Data Dump? (Y/N)	Required for Research? (Y/N)
Badges	Y	Y	
CloseAsOffTopicReasonTypes	Y		
CloseReasonTypes	Y		
Comments	Y	Y	Y
FlagTypes	Y		
PendingFlags	Y		
PostFeedback	Y		
PostHistory	Y	Y	
PostHistoryTypes	Y		
PostLinks	Y	Y	
PostNotices	Y		
PostNoticeTypes	Y		
Posts	Y	Y	Y
PostsWithDeleted	Y		
PostTags	Y	Y	
PostTypes	Y		Y
ReviewRejectionReasons	Y		
ReviewTaskResults	Y		
ReviewTaskResultTypes	Y		
ReviewTasks	Y		
ReviewTaskStates	Y		
ReviewTaskTypes	Y		
SuggestedEdits	Y		
SuggestedEditVotes	Y		
Tags	Y		
TagSynonyms	Y		
Users	Y	Y	Y
Votes	Y	Y	Y
VoteTypes	Y		Y
SuggestedEdits	Y		

Table 6.1 - Stack Overview Entities List

Entity	Attribute	Description	Measurement Level
Comments	Id	Comment unique id.	Nominal
Comments	UserId	Community user who submitted the comment. NOTE: Absent if user has been deleted.	Nominal
Comments	PostId	Identifying the post record that this comment relates.	Nominal
Comments	CreationDate	Date when the comment was created.	Ordinal
Comments	Score	Score of the comment. Calculated based upon upvotes minus downvotes.	Interval
Posts	Id	Post unique id.	Nominal
Posts	CreationDate	Date when the post was created.	Ordinal
Posts	PostTypeId	Id identifying the post type.	Nominal
Posts	ParentId	The parent post record i.e., the Question record, and is only present on Answer records i.e., when PostTypeId = 2	Nominal
Posts	OwnerUserId	The community user who created post	Nominal
PostTypes	Id	Post type unique Id.	Nominal
PostTypes	Name	Post type description.	Nominal
Users	Id	Community user unique id.	Nominal
Users	CreationDate	Community member registration date.	Nominal
Users	Reputation	Reputation of Community member.	Ordinal
Votes	Id	Vote unique Id.	Nominal
Posts	OwnerUserId	Identifies the community user who create the post that this vote pertains.	Nominal
Votes	VoteTypeId	Id identifying the vote type. The foreign key from vote type table.	Nominal
Votes	PostId	Identifying the post record that this vote relates.	Nominal
Votes	CreationDate	Date when the vote was cast.	Ordinal
Votes	BountyAmount	Bounty amounts present only if VoteTypeId in (8,9).	Ratio
VoteTypes	Id	Vote type unique Id.	Nominal
VoteTypes	Name	Vote type description.	Nominal

Table 6.2 - Data Descriptor Detail

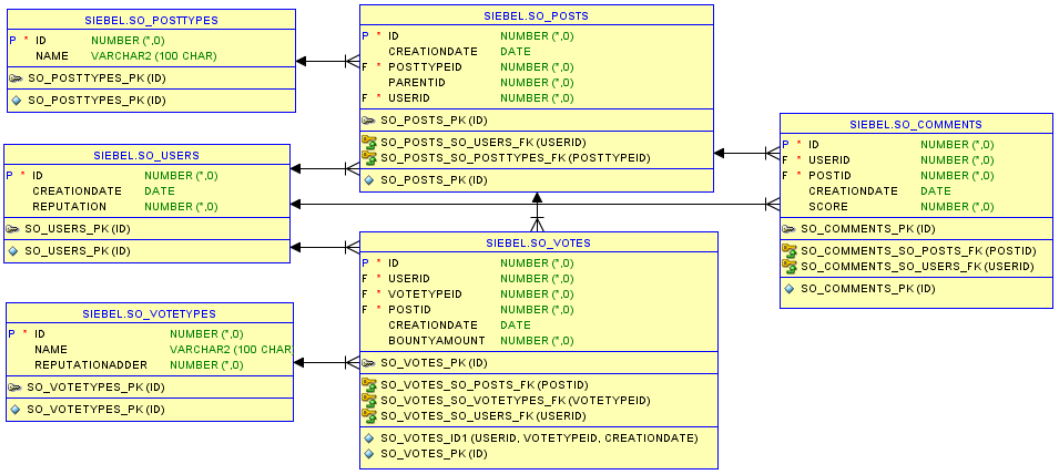


Figure 6.1 – Oracle Database Schema Data Model

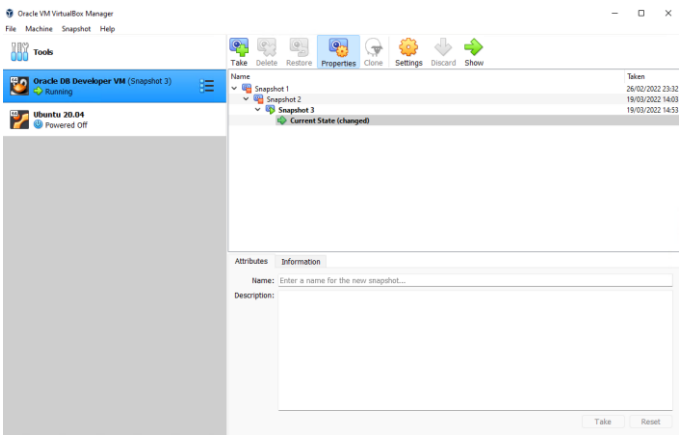


Figure 6.2 - Oracle Virtual Box Configuration

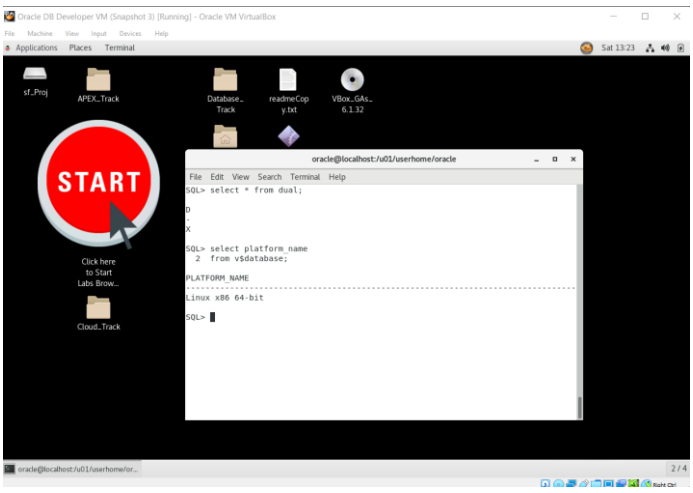


Figure 6.3 - Linux VM with pre-installed Oracle Database

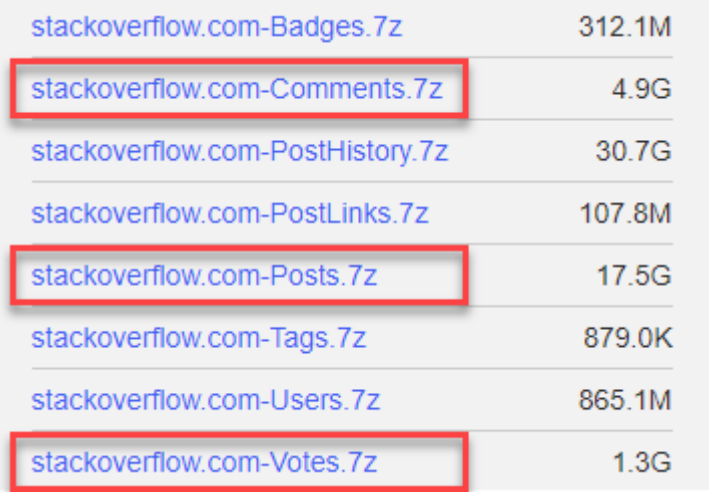


Figure 6.4 - Stack Overflow Data Dumps

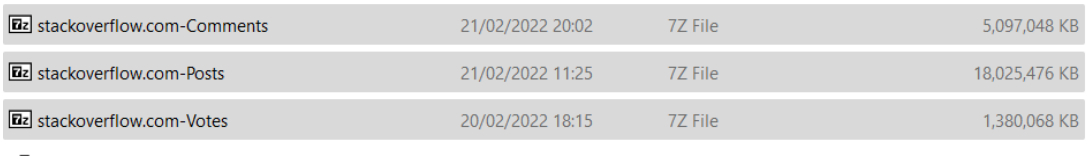


Figure 6.5 - Downloaded Data Dump Files

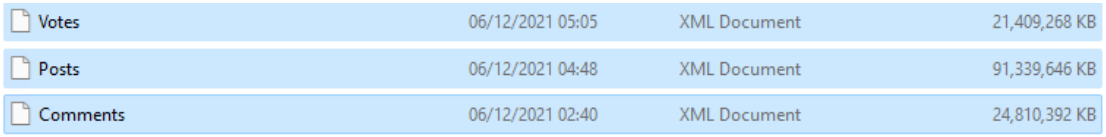


Figure 6.6 - Decompressed XML Files

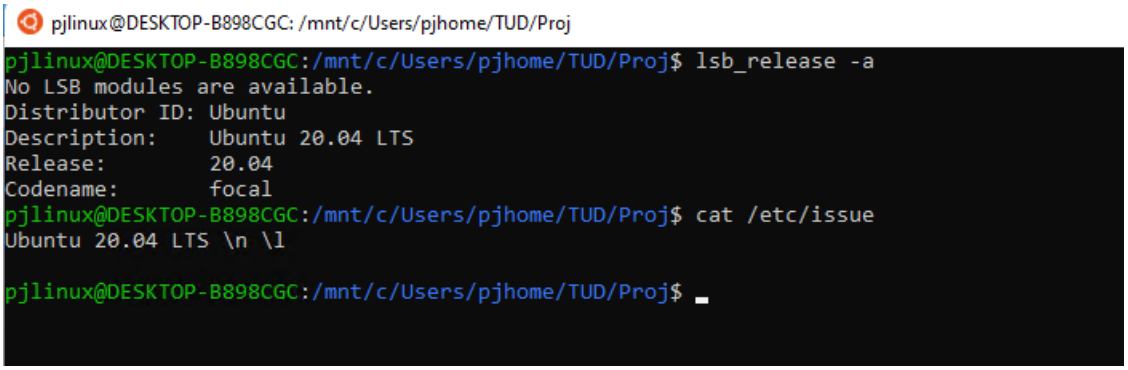


Figure 6.7 - Ubuntu for Windows Screenshot

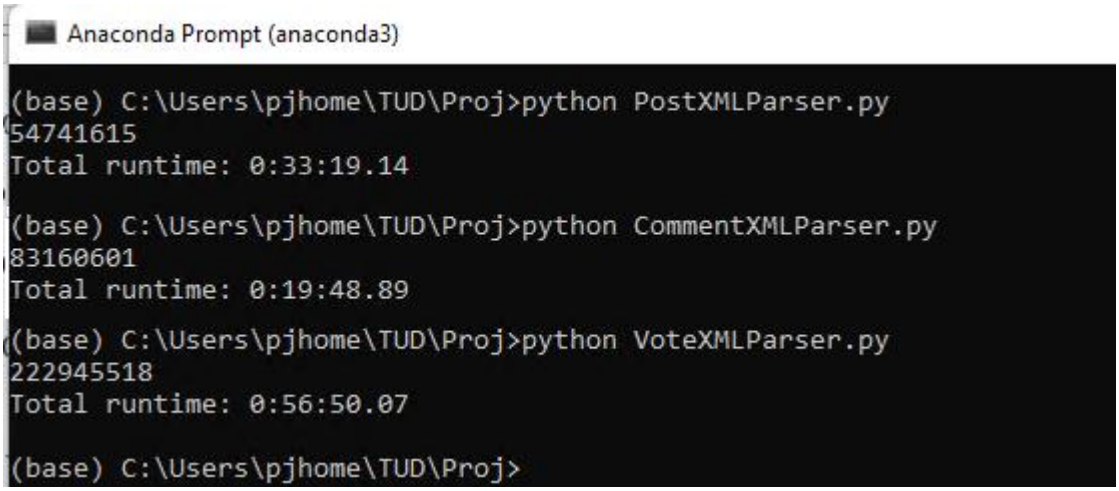


Figure 6.8 - Parser Execution Stats.

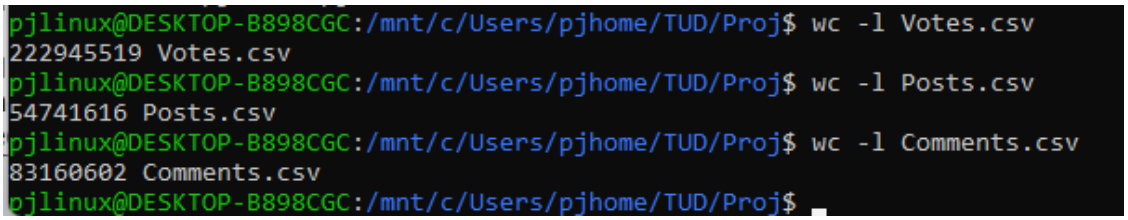


Figure 6.9 - CSV File Record Counts

XML Parser Code	Input File	Volume	Output File	Volume	Execution Time (mins)
CommentXMLParser.py	Comments.xml	83160603	Comments.csv	83160602	19
PostXMLParser.py	Posts.xml	54741617	Posts.csv	54741616	33
VoteXMLParser.py	Votes.xml	222945520	Votes.csv	222945519	56

Table 6.3 – XML Parser Stats.

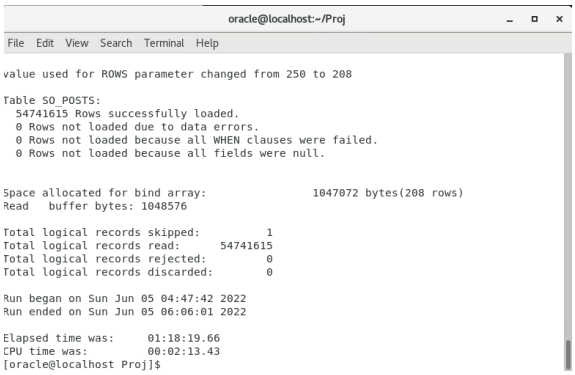


Figure 6.10 - Posts Data Load Log

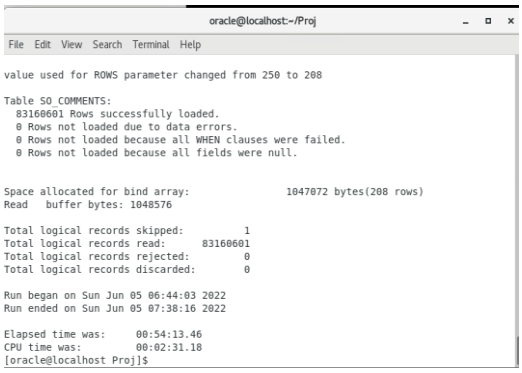


Figure 6.11 - Comments Data Load Log

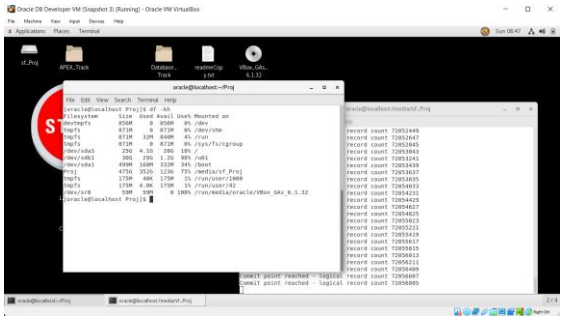


Figure 6.12 - Virtual Machine Storage Issue

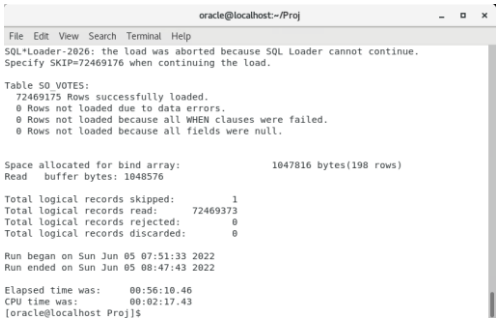


Figure 6.13 - Votes Data Load Log

SQL*Loader Control File	Input File	Volume	Database Table	Volume	Execution Time (mins)
Comments.ctl	Comments.csv	83160602	SO_COMMENTS	83160601	54
Posts.ctl	Posts.csv	54741616	SO_POSTS	54741615	78
Votes.ctl	Votes.csv	222945519	SO_VOTES	72469373	56

Table 6.4 - Data Loading Stats

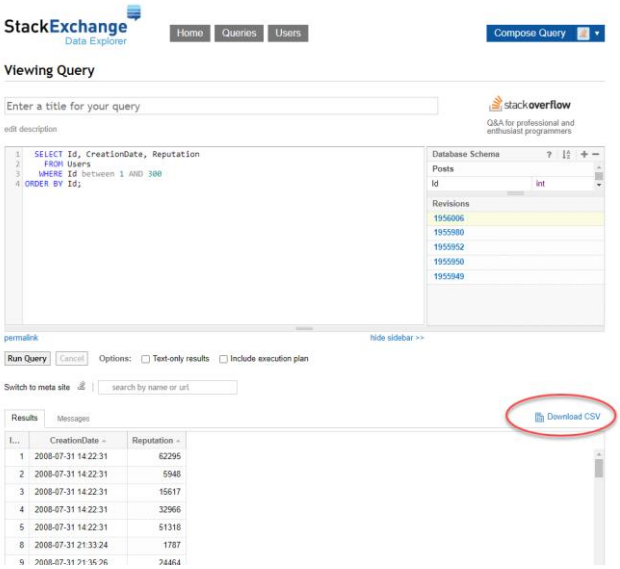


Figure 6.14 - SEDE Tool Screenshot

```
Table "SO_COMMENTS":
  48609 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                  82560 bytes(64 rows)
Read   buffer bytes: 1048576

Total logical records skipped:          1
Total logical records read:         48609
Total logical records rejected:         0
Total logical records discarded:        0

Run began on Sun Jun 05 18:38:57 2022
Run ended on Sun Jun 05 18:46:45 2022

Elapsed time was:     00:07:47.96
CPU time was:         00:00:00.28
```

Figure 6.15 - Comments Data Load Log

```
Table "SO_USERS":
  236 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                  49536 bytes(64 rows)
Read   buffer bytes: 1048576

Total logical records skipped:          1
Total logical records read:           236
Total logical records rejected:         0
Total logical records discarded:        0

Run began on Sun Jun 05 18:38:30 2022
Run ended on Sun Jun 05 18:38:38 2022

Elapsed time was:     00:00:07.73
CPU time was:         00:00:00.04
```

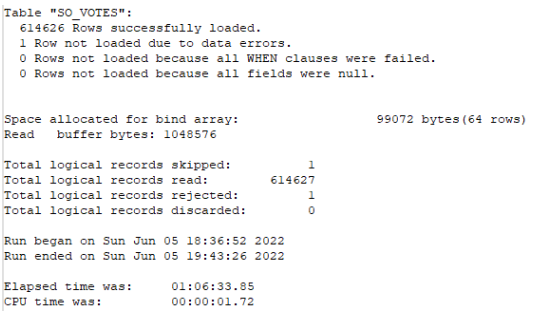
Figure 6.16 - Users Data Load Log

```
Table "SO_POSTS":
  33327 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                  82560 bytes(64 rows)
Read   buffer bytes: 1048576

Total logical records skipped:          1
Total logical records read:         33327
Total logical records rejected:         0
Total logical records discarded:        0

Run began on Sun Jun 05 18:32:39 2022
Run ended on Sun Jun 05 18:36:24 2022

Elapsed time was:     00:03:45.63
CPU time was:         00:00:00.25
```

Figure 6.17 - Posts Data Load Log

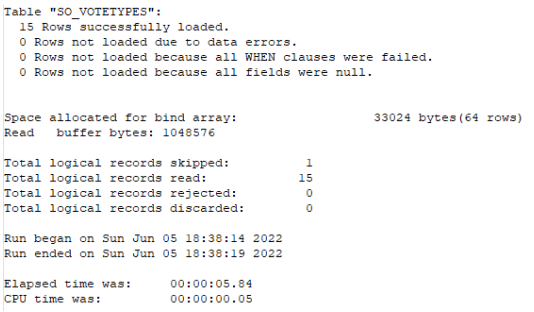
```
Table "SO_POSTTYPES":
  8 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                  33024 bytes(64 rows)
Read   buffer bytes: 1048576

Total logical records skipped:          1
Total logical records read:             8
Total logical records rejected:         0
Total logical records discarded:        0

Run began on Sun Jun 05 18:38:00 2022
Run ended on Sun Jun 05 18:38:06 2022

Elapsed time was:     00:00:06.49
CPU time was:         00:00:00.06
```

Figure 6.18 – Post Types Data Load Log



```
Table "SO_VOTES":
  614626 Rows successfully loaded.
  1 Row not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                 99072 bytes(64 rows)
Read   buffer bytes: 1048576

Total logical records skipped:          1
Total logical records read:        614627
Total logical records rejected:         1
Total logical records discarded:        0

Run began on Sun Jun 05 18:36:52 2022
Run ended on Sun Jun 05 19:43:26 2022

Elapsed time was:     01:06:33.85
CPU time was:         00:00:01.72
```

Figure 6.19 – Votes Data Load Log



```
Table "SO_VOTETYPES":
  15 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.


Space allocated for bind array:                 33024 bytes(64 rows)
Read   buffer bytes: 1048576

Total logical records skipped:          1
Total logical records read:            15
Total logical records rejected:         0
Total logical records discarded:        0

Run began on Sun Jun 05 18:38:14 2022
Run ended on Sun Jun 05 18:38:19 2022

Elapsed time was:     00:00:05.84
CPU time was:         00:00:00.05
```

Figure 6.20 – Vote Types Data Load Log

SQL*Loader Control File	Input File	Volume	Database Table	Volume	Execution Time (mins)
Comments.ctl	Comments.csv	48610	SO_COMMENTS	48609	8
Posts.ctl	Posts.csv	33328	SO_POSTS	33327	4
PostTypes.ctl	PostTypes.csv	9	SO_POSTTYPES	8	~0
Users.ctl	Users.csv	237	SO_USERS	236	~0
Votes.ctl	Votes.csv	614627	SO_VOTES	614626	66
VoteTypes.ctl	VoteTypes.csv	16	SO_VOTETYPES	15	~0

Table 6.5 – Oracle Database Data Loading Stats