A PROJECT REPORT ON

DUPLICATE BUG REPORT DETECTION

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN THE FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

OF

BACHELOR OF ENGINEERING

IN

COMPUTER ENGINEERING

SUBMITTED BY

OMKAR AMILKANTHWAR	**B150054211**
ANIRUDDHA DESHMUKH	**B150054212**
ATHARVA SATPUTE	**B150054454**
PRANAV DESHMUKH	**B150054271**

UNDER THE GUIDANCE OF
PROF. PREETI A. JAIN

DEPARTMENT OF COMPUTER ENGINEERING

PUNE INSTITUTE OF COMPUTER TECHNOLOGY

DHANKAWADI, PUNE – 43

SAVITRIBAI PHULE PUNE UNIVERSITY
2021-2022

CERTIFICATE

This is to certify that the project report entitles

DUPLICATE BUG REPORT DETECTION

SUBMITTED BY

OMKAR AMILKANTHWAR	**B150054211**
ANIRUDDHA DESHMUKH	**B150054212**
ATHARVA SATPUTE	**B150054454**
PRANAV DESHMUKH	**B150054271**

are bonafide student of this institute and the work has been carried out by them under the supervision of **Prof. Preeti A. Jain** and it is approved for the fulfillment of the requirement of Savitribai Phule Pune University, for the award of the degree of **Bachelor of Engineering** (Computer Engineering) .

Prof. Preeti A. Jain
Guide
Dept. of Computer
Engineering

Prof. Dr. Geetanjali V. Kale
Head
Dept. of Computer
Engineering

Prof. Dr. R. Sreemathy
Principal
Pune Institute of Computer Technology

Place: Pune
Date:

ACKNOWLEDGEMENT

It gives us great pleasure in presenting the project report on **'Duplicate Bug Report Detection'**.

We would like to take this opportunity to thank **Prof. Preeti A. Jain** for giving us all the help and guidance we needed. We are really grateful to her for her kind support. Her valuable suggestions were very helpful.
We are also grateful to **Prof. Dr. Geetanjali V. Kale**, Head of Computer Engineering Department, PICT for her indispensable support, suggestions.

Omkar Amilkanthwar
Aniruddha Deshmukh
Atharva Satpute
Pranav Deshmukh

ABSTRACT

The process of finding a duplicate bug report in the bug tracking system is known as duplicate bug report detection. To avoid repetition of work and rediscovery, this procedure is essential. The bug tracking systems receive thousands of duplicate bug reports each day. As a result, human costs, effort, and time rise. As a result, it is a significant issue in the software management process. Because bug reports are usually written in natural human language, the same bug might be described in a variety of ways. The approach is to automate the duplicate bug report detection mechanism in order to reduce manual work and thereby boost developer productivity. Streamlining the software management process reduces the cost of software maintenance as well. Existing systems, however, aren't nearly precise yet, despite the fact that they employ a variety of machine learning techniques. In this study, we are implementing a research paper to suggest comparable errors. In addition to using an information retrieval strategy with word embedding techniques, the method also takes into account the error title, description, defective product, and component information. For evaluation purposes, we used datasets from Eclipse and Mozilla, two popular open source projects.

Keywords: Skip-Gram, Prototype, TF-IDF, Cosine Similarity

Contents

LIST OF ABBREVIATIONS v

LIST OF TABLES vi

LIST OF FIGURES vii

1 INTRODUCTION 1
- 1.1 Introduction 2
- 1.2 Motivation 2
- 1.3 Problem Definition & Objectives 3
  - 1.3.1 Problem Definition 3
  - 1.3.2 Objectives 3
- 1.4 Project Scope & Limitations 3
  - 1.4.1 Project Scope 3
  - 1.4.2 Limitations 3

2 LITERATURE SURVEY 4
- 2.1 Automated Duplicate Bug Reports Detection 6
- 2.2 Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports 6
- 2.3 Automated Duplicate Bug Reports Detection 6
- 2.4 Towards More Accurate Retrieval of Duplicate Bug Reports 6

3 SOFTWARE REQUIREMENTS SPECIFICATION 7
- 3.1 Assumptions and Dependencies 8
  - 3.1.1 Assumptions 8
  - 3.1.2 Dependencies 8
- 3.2 User Classes and Characteristics 8
- 3.3 Functional requirements 9
  - 3.3.1 Corpus Creation for Model Training 9
  - 3.3.2 Preprocessing of Text Data 9
  - 3.3.3 Similarity Scores 10
- 3.4 External interface requirements 12
  - 3.4.1 Hardware Interfaces 12
  - 3.4.2 Software Interfaces 12
  - 3.4.3 Communications Interfaces 12
- 3.5 Non-functional requirements 13
  - 3.5.1 Performance Requirements 13

3.5.2 Software Quality Attributes . . . 13
3.6 System requirements . . . 14
3.6.1 Data Requirements . . . 14
3.6.2 User device requirements . . . 14
3.6.3 Software and Platform requirements . . . 14
3.6.4 Hardware Requirements . . . 14
3.7 Analysis Models: SDLC Model to be applied . . . 15

4 PROJECT PLAN 17
4.1 Project Estimates . . . 18
4.1.1 Reconciled Estimates . . . 18
4.1.2 Human Resources . . . 18
4.1.3 Development Resources . . . 18
4.2 Risk Management . . . 19
4.2.1 Risk Identification & Analysis . . . 19
4.2.2 Gantt Chart . . . 20

5 SYSTEM DESIGN 21
5.1 System Architecture . . . 22
5.2 Data Flow Diagram . . . 23
5.3 ER Diagram . . . 25
5.4 Usecase Diagram . . . 26
5.5 Class Diagram . . . 27
5.6 Activity Diagram . . . 28
5.7 Sequence Diagram . . . 29

6 PROJECT IMPLEMENTATION 30
6.1 Overview of the Project Modules . . . 31
6.2 Tools and Technologies Used . . . 32
6.3 Algorithm Details . . . 32
6.3.1 TF-IDF . . . 32
6.3.2 Word Embeddings . . . 32
6.3.3 Word2Vec . . . 32
6.3.4 Skip Gram Model . . . 33

7 SOFTWARE TESTING 34
7.1 Types of Testing . . . 35
7.2 Test Cases & Test Result . . . 37

8 RESULTS 38
8.1 Outcomes . . . 39
8.2 Screenshots . . . 39

9 CONCLUSION & FUTURE WORK 42
9.1 Conclusion . . . 43
9.2 Future Scope . . . 43

APPENDIX A MATHEMATICAL FORMULAE 44

APPENDIX C PLAGIARISM REPORTS **46**

REFERENCES **49**

List of Abbreviations

Abbreviation	Full form
UI	User Interface
DL	Deep Learning
NLP	Natural Language Processing
API	Application Programming Interface
SDLC	Software Development Life Cycle
UML	Unified Modeling Language
DBRD	Duplicate Bug Report Detection
TF-IDF	Term Frequency-Inverse Document Frequency
HTTP	Hyper-Text Transfer Protocol
REST	Representational State Transfer
GCP	Google Cloud Platform
AWS	Amazon Web Services

Table 1: Abbreviations

List of Tables

1 Abbreviations v

2.1 Literature survey on various approaches for Duplicate Bug Report Detection 5

4.1 Risk Probability Levels 19
4.2 Risk Impact Severity 19

7.1 Test Cases 37

List of Figures

3.1 Corpus Building Workflow 9
3.2 Text Processing Workflow 9
3.3 TF-IDF 10
3.4 Skip-Gram model architecture 11
3.5 Workflow 16

4.1 Gantt Chart(expected) 20
4.2 Gantt Chart(Planned) 20

5.1 System Architecture Diagram 22
5.2 Context Diagram 23
5.3 DFD Level 1 23
5.4 DFD Level 2 24
5.5 ER Diagram 25
5.6 Usecase Diagram 26
5.7 Class Diagram 27
5.8 Activity Diagram 28
5.9 Sequence Diagram 29

8.1 Home Page 39
8.2 Search Bug ID 40
8.3 File Upload 40
8.4 Comparing Reports 41

CHAPTER 1

INTRODUCTION

1.1 Introduction

A bug repository is an important aspect of many open-source projects and corporate projects for development and maintenance, which enables both developers and users to report bugs, request more efficient features, and suggest the flaws present. Using these bug repository has many but least two major advantages. First, bug repositories allow users around the world to act as "testers" for your software, increasing your chances of discovering problems and, as a result, improving the quality of your software. Second, it allows the software to grow in response to user demands, which allows it to meet the needs of more users.

However, these benefits come at a price. This type of test is asynchronous and poorly structured because the project relies on a large number of users as testers. In addition, the cost of a user checking the repository (checking if the issue has been resolved) is higher than the cost of creating a new bug report. Therefore, some reported issues are duplicates previously reported errors also there may be same bug reported multiple times. To avoid this, each bug report submitted should be checked to make sure there are duplicates.

Due to the large number of existing error reports, it is difficult to investigate all the present error reports to find duplicates, manually.

1.2 Motivation

Using bug tracking tools, organizations can manage test reports more efficiently and deliver high quality products with reduced development costs and higher customer satisfaction. It is common for all testers to encounter the same bug, resulting in duplicate bugs being reported in the system. These duplicate bugs can be solved if common code files are handled properly.In light of this, duplicate bug report detection is a critical process that can significantly improve development quality and efficiency. It also helps organizations provide their customers with better services.

1.3 Problem Definition & Objectives

1.3.1 Problem Definition

Developers rely on bug reports to fix bugs. Typically, bug reports are stored and managed via bug tracking software. There is a high possibility that many testers may encounter same bug as a result of which bug tracking systems may contains many duplicate bug reports. Thus implement a tool for effective detection of duplicate bug reports so as to reduce time and cost of operation and human resources.

1.3.2 Objectives

- To survey available techniques and approaches that efficiently detects duplicates bugs reports.
- APIs to predict whether new bug reported are already present in the system or not.
- To expose the database through an intuitive, friendly UI.

1.4 Project Scope & Limitations

1.4.1 Project Scope

- The Project includes the training of Skip-Gram model on Eclipse's Duplicate Bug Dataset
- The project focuses on implementation of research paper to achieve state-of-the-art performance on Duplicate Bug Report Detection Task.
- The project aims to explore all the different obstacles that are likely to be faced for a production ready system and test the effectiveness of existing strategies like model pruning and continuous training.

1.4.2 Limitations

- Skip-gram model needs to be trained on huge corpus of words.
- While the end goal of our efforts is to build a extension for Bug Tracking Tools for firms, the scope of this project is limited to research and development rather than deployment.
- There is direct relationship between data and resulting accuracy.
- The system is not highly scalable as lot of computation power is required for Word2Vec task. It is limited within firms only.

CHAPTER 2

LITERATURE SURVEY

No.	Title	Keywords	Dataset	Summary
1	Automated Duplicate Bug Reports Detection, 2017. [1]	Similar Bugs, Paragraph Vector, Information Retrieval	Eclipse, Mozilla, Firefox, Open Office	The paper examines and identifies proposed evaluation methods for identifying automated duplicate bug report detection methods.
2	Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports, 2016. [2]	Skip-Gram [3], Word2vec, TF-IDF	Duplicate Bug Datasets MSR 2014	The paper suggested a way by making use of word embedding and information retrieval technique to calculate similarity score between two bug reports.
3	Towards More Accurate Retrieval of Duplicate Bug Reports, 2011. [4]	Duplicate Bug, BM25F	OpenOffice, Firefox, Eclipse	A retrieval function (REP) is proposed as a way to quantify the similarity between two bug reports in order to properly identify them as duplicates.
4	Duplicate Bug Report Detection and Classification System Based on Deep Learning Technique, 2020. [5]	Feature Extraction, Deep Learning	Mozilla, Eclipse, NetBeans	Convolutional Neural Networks based architecture is used to extract information from the bug reports.

Table 2.1: Literature survey on various approaches for Duplicate Bug Report Detection

2.1 Automated Duplicate Bug Reports Detection

- Using a systematic mapping analysis, this paper determines and classifies 44 publications related to duplicate bug report detection.
- Using a bug tracking tool along with duplicate bug report detection module, the list of Top-N related bugs is thus created with a number value indicating the similarity between them.
- The combined use of Doc2vec and TF-IDF produces the best results.

2.2 Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports

- In addition to the bug title and description, product and component information is taken into account in the scoring process.
- Eclipse and Mozilla datasets are used.
- Reported significant performance improvement over traditional methods.

2.3 Automated Duplicate Bug Reports Detection

- TOP N recommendation and ranking strategy, binary classification approach, and decision-making approach are the three categories of automated duplicate bug report identification systems.
- To test the proposed method, an experiment is carried out with a identified number of bug reports from Axis Communication AB's internal bug report database.

2.4 Towards More Accurate Retrieval of Duplicate Bug Reports

- Detailed analyses of the top-k related bug reports were presented for each new bug report under investigation.
- By extending BM25F, a model for retrieving duplicate bug reports was proposed.
- Compared to other retrieval measures in the literature, the proposed retrieval method has textual and categorical similarities that outperform state-of-the-art methods.

CHAPTER 3

SOFTWARE REQUIREMENTS SPECIFICATION

3.1 Assumptions and Dependencies

3.1.1 Assumptions

1. The bugs report uploaded should be valid, fields in report should not blank which in case the system will reject the report.

2. The bug report should be uploaded in json format with standard character encodings, no other formats such as XML are allowed.

3.1.2 Dependencies

1. The project has taken into consideration non-relational database eg. MongoDB and relational database e.g. SQLite3

2. The system logic is written in Python, and need Python version 3.8 or higher installed to run, along with the following dependencies

   - nltk 3.6.5
   - Tensorflow 2.7.0
   - Scikit learn 1.0.1
   - Keras 2.7.0
   - Pymongo 3.11.0
   - Flask 2.0.2
   - ReactJS 17.0.2

3. The backend application runs on a flask server with UI in ReactJS

3.2 User Classes and Characteristics

Users: Users will have to search for bug ID to find reports similar to the one user searched.

3.3 Functional requirements

3.3.1 Corpus Creation for Model Training

a. Description: To train Skip-Gram Word-to-Vec model, a corpus is needed. To keep the corpus relevant to the type of data the project is dealing with, the corpus can be constructed from the existing Bug Datasets. The process includes concatenation of descriptions from bug reports after preprocessing techniques which include tokenization, lower casing, stop-word removal, stemming, and lemmatization.

b. Sequence Diagram:

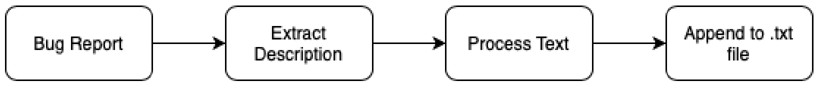


Figure 3.1: Corpus Building Workflow

c. Functional Requirements:

i. The corpus should be in plain text format with .txt file extension.

ii. The sentences in corpus should be meaningful for analyzation of the context.

iii. The corpus must be huge to cover wide range of words.

3.3.2 Preprocessing of Text Data

a. Description: Data preprocessing is an important stage in developing a machine learning model, and the outcomes are dependent on how well the data has been handled. Text preprocessing techniques include tokenization, lower casing, stop-word removal, stemming, and lemmatization. Dimensionality reduction techniques like this are extensively employed.

b. Response sequence:

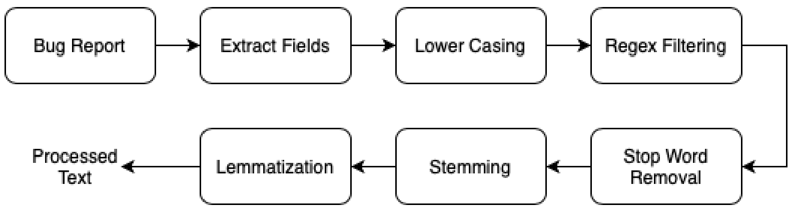


Figure 3.2: Text Processing Workflow

c. Functional requirement:

i. For users to have access to system, they should have an authenticated account.

ii. To be able to send images back and forth between server and client, there must be an active internet connection.

iii. Response should be real time

3.3.3 Similarity Scores

The similarity score indicates how similar two data objects are. It is the distance with dimensions reflecting features of the objects in the context of data mining or machine learning. When the distance between the characteristics is short, the features are considered to be with greater similarity. A large distance, on the other hand, will result in a low degree of similarity.
Skip-Gram model is used to calculate score2. A hugs corpus is needed to train the skip-gram model. This corpus can be made by combining some fields in existing bug report in database such as title, description, etc. With this approach, since the corpus is related to our specific task, there might be increase in accuracy while predicting duplicate bug reports.

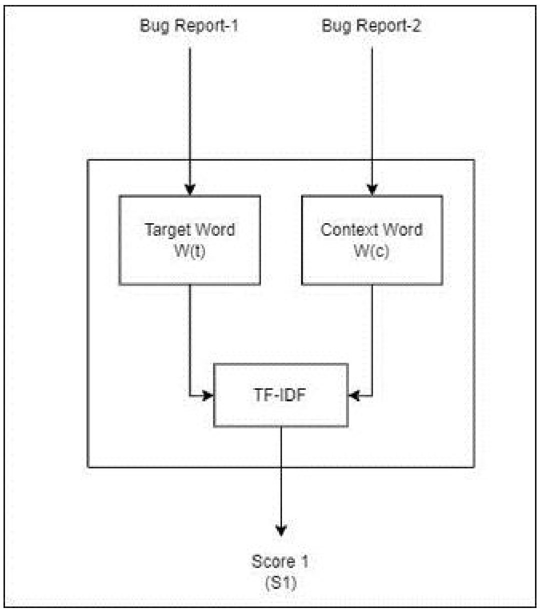


Figure 3.3: TF-IDF (Score 1)

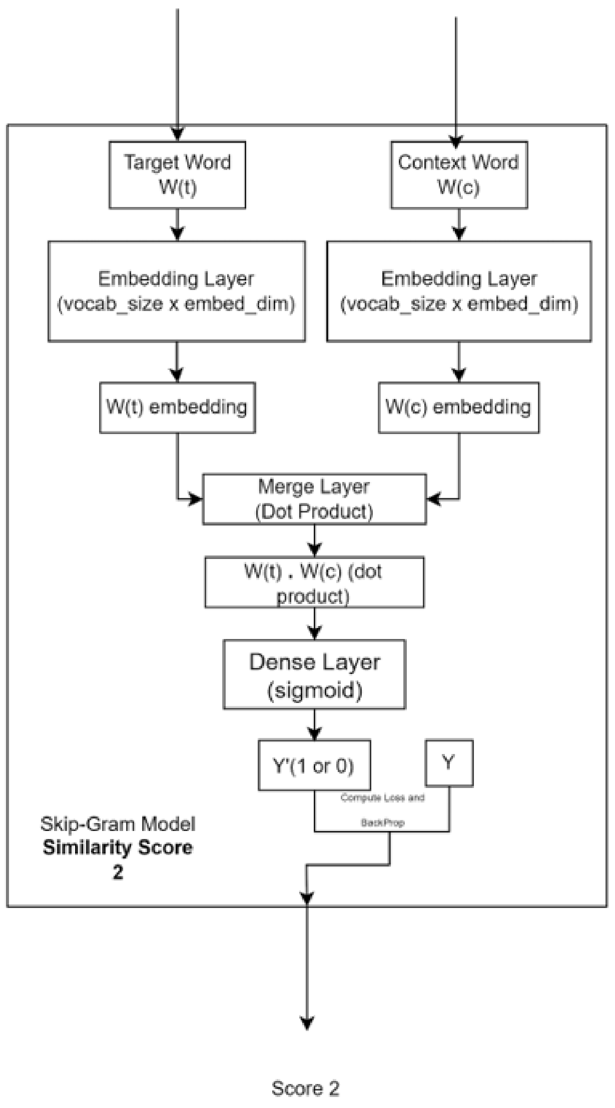


Figure 3.4: Skip-Gram model architecture (Score 2)

3.4 External interface requirements

3.4.1 Hardware Interfaces

- The server side code would interact directly with specialized hardware.
- On the client side, for uploading bug reports, decent system is required.

3.4.2 Software Interfaces

- The client side web-app interacts with the user directly through ans easy GUI allowing user to have a look at status of bugs in database and uploading newer bug reports. Thus, the UI id divided into two interfaces, the visualization and query interface. This web-app is create using React.js.
- Server side application consists of inference engine. The engine is created using Flask framework, which interacts with a skip-gram model trained and run using the TensorFlow Deep Learning libraries.
- To manage unique and duplicate bug reports, MongoDB database will be used.

3.4.3 Communications Interfaces

- The product will have a simple HTTP based interface between the client and server side applications. All server endpoints will be exposed as secure, authenticated REST endpoints, through which the front end application can communicate and access the models.
- The product will have a REST APIs with which the System can be integrated with existing Bug Tracking Tools.

3.5 Non-functional requirements

3.5.1 Performance Requirements

Minimum 8/16 GB RAM required (for small datasets) and 32 GB or higher required (for large datasets) for preprocessing all the documents.

3.5.2 Software Quality Attributes

Software Quality Attributes help Software engineers to assess the exhibition of a product application. These quality credits conclude whether or not the product is of good quality.

The following are some key quality characteristics that will be important to objectively judge the end result and feasibility of the product:

- Extensibility : Extensibility is a measure of a system's ability to be extended and the amount of effort necessary to do so. To scale the project, some optimization techniques such as Database sharding etc. can be applied.
- Reliability : It is a system attribute that determines the system's ability to continue operating under predetermined conditions. External elements such as databases, systems, and network connections are frequently inaccessible, causing the DBRD system to fail.
- Accuracy : Accuracy of DBRD system depends upon value of similarity scores calculated. More is the value of score, more is the similarity.
- Robustness : To ensure the factor of Robustness system will be tested multiple number of times with error prone inputs. The aim while testing the system will be to cover different test cases in implementation.
- Maintainability : Maintainability is the ease with which a product can be maintained to correct defects and their cause, Repair or replace faulty components, etc. The code of system will be modular and API driven and will follow all object oriented paradigms. This will enhance ability to fix bugs and maximize its lifespan. Bugs related to each modules can be handled separately.
- Vulnerability: A vulnerability in an asset or collection of assets that can be exploited by one or more cyber threats, where an asset is defined as everything of value to the organisation, its business operations, and their continuity, including information resources that support the organization's goal. Different checks will be done before we make any permanent change in system or database. This will make system less vulnerable.

3.6 System requirements

3.6.1 Data Requirements

- Large amount of data is required to correctly analyse system's accuracy.
- We primarily use the Eclipse dataset which has 423559 bug reports including duplicates. The dataset has been provided with 271098 pairs of bugs flagged duplicate or not.

3.6.2 User device requirements

The user interface for the system will be an web-app written in ReactJS. Any browser with JavaScript enabled browser will be able to run the application.

3.6.3 Software and Platform requirements

- Python programming language and the related ecosystem of deep learning libraries
- Ubuntu 20.04/Windows as the primary OS of implementation
- nltk 3.6.5
- Tensorflow 2.7.0
- Scikit learn 1.0.1
- Keras 2.7.0
- Pymongo 3.11.0
- Flask 2.0.2
- ReactJS 17.0.2

3.6.4 Hardware Requirements

Sr. No.	Parameter	Minimum Requirements	Justification
1	CPU Speed	2 GHz or higher	Support the operations of the UI and APIs
2	RAM	8 GB or higher	For handling large size embeddings matrix.

3.7 Analysis Models: SDLC Model to be applied

This project uses the waterfall model. It entails following a series of stages. In a waterfall paradigm, each step must be completed in its entirety before moving on to the next. This software development paradigm is typically used for projects that are modest and have few ambiguous requirements.

- Throughout an organization, software projects are managed using the SDLC.
- In this document is detailed information about how the software will be developed, how it will be maintained and how specific parts of the software will be replaced, altered, or enhanced.
- Waterfall model: The Waterfall model was the first to be utilised as a process model. It's really easy to learn and use, and each phase must be finished before moving on to the next. It doesn't have any overlapping cycles. The waterfall model is divided into six stages.

1. **Requirement Analysis**
   Detailed requirements for the software system to be created are acquired during this phase. We explored many datasets to work with as well as the scope and application of our project. The various interpretability strategies were investigated.
2. **Design**
   This is the phase where the programming language is decided along with other technical details of the project. We developed the design of our project and decided how the various components of our application can be integrated. We decided the architecture and modules of our project.
3. **Implementation**
   Based on the design done in previous phase, coding of the system will be performed. Till now, approximately 20% of the system is implemented.
4. **Testing**
   In this phase, the developed software is verified that it is built as per the specifications mentioned and that it is working properly. API testing will be carried out.
5. **Deployment**
   The project can be deployed on services such as GCP, AWS, etc. for demonstration purpose. Client can make the required changes according to their needs and deploy it within their organization.
6. **Maintenance**
   Once the system is ready to use, additional changes may be required in the code, later as per requirements. In this case, if error is encountered, we will make sure to note and fix the issues. The deep learning model can be further optimized specific to the needs.

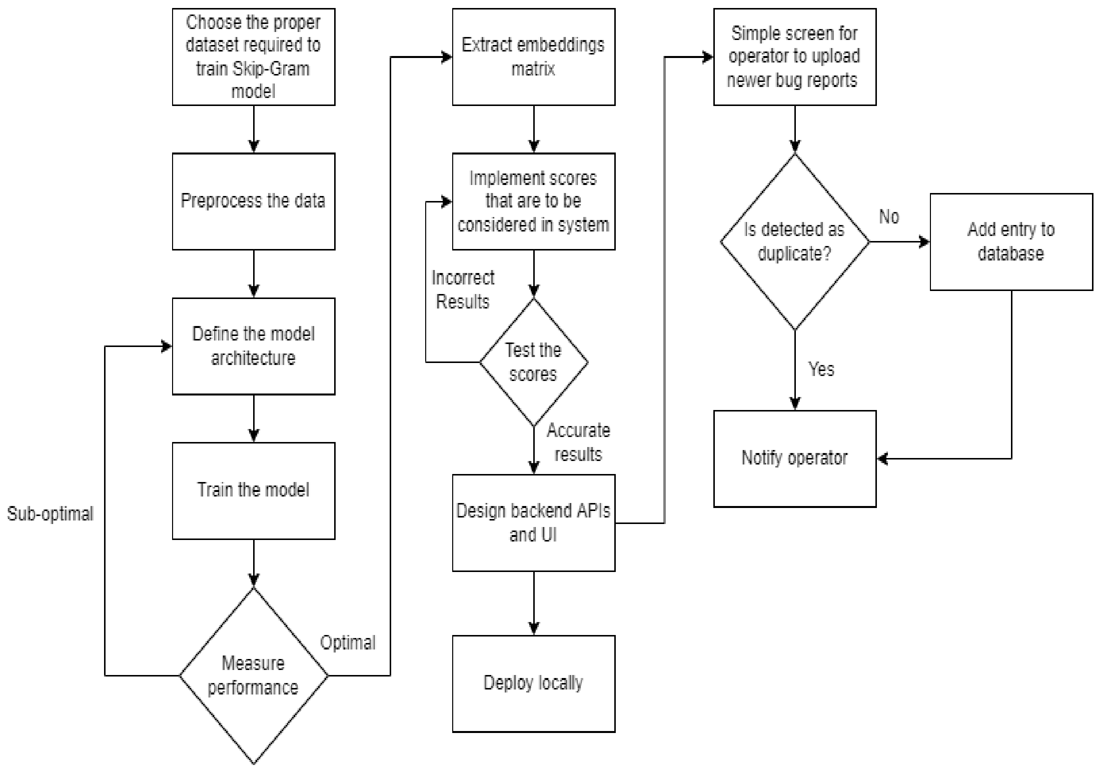

Choose the proper dataset required to train Skip-Gram model
Preprocess the data
Define the model architecture
Train the model
Measure performance
Sub-optimal
Optimal
Extract embeddings matrix
Implement scores that are to be considered in system
Incorrect Results
Test the scores
Accurate results
Design backend APIs and UI
Deploy locally
Simple screen for operator to upload newer bug reports
Is detected as duplicate?
No
Add entry to database
Yes
Notify operator

Figure 3.5: Waterfall Model Workflow

CHAPTER 4
PROJECT PLAN

4.1 Project Estimates

4.1.1 Reconciled Estimates

- Cost Estimate: The software used for the application will incur no cost as it is all released under free open source licenses.
- Time Estimate: 6 months

4.1.2 Human Resources

- Number of people : 4
- Skills: Python, Natural Language Processing, Machine Learning, React.
- Client: In our project, the users can be a firm using the product in SDLC
- Stakeholders: Stakeholders will be the people who will be using the application, i.e, the firms

4.1.3 Development Resources

- Data Collection: The datasets used for experiments have been released under open source licenses for R&D projects
- Software:
  1. Python libraries (Tensorflow, nltk, pyMongo)
  2. Flask framework
  3. ReactJS libraries

4.2 Risk Management

Risks are a natural part of developing applications. While risk is normally affiliated with negative connotations, a mature understanding of risk defines it as an essential element that helps corporate new opportunities and conquer new frontiers, that is, if risk is handled properly. Hence, the management of risk becomes an important part of developing an application.

4.2.1 Risk Identification & Analysis

The risk analysis is performed using the following guidelines where x is a occurrence of failure:

Risk Probability

a.	High Probability	$6\% \leq x \leq 100\%$
b.	Medium High Probability	$4\% \leq x \leq 5\%$
c.	Medium-Low Probability	$2\% \leq x \leq 3\%$
d.	Low Probability	$0\% \leq x \leq 1\%$

Table 4.1: Risk Probability Levels

Risk Impact

a. Lost connection with the database will be considered as catastrophic failure as system will stop working completely.

b. Failing POST API calls can be considered as critical failure and need to be intervene.

c. Failure in rendering UI components can be considered as moderate failure.

d. Key not found error for a word can be considered as marginal failure and 2-3 words can be directly skipped without affecting the accuracy.

	Failure Probability	Impact
a.	High	Marginal
b.	Medium High	Moderate
c.	Medium Low	Critical
d.	Low	Catastrophic

Table 4.2: Risk Impact Severity

4.2.2 Gantt Chart

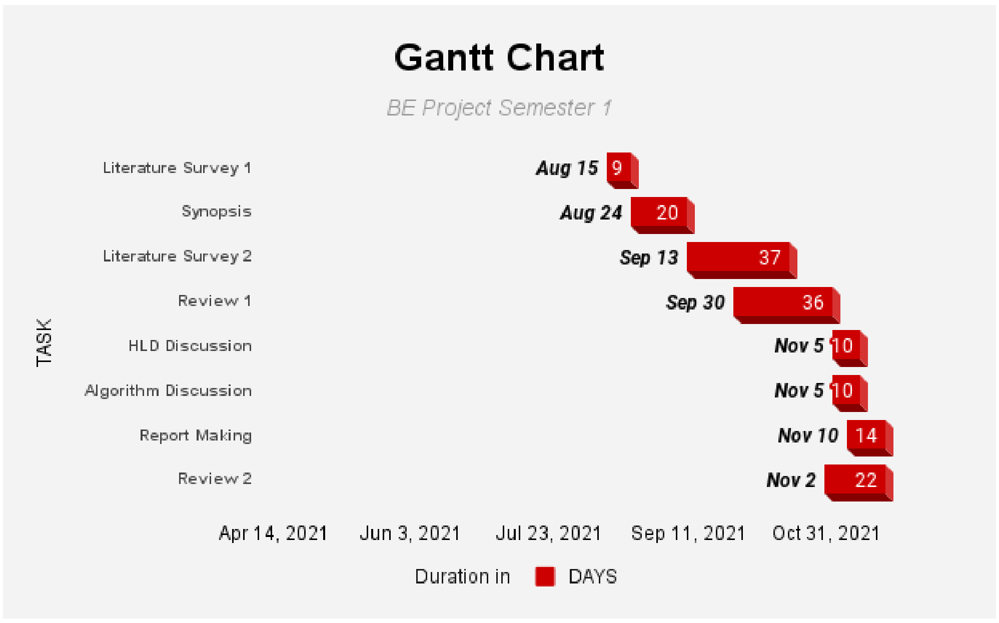


Figure 4.1: Gantt Chart(expected)

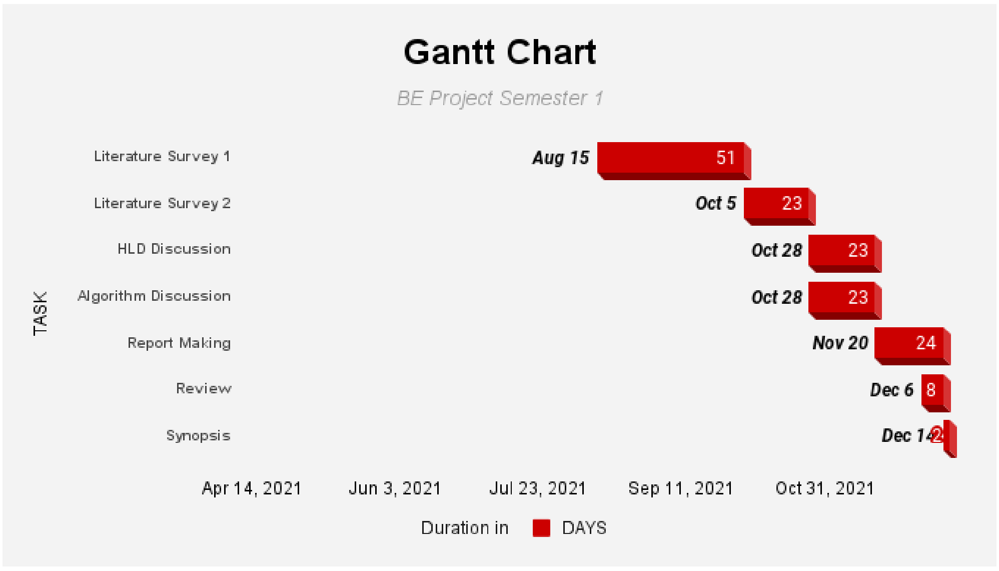


Figure 4.2: Gantt Chart(Planned)

CHAPTER 5

SYSTEM DESIGN

5.1 System Architecture

The high level system design shows how the different computational components, APIs and clients will be laid out.

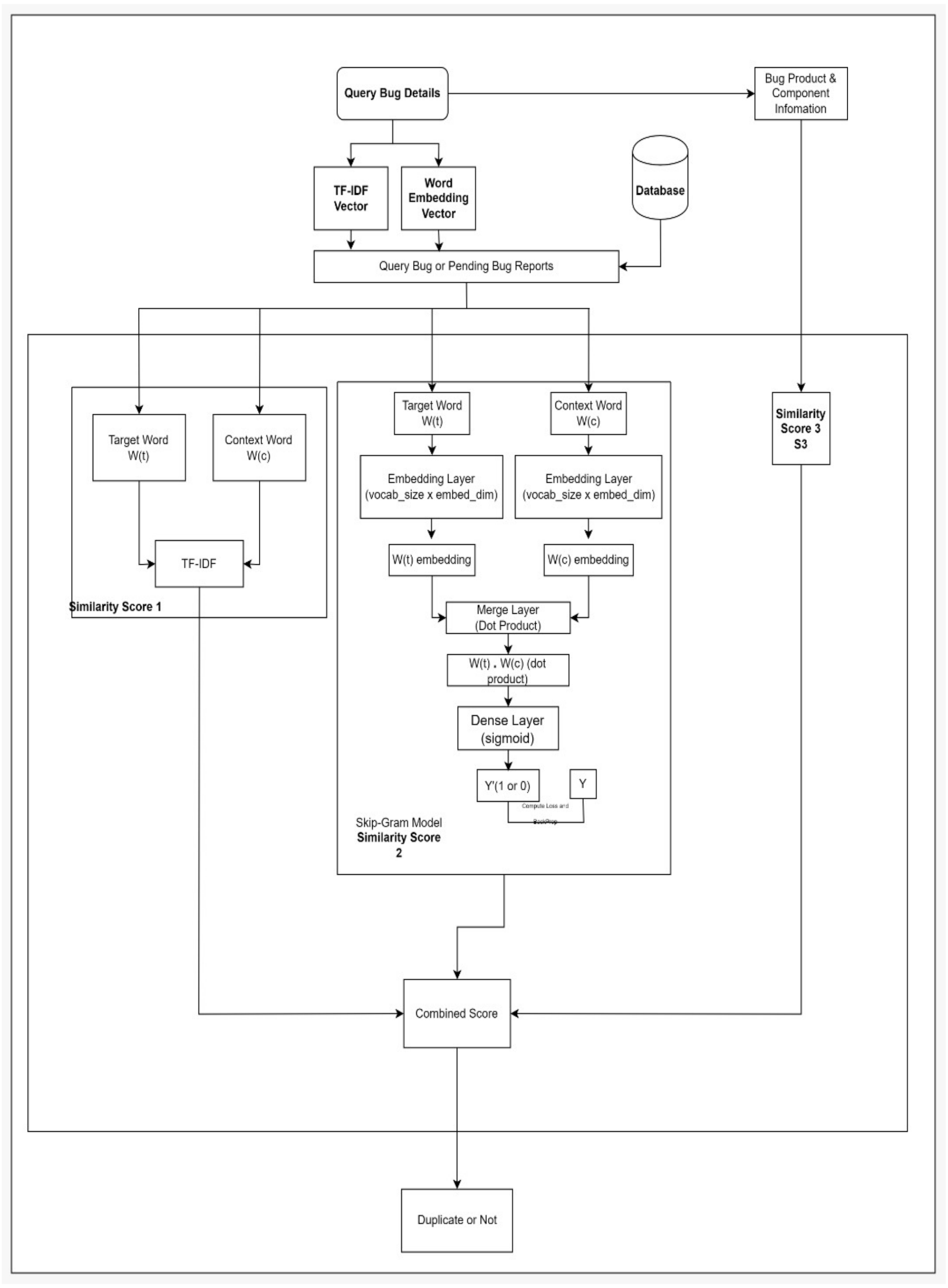


Figure 5.1: System Architecture Diagram

5.2 Data Flow Diagram

The data flow diagram illustrates the flow of data throughout the life time of the application, for either of the two main use cases.

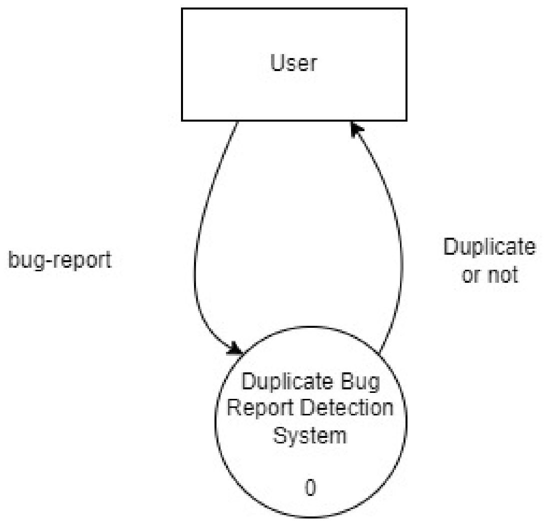


Figure 5.2: Context Diagram

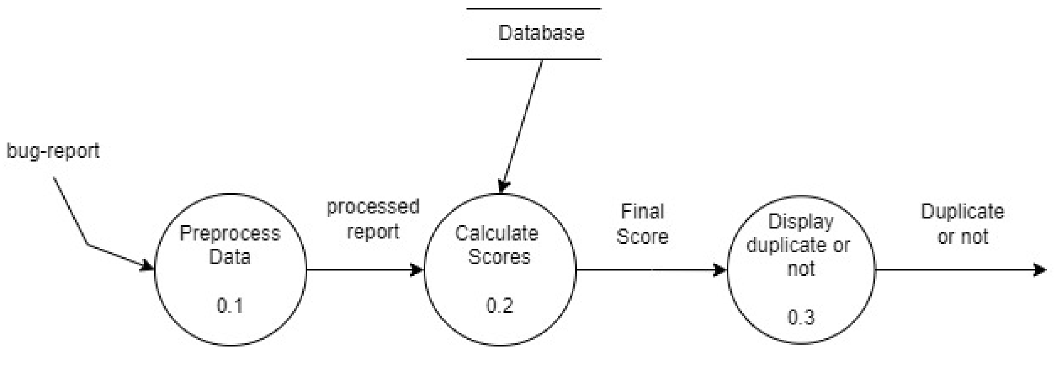


Figure 5.3: Level-1

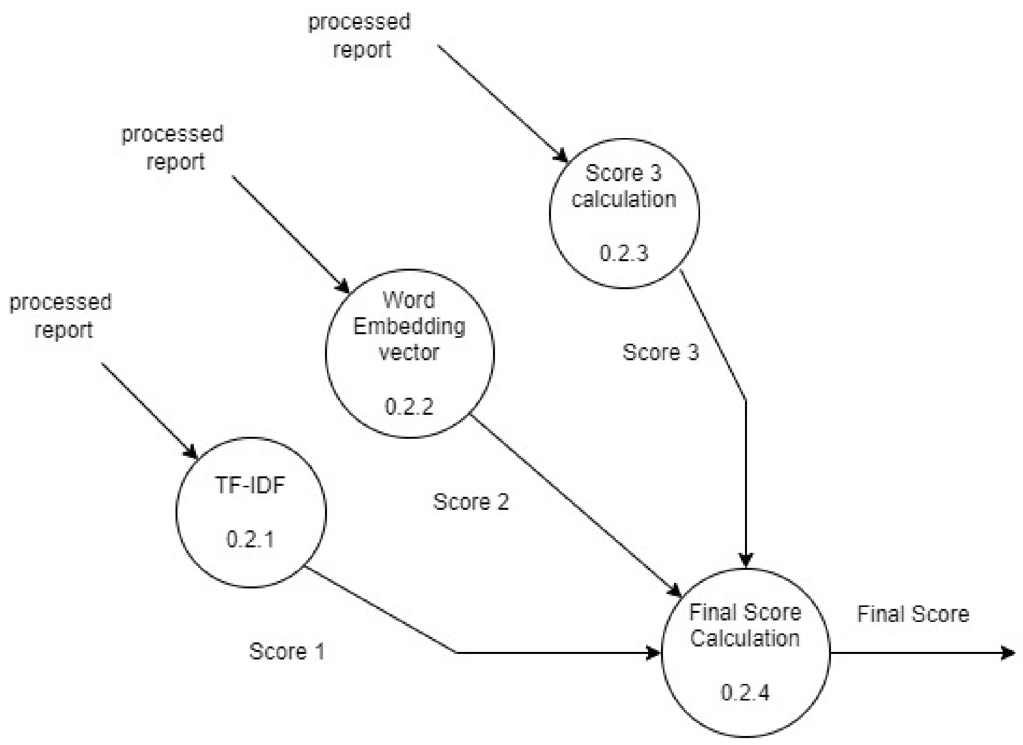


Figure 5.4: Level-2

5.3 ER Diagram

The ER diagram lays out the entire system as a collection of entities with properties and illustrates the relations between different entities

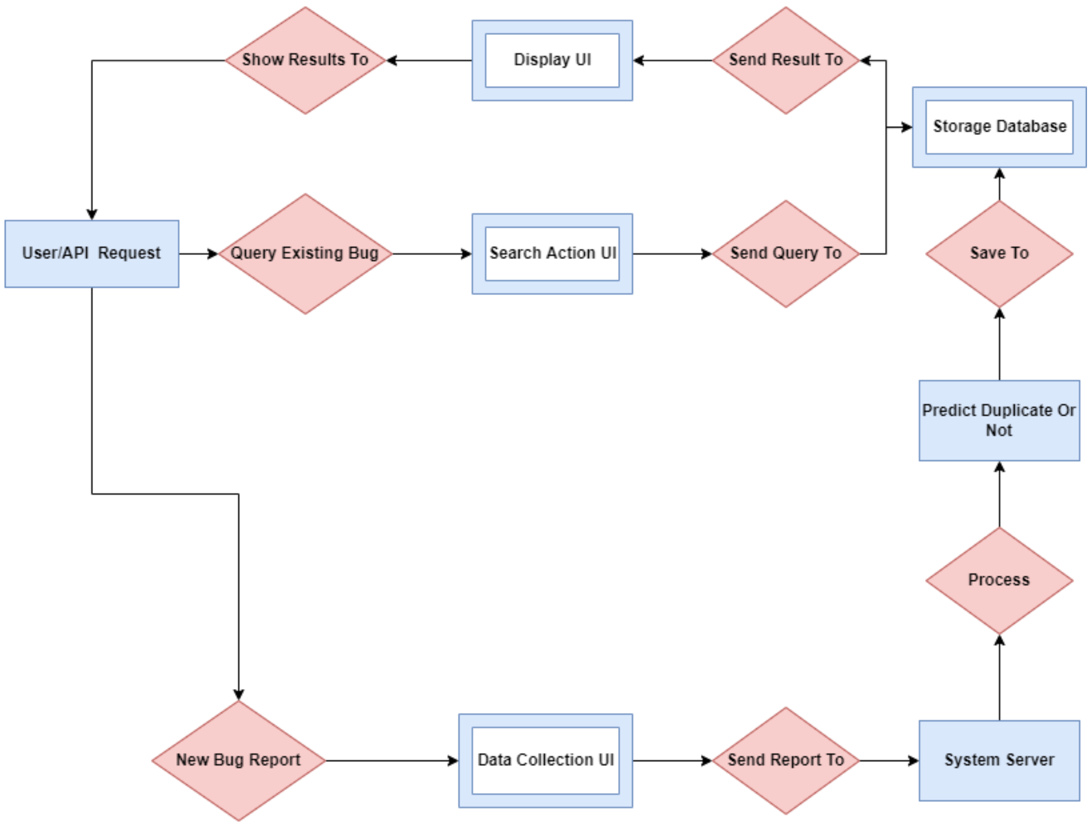


Figure 5.5: ER Diagram

5.4 Usecase Diagram

The major use cases of the application are illustrated here, with users as the actors interacting with different modules

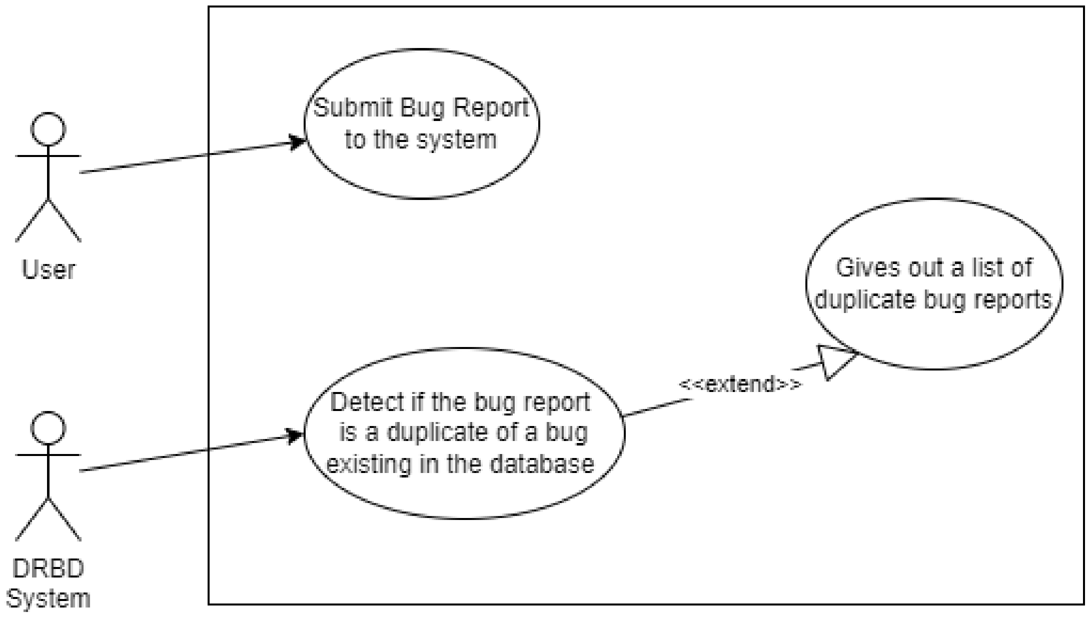

User
DRBD System
Submit Bug Report to the system
Detect if the bug report is a duplicate of a bug existing in the database
<<extend>>
Gives out a list of duplicate bug reports

Figure 5.6: Usecase Diagram

5.5 Class Diagram

The components of the application can be divided into three main components as illustrated below

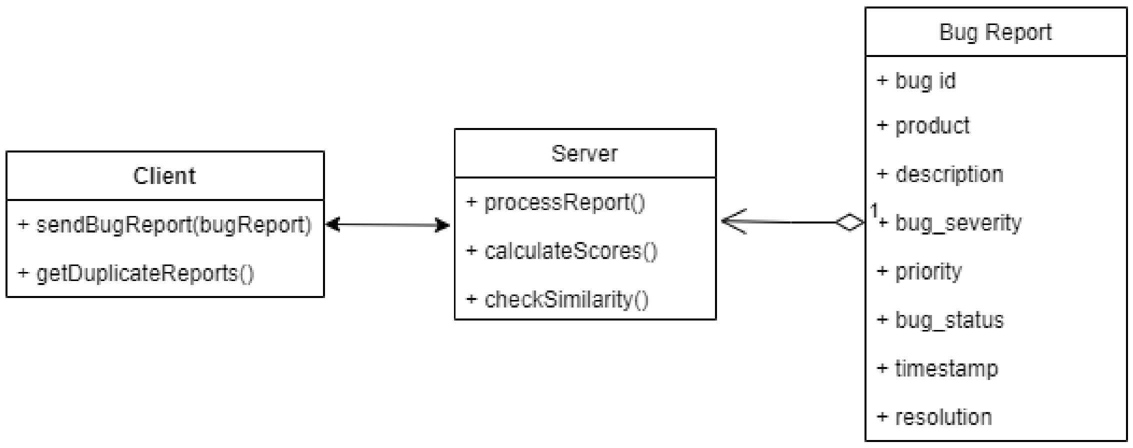


Figure 5.7: Class Diagram

5.6 Activity Diagram

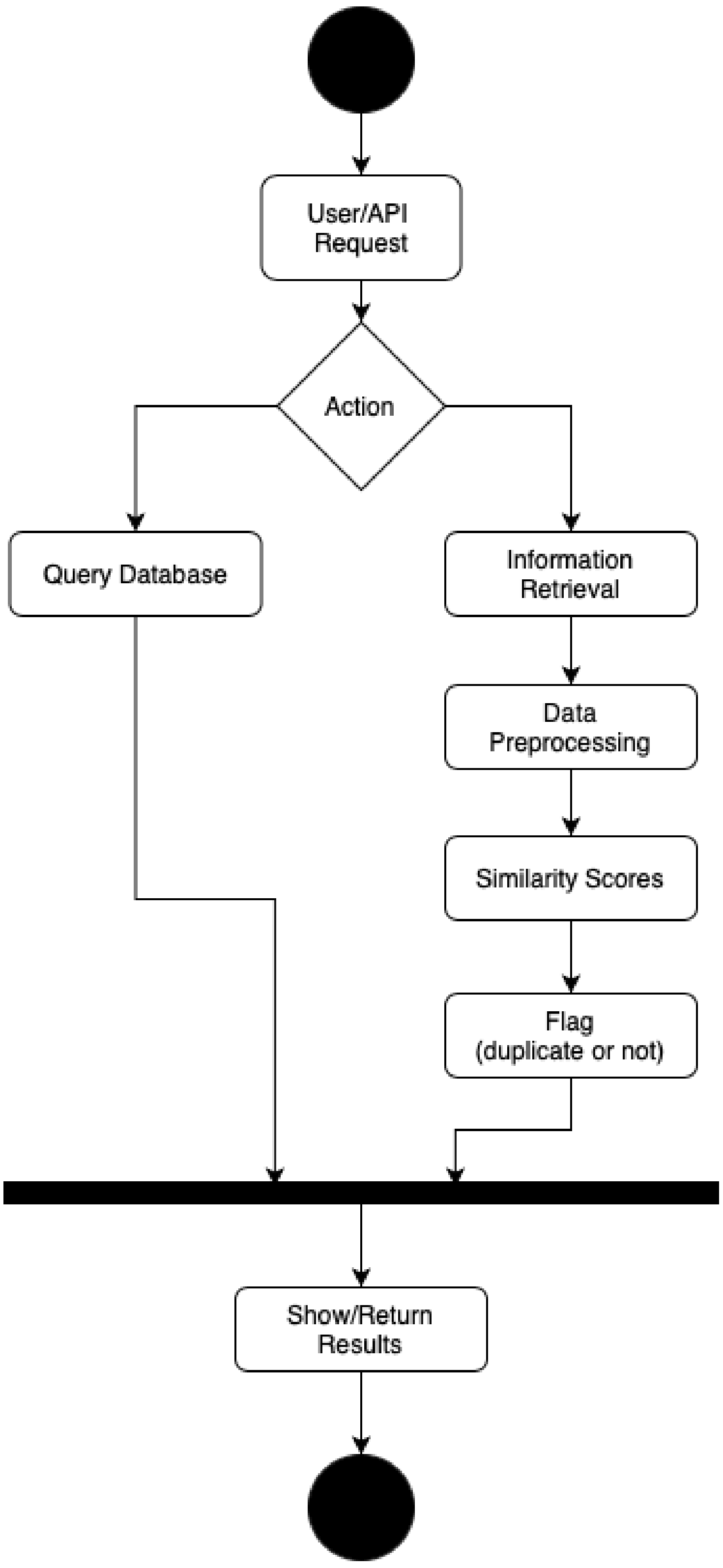


Figure 5.8: Activity Diagram

5.7 Sequence Diagram

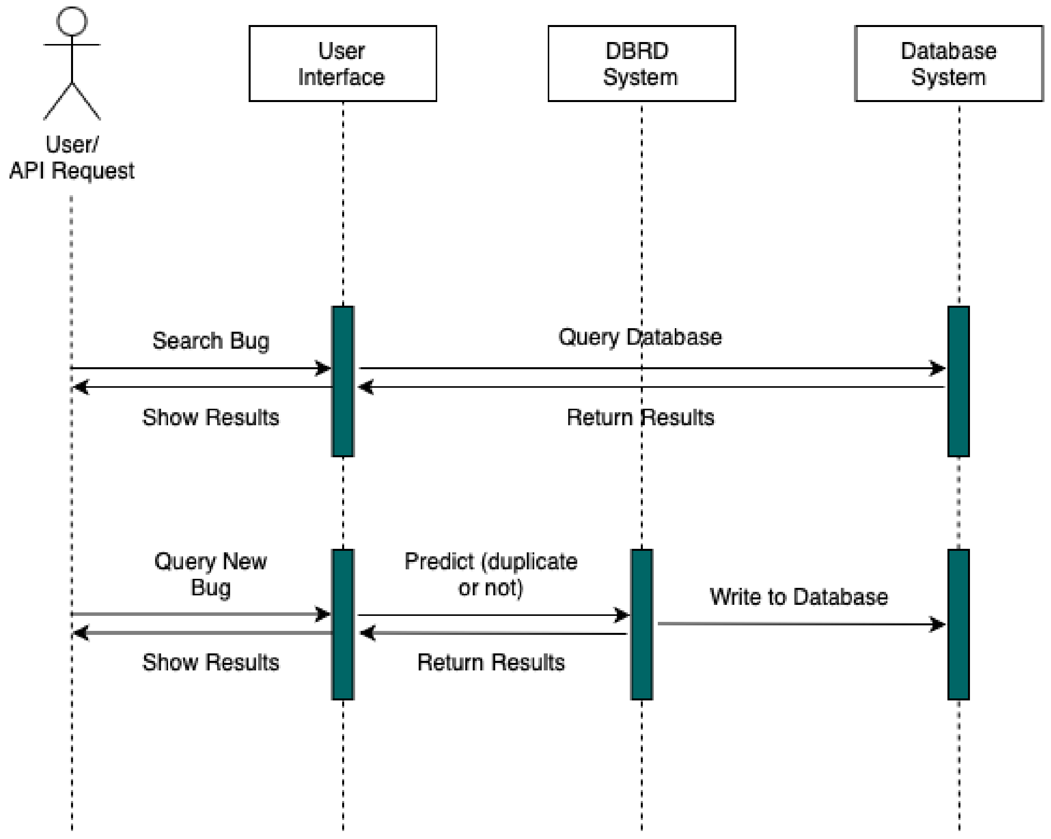


Figure 5.9: Sequence Diagram

CHAPTER 6

PROJECT IMPLEMENTATION

6.1 Overview of the Project Modules

1) Building Corpus

We considered Eclipse dataset for this project which contains over 423559 bug reports. We processed "Description" field in each of the bug report, concatenated them to build a corpus which is needed to train Word2Vec model to calculate one of the similarity scores.

2) Preprocessing of text

For converting the textual information into machine readable raw format, we imported the nltk library. We removed stopwords, removed punctuation and applied stemming. For transforming text to feature vectors, we used Skip-Gram Word2Vec Model.

3) Train Skip-Gram model

We trained a Skip-Gram model using implementation from the Gensim library with prepared corpus from the existing dataset.

4) UI Components

Our project have following UI components.

1. Search Bar: To query single Bug IDs.
2. File Chooser: To upload .csv file to query no. of. Bug IDs.
3. Split View: To compare Bug Reports side by side which are predicted to be of similar kind.

6.2 Tools and Technologies Used

1) Tools Used:

- 64 bit OS
- 8 GB RAM or higher
- VS Code/ Jupyter Notebook

2) Technologies Used

- Python
- MongoDB / SQLite3
- Javascript / React JS
- Machine Learning
- Natural Language Processing

6.3 Algorithm Details

6.3.1 TF-IDF

Term Frequency–Inverse Document Frequency(TF-IDF), is a numerical statistic used to get importance of the word in the document or text. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling to assign weightage to each of the word in a sample.

6.3.2 Word Embeddings

In natural language processing, word embedding is a term used to represent words for text analysis, typically in real-valued vectors that encode the word's meaning. The words that are closed in vector space are expected to have a similar meaning. Word embedding uses language modeling and feature learning techniques where words from the vocabulary are mapped to vectors of real numbers.

6.3.3 Word2Vec

Word2vec is a technique for natural language processing published in 2013. The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text.

Word2vec is a two-layer neural network that processes text by "vectorizing" words. Its input is a text corpus, and its output is a set of vectors. Feature vectors that represent words in that corpus. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence. As the name implies, word2vec represents each distinct word with a particular list of numbers called a vector. The vectors are chosen carefully such that a simple mathematical function (the cosine similarity between the vectors) indicates the level of semantic similarity between the words represented by those vectors.

Word2vec is not a single algorithm but a combination of the techniques mentioned above, i.e. CBOW(Continuous Bag of word) and Skip-Gram.

6.3.4 Skip Gram Model

The continuous skip-gram model learns by predicting the surrounding words given a current word. In other words, the Continuous Skip-Gram Model predicts words within a certain range before and after the current word in the same sentence. The Skip-Gram model is trained on n-gram pairs of (target_word, context_word) with a token as 1 and 0. The token specifies whether the context_words are from the same window or generated randomly. The pair with token 0 is neglected.

CHAPTER 7

SOFTWARE TESTING

7.1 Types of Testing

- Alpha Testing : It is the most common type of testing used in the Software industry. The objective of this testing is to identify all possible issues or defects before releasing it into the market or to the user. Alpha testing is carried out at the end of the software development phase but before the Beta Testing.

  Before deployment of Duplicate Bug Report Detection alpha testing is done at the end of software development life cycle. All features along with UI are tested by developers.

- Accepting Testing : An acceptance test is performed by the client and verifies whether the end to end the flow of the system is as per the business requirements or not and if it is as per the needs of the end user. Client accepts the software only when all the features and functionalities work as expected It is the last phase of the testing, after which the software goes into production. This is also called User Acceptance Testing (UAT).

  Veritas Technologies LLC has sponsored Duplicate Bug Detection System project. External guide from Veritas has tested system with all features and functionalities expected to them and gave clearance to project.

- Ad-Hoc Testing: The name itself suggests that this testing is performed on an ad-hoc basis i.e. with no reference to the test case and also without any plan or documentation in place for such type of testing. The objective of this testing is to find the defects and break the application by executing any flow of the application.

  Duplicate Bug Report Detection has been tested randomly by developers until UI bug detected. Changes has been made into system and again tested randomly in it no bug detected.

- Beta Testing : Beta Testing is a formal type of software testing which is carried out by the customer. It is performed in the Real Environment before releasing the product to the market for the actual end users.

  Duplicate Bug Report Detection is sponsored project by Veritas Technologies LLC and External Guide from Veritas has done Beta testing.

- Compatibility Testing : It is a testing type in which it validates how software behaves and runs in a different environment, web servers, hardware, and network environment.

  Duplicate Bug Report Detection has been tested on different operating system like Windows and Linux with different configurations.

- Back End Testing : Whenever an input or data is entered on front-end application, it stores in the database and the testing of such database is known as Database Testing or Backend testing. There are different databases like SQL Server, MySQL, and Oracle etc. There can be issues identified like data loss, deadlock, data corruption etc during this back-end testing and these issues are critical to fixing before the system goes live into the production environment.

Duplicate Bug Report Detection is compatible with SQLite and MongoDB. The backed testing of system has been done along with APIs with no bug observed.

7.2 Test Cases & Test Result

Module	**Input**	**Expected Output**	**Actual Output**	**Result**
Upload File	Upload .json file	Only .csv files are allowed	Files with extension other than .csv will not be accepted.	Pass
Blank Fields	Blank mandatory fields in bug report	Empty fields will be accepted as empty strings	Blank fields will be considered as empty strings	Pass
Special Characters	Special Characters in the input field	User will not be able to insert special symbols.	System does not let user insert special characters in the input	Pass

Table 7.1: Test Cases

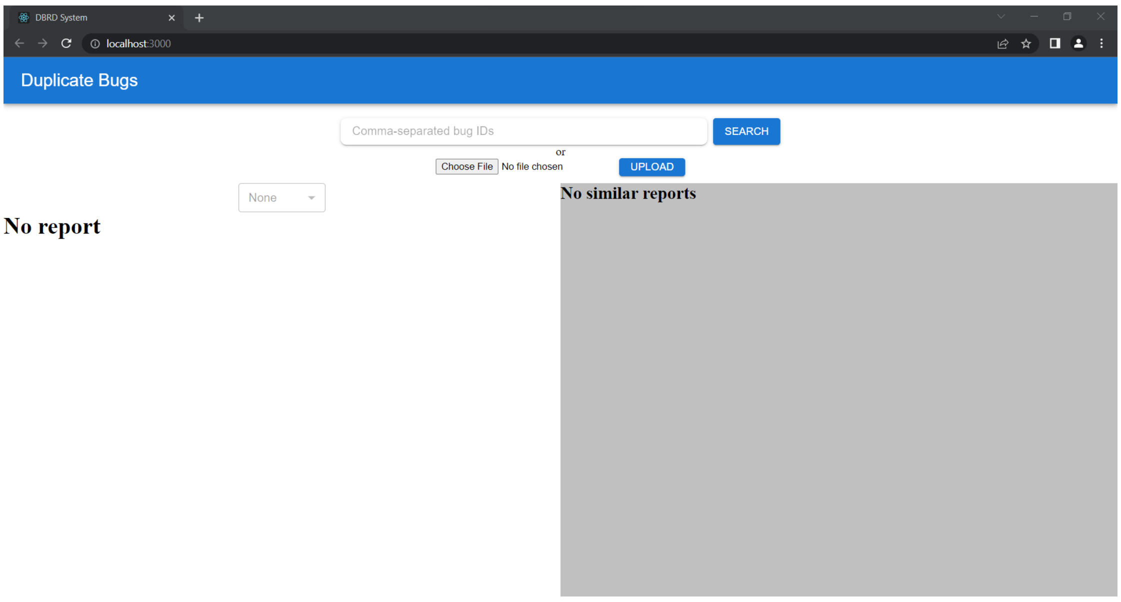
CHAPTER 8

RESULTS

8.1 Outcomes

- We created a corpus of sentences to train our skip-gram model from the Eclipse dataset containing 423559 reports.
- Using gensim library, we trained our model on 120000 bug reports out of 423559 from the dataset.
- This model is used to find the similarity between the bug reports.
- Combining all the three scores (Score-1, Score-2, Score-3) a final score will be displayed with each bug report. Higher score denotes more similarity and vice-versa.
- User will be able to compare reports side by side thus, making the task of finding duplicates easier.

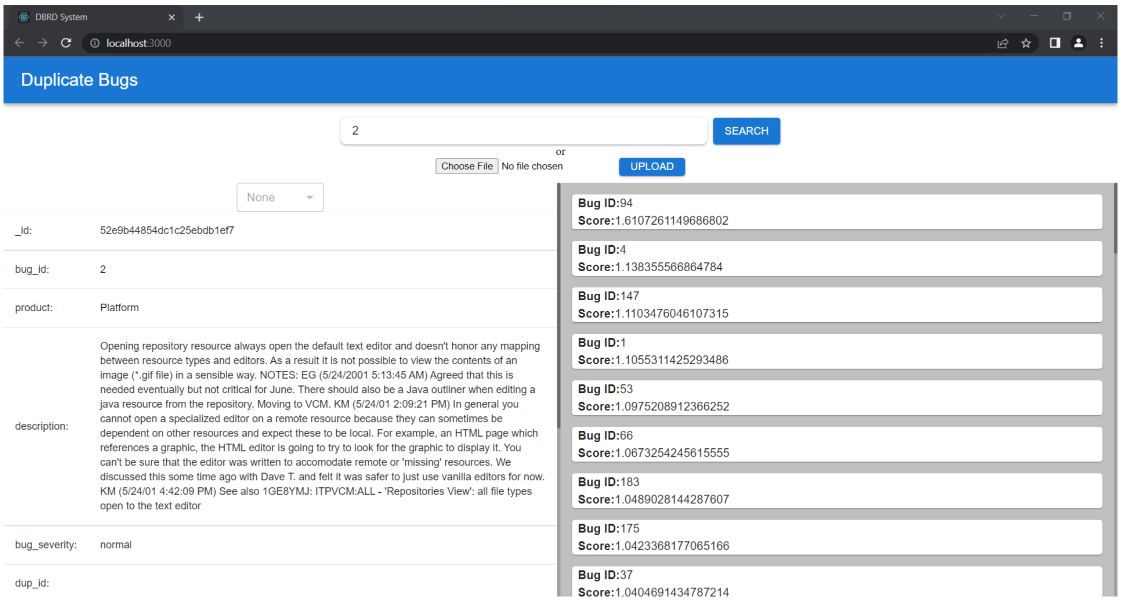
8.2 Screenshots



No report

No similar reports

Figure 8.1: Home Page



2

_id:	52e9b44854dc1c25ebdb1ef7
bug_id:	2
product:	Platform
description:	Opening repository resource always open the default text editor and doesn't honor any mapping between resource types and editors. As a result it is not possible to view the contents of an image (*.gif file) in a sensible way. NOTES: EG (5/24/2001 5:13:45 AM) Agreed that this is needed eventually but not critical for June. There should also be a Java outliner when editing a java resource from the repository. Moving to VCM. KM (5/24/01 2:09:21 PM) In general you cannot open a specialized editor on a remote resource because they can sometimes be dependent on other resources and expect these to be local. For example, an HTML page which references a graphic, the HTML editor is going to try to look for the graphic to display it. You can't be sure that the editor was written to accomodate remote or 'missing' resources. We discussed this some time ago with Dave T. and felt it was safer to just use vanilla editors for now. KM (5/24/01 4:42:09 PM) See also 1GE8YMJ: ITPVCM:ALL - 'Repositories View': all file types open to the text editor
bug_severity:	normal
dup_id:	

Bug ID:94
Score:1.6107261149686802

Bug ID:4
Score:1.138355566864784

Bug ID:147
Score:1.1103476046107315

Bug ID:1
Score:1.1055311425293486

Bug ID:53
Score:1.0975208912366252

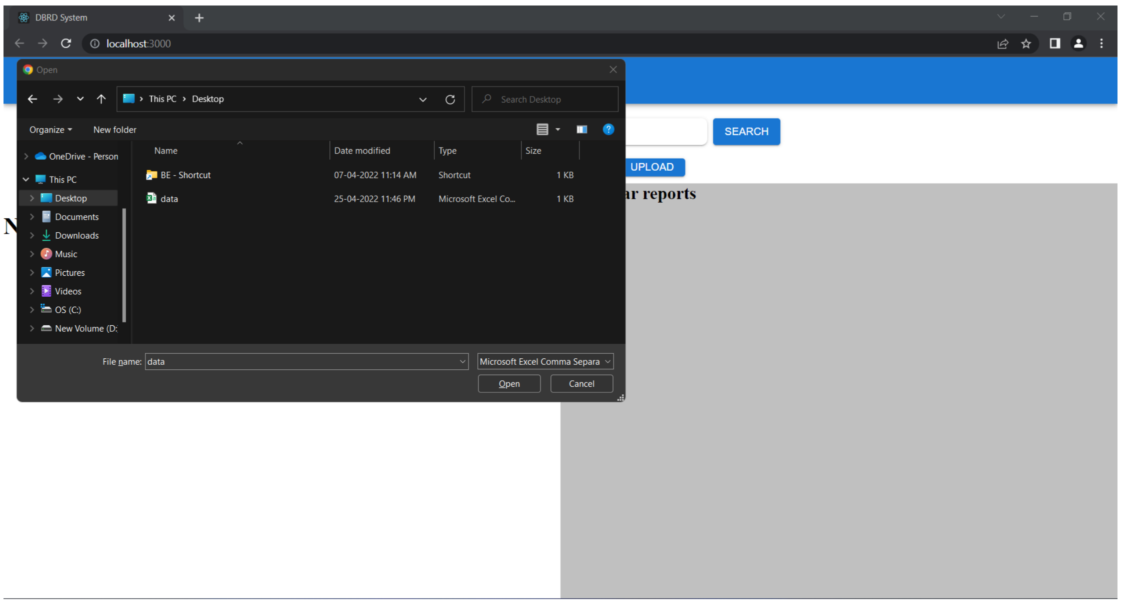
Bug ID:66
Score:1.0673254245615555

Bug ID:183
Score:1.0489028144287607

Bug ID:175
Score:1.0423368177065166

Bug ID:37
Score:1.0404691434787214

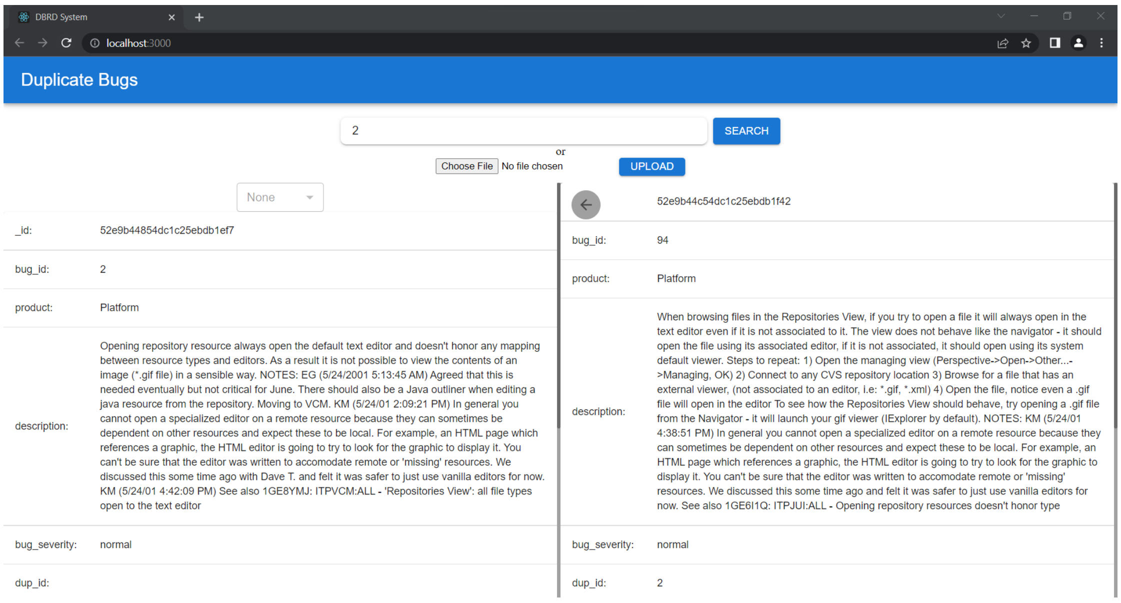
Figure 8.2: Search Bug ID



Name	Date modified	Type	Size
BE - Shortcut	07-04-2022 11:14 AM	Shortcut	1 KB
data	25-04-2022 11:46 PM	Microsoft Excel Co...	1 KB

File name: data
Microsoft Excel Comma Separa

Figure 8.3: File Upload



_id:	52e9b44854dc1c25ebdb1ef7
bug_id:	2
product:	Platform
description:	Opening repository resource always open the default text editor and doesn't honor any mapping between resource types and editors. As a result it is not possible to view the contents of an image (*.gif file) in a sensible way. NOTES: EG (5/24/2001 5:13:45 AM) Agreed that this is needed eventually but not critical for June. There should also be a Java outliner when editing a java resource from the repository. Moving to VCM. KM (5/24/01 2:09:21 PM) In general you cannot open a specialized editor on a remote resource because they can sometimes be dependent on other resources and expect these to be local. For example, an HTML page which references a graphic, the HTML editor is going to try to look for the graphic to display it. You can't be sure that the editor was written to accomodate remote or 'missing' resources. We discussed this some time ago with Dave T. and felt it was safer to just use vanilla editors for now. KM (5/24/01 4:42:09 PM) See also 1GE8YMJ: ITPVCM:ALL - 'Repositories View': all file types open to the text editor
bug_severity:	normal
dup_id:	

	52e9b44c54dc1c25ebdb1f42
bug_id:	94
product:	Platform
description:	When browsing files in the Repositories View, if you try to open a file it will always open in the text editor even if it is not associated to it. The view does not behave like the navigator - it should open the file using its associated editor, if it is not associated, it should open using its system default viewer. Steps to repeat: 1) Open the managing view (Perspective->Open->Other...->Managing, OK) 2) Connect to any CVS repository location 3) Browse for a file that has an external viewer, (not associated to an editor, i.e: *.gif, *.xml) 4) Open the file, notice even a .gif file will open in the editor To see how the Repositories View should behave, try opening a .gif file from the Navigator - it will launch your gif viewer (IExplorer by default). NOTES: KM (5/24/01 4:38:51 PM) In general you cannot open a specialized editor on a remote resource because they can sometimes be dependent on other resources and expect these to be local. For example, an HTML page which references a graphic, the HTML editor is going to try to look for the graphic to display it. You can't be sure that the editor was written to accomodate remote or 'missing' resources. We discussed this some time ago and felt it was safer to just use vanilla editors for now. See also 1GE6I1Q: ITPJUI:ALL - Opening repository resources doesn't honor type
bug_severity:	normal
dup_id:	2

Figure 8.4: Comparing Reports

CHAPTER 9

CONCLUSION & FUTURE WORK

9.1 Conclusion

Based on our implementation of [2], we combine a traditional information retrieval technique with a word embedding technique that takes both bug report titles and descriptions, as well as product and component information into consideration. Using pre-processed bug report documents, TF-IDF vectors and word embeddings two similarity scores were calculated. Furthermore, a third similarity score is also calculated using product and component information. Lastly, we combine the three similarity scores into one final score and based on it's value, similar bug reports are recommended.

Due to high modularity, the system can be integrated into the existing Bug Tracking Tools with the helps of APIs. Since inter module communication is done with the help of APIs, the system database can be chosen according to the needs and purpose.

9.2 Future Scope

In the future, we intend to optimize the parameters of the word embedding technique, and apply optimized search techniques to fetch in required bug reports from the database in order to further enhance the approach. Due to modularity of the system, modules such as Analytics can be added in the future.

APPENDIX A

MATHEMATICAL FORMULAE

1. **TF-IDF**

$$\text{TF(t,d)} = \frac{\text{Occurrence of } t \text{ in d}}{\text{Total number of terms in } d}$$

$$\text{IDF(t)} = \log \frac{\text{Total number of documents}}{\text{Number of documents that contain } t + 1}$$

$$\text{TF-IDF(t,d)} = TF(t,d) \times IDF(t)$$

   where t is a term in document d.

2. **Score 1**

$$Score1 = \frac{v1 \cdot v2}{|v1| * |v2|}$$

   where *v1* and *v2* are TF-IDF vectors for *bug_report_1* and *bug_report_2* respectively.

3. **Score 2**
   Function J of skip gram, is the sum of log probabilities of the surrounding context words under the the condition of a center word.

$$J = \sum_{i=1}^{n} \sum_{w_j \in C_{wi}} \log p(w_j|w_i)$$

   where n denotes length of word sequence. Furthermore, $p(w_j|w_i)$ is the conditional probability calculated using the following *softmax* function:

$$p(w_j \in C_{wi}|w_i) = \frac{exp(v_{w_j}^T v_{w_i})}{\sum_{w \in W} exp(v_w^T v_{w_i})}$$

4. **Score 3**

$$\frac{|set1 \cap set2|}{|set1 \cup set2|}$$

   where *set1* and *set2* refer to the product and component details of bug_report_1 and bug_report_2 respectively.

5. **Final Score**

$$Score = (Score_1 + Score_2) \times Score_3$$

APPENDIX C

PLAGIARISM REPORTS

Ouriginal

Document Information

Analyzed document	BE_Report_Group_ID_54_SEM2_PlagCheck.pdf (D136850785)
Submitted	2022-05-17T11:16:00.0000000
Submitted by	PREETI A. JAIN
Submitter email	pajain@pict.edu
Similarity	14%
Analysis address	pajain.pict@analysis.urkund.com

Sources included in the report

W	URL: https://www.researchgate.net/figure/An-example-of-duplicate-bug-reports_fig1_344774539 Fetched: 2022-05-17T11:16:26.2600000	1
W	URL: https://www.researchgate.net/publication/344774539_Duplicate_Bug_Report_Detection_and_Classification_System_Based_on_Deep_Learning_Technique Fetched: 2021-07-15T08:05:42.0230000	1
W	URL: https://xin-xia.github.io/publication/icpc201.pdf Fetched: 2020-04-23T15:27:25.1070000	2
W	URL: https://taoxie.cs.illinois.edu/publications/icse08-bugreports.pdf Fetched: 2021-11-19T16:40:49.0170000	1
W	URL: https://www.diva-portal.org/smash/get/diva2:1153748/FULLTEXT02.pdf Fetched: 2020-12-15T01:30:03.7530000	2
SA	**Pune Institute of Computer Technology / BE_Project_Report_58.pdf** Document BE_Project_Report_58.pdf (D135405358) Submitted by: loharmalhar@gmail.com Receiver: yahandge.pict@analysis.urkund.com	8
SA	**Pune Institute of Computer Technology / Group_55_Final_Report (1).pdf** Document Group_55_Final_Report (1).pdf (D108523077) Submitted by: rvbidwe@pict.edu Receiver: rvbidwe.pict@analysis.urkund.com	2
SA	**Major Project_Diagnostic_Mukesh.pdf** Document Major Project_Diagnostic_Mukesh.pdf (D50630116)	1
SA	**Pune Institute of Computer Technology / Group_8_Final_Report.pdf** Document Group_8_Final_Report.pdf (D108584456) Submitted by: mschavan@pict.edu Receiver: mschavan.pict@analysis.urkund.com	2
SA	**Fake News Detection using.pdf** Document Fake News Detection using.pdf (D110052970)	1
SA	**Pune Institute of Computer Technology / Group-26-report-onlyContent.pdf** Document Group-26-report-onlyContent.pdf (D136605874)	1

Curiginal

Document Information

Analyzed document	BE_Report_Group_ID_54.pdf (D122856386)
Submitted	2021-12-17T11:52:00.0000000
Submitted by	PREETI A. JAIN
Submitter email	pajain@pict.edu
Similarity	7%
Analysis address	pajain.pict@analysis.urkund.com

Sources included in the report

W	URL: https://www.semanticscholar.org/paper/Duplicate-Bug-Report-Detection-and-Classification-Kukkar-Mohana/2284dcd2a3e0b2cf0e90de5c9974d57ec4173328 Fetched: 2021-12-17T12:22:51.0630000	1
W	URL: https://www.researchgate.net/publication/344774539_Duplicate_Bug_Report_Detection_and_Classification_System_Based_on_Deep_Learning_Technique Fetched: 2021-07-15T08:05:42.0230000	1
W	URL: https://xin-xia.github.io/publication/icpc201.pdf Fetched: 2020-04-23T15:27:25.1070000	2
W	URL: https://lupinepublishers.com/computer-science-journal/fulltext/duplicate-detection-models-for-bug-reports-of-software-triage-systems-a-survey.ID.000123.php Fetched: 2021-11-19T16:40:43.6130000	2
W	URL: https://ieeexplore.ieee.org/document/9235309 Fetched: 2021-05-10T12:26:05.4430000	2
W	URL: https://www.diva-portal.org/smash/get/diva2:1153748/FULLTEXT02.pdf Fetched: 2020-12-15T01:30:03.7530000	2
SA	**Pune Institute of Computer Technology / Group_55_Final_Report (1).pdf** Document Group_55_Final_Report (1).pdf (D108523077) Submitted by: rvbidwe@pict.edu Receiver: rvbidwe.pict@analysis.urkund.com	1
W	URL: https://ojcchar.github.io/files/12-saner19.pdf Fetched: 2021-12-12T15:23:14.5770000	1
SA	**Paper_with_Ruby.pdf** Document Paper_with_Ruby.pdf (D119197917)	1
W	URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.224.4255 Fetched: 2021-12-17T12:23:34.8070000	1

REFERENCES

[1] Li. Kang. Automated duplicate bug reports detection-an experiment at axis communication ab., 2017.

[2] Xinli Yang, David Lo, Xin Xia, Lingfeng Bao, and Jianling Sun. Combining word embedding with information retrieval to recommend similar bug reports, 2016.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[4] Chengnian Sun, David Lo, Siau-Cheng Khoo, and Jing Jiang. Towards more accurate retrieval of duplicate bug reports, 2011.

[5] Ashima Kukkar, Rajni Mohana, Yugal Kumar, Anand Nayyar, Muhammad Bilal, and Kyung-Sup Kwak. Duplicate bug report detection and classification system based on deep learning technique, 2020.

Log Sheet

Team Id: 54
Project Title: **Duplicate Bug report Detection**
Team Members: Omkar Amilkanthwar (41403)
Aniruddha Deshmukh (41405)
Atharva Satpute (41409)
Pranav Deshmukh (41417)

Internal Guide
Prof. Preeti Jain

External Guide
Mr. Ganesh Warwatkar

Sr. No	Day and Time	Guide	Discussion	Remarks	Sign
1.	03/09/2021	Prof. Preeti Jain	Project topic discussion		
2.	04/09/2021	Prof. Preeti Jain	Project topic discussion		
3.	18/09/2021	Prof. Preeti Jain	Project topic discussion		
4.	04/10/2021	Mr. Ganesh Warwatkar	Overview of the project title		
5.	12/10/2021	Mr. Ganesh Warwatkar	Discussion on Literature Survey		
6.	25/10/2021	Mr. Ganesh Warwatkar	Discussion on tech-stack & flow of the project		
7.	02/11/2021	Mr. Ganesh Warwatkar	Follow-up meet		
8.	16/11/2021	Mr. Ganesh Warwatkar	Discussion regarding implementation, optimization and scope improvisation		

9.	09/12/2021	Prof. Preeti Jain	Discussed changes in the report		
10.	14/12/2021	Prof. Preeti Jain	Review with the experts		
11.	16/12/2021	Prof. Preeti Jain	Changes in synopsis		
12.	23/12/2021	Prof. Preeti Jain	Project Report Submission		