AIJR Preprints
Section: Working Paper
Article Id: 100, Version: 1, 2020
URL: https://preprints.aijr.org/index.php/ap/preprint/view/100
{Click on above link to see the latest available version of this article}

NOT PEER-REVIEWED

Version 1: Received: 28 May 2020 / Approved: 03 June 2020 / Online: 04 June 2020

Validation Optimisation using Machine Learning Techniques

R Akshay Dharmapuri

Electronics and Communication Engineering, Nirma University

First Author's Full Name: R Akshay Dharmapuri
Highest Qualification: Mtech
Department: ECE
Post/Rank (If a student, provide course name and course year):Embedded Systems,2018-20
Affiliation: Nirma University,Sarkhej-Gandhinagar Highway,Ahmedabad,Gujarat
email id:**********

ABSTRACT

Integration and validation is the most vital part before releasing products to customers in Intel. The validation team qualifies the release based on multiple stages of validation on hardware and software stack. Bugs are raised after execution of test cases on each platform and so similar bugs arise which are filed by the user. There is a immediate concern on this and hence, many issues are closed as duplicates.The main objective is to find these similar bugs for each bug filed and thereby,debug efforts can be reused.Similar bugs are found by term based search using ElasticSearch ,a text search engine and neural network based search where context is considered.Using elasticsearch,scoring algorithms based on driver versions and platform hierarchy are applied to rank the similar bugs. LSTM neural networks are also incorporated to predict duplicate bugs by considering context of the sentence and thereby, increasing accuracy.

Keywords: NLP, Machine Learning, ElasticSearch

1 Introduction

Machine Learning relates with the study, design, and development of models and algorithms that give computers the capability to learn from data.It is the most effective method in the field of data analytics in order to predict something by devising some models and algorithms.These analytical models allow researchers,engineers and data scientists to produce reliable and valid results and decisions.Natural Language Processing (NLP) is a field that makes computers build their understanding and interpreting skills on human

How to Cite:

R Akshay Dharmapuri "Validation Optimisation using Machine Learning Techniques". *AIJR Preprints*, 100, version 1, 2020.
https://preprints.aijr.org/index.php/ap/preprint/view/100

language.Translation of languages,Summarization of paragraphs,Named Entity Recognition,Recognition of Speech,Topic Modelling,etc are the fields where NLP is applied.[1, 2]

The validation team qualifies the release based on multiple stages of validation on the hardware and software stack.Every week ingredient teams come up with software changes in components that will be integrated into software stack.Validation of software stack is performed by executing test cases for each domain.Test cases are prioritized,selected and executed to qualify the software and firmware stack. All test cases will not get impacted every release and hence, there is scope of optimising validation efforts. Different teams in platform validation work together to validate and identify bugs.These bugs can arise due to mismatch of product specifications.[1]

To identify possible similar bugs while filing a new bug which otherwise can lead to wastage of efforts,resources and time.The detailed insights of the bug can be found from existing fixed similar bugs thereby, reusing debug efforts.This calls for a mechanism to be in place to check bug duplication before filing a new bug.

2 ElasticSearch Overview

Elasticsearch is the distributed search and analytics engine which provides real-time search and analytics for all types of data. It is open source, built in Java and is platform independent.It uses a data structure in which it lists every unique word that appears in any document and identifies all of the documents each word occurs in.An ElasticSearch index is a collection of documents that are related to each other.Each document correlates a set of keys with their corresponding values such as strings ,numbers, boolean,etc.

This engine works on Term Frequency (TF), Inverse Document Frequency (IDF), Field length normalization (norm), Coordination (coord) factor and many others.

The main purpose of doing a search is to find out similar documents matching the query. Since Term Frequency considers all terms equally important, and so it cannot be only used to calculate the weight of a term in the document. There are some terms like stop words, such as “is”, “of”, and “that”, appear a lot of times but they are least valued. So, the frequent terms should be weighed down and the rare ones should be given more weight and so, more weightage should be given to it. [4]

ElasticSearch is based on term-based search where context is not considered. Word2vec is a model to produce word embedding for better word representation,as neural networks take a sequence of vectors as input and produces a sequence of vectors.So, LSTM,a neural network based search is applied to find similar bugs.It uses word embeddings in the neighbourhood of each word and tries to predict the next word.It gives better accuracy than RNN as it eliminates long term dependencies by preserving the context for larger distance.[3]

3 Methodology and Scoring Algorithms

Duplicate bugs are found using two methods described in this section. Firstly, similar bugs are found using ElasticSearch and scoring algorithms in order to get an appropriate score. In the second method, bugs are found using LSTM networks which takes context also in consideration.

The methodology and implementation of the work done using ElasticSearch is described below.

- An ElasticSearch index of defect details is created.
- The bug id, title and other details are entered by the user and processing of title is done which includes removal of stopwords and punctuations.
- The entire title is broken into tokens for POS (Part of Speech) tagging to get the nouns, verbs and its derivatives.
- The tokenized words are tagged into entities based on dictionary created which consists of list of unique words derived from entire data of defects. These words are tagged into entities and domains using NER (Named Entity Recognition).This can be understood in the table given below :[4]

Table 3.1: Assignment of terms and domains identified

Terms	Entity	Domain1	Domain2	Domain3	Domain4
Term1	Entity1	X	Y	Z	W
Term2	Entity2	X	Y		
Term3	Entity1	Z			
Term4	Entity3	Z	W	X	

- Entity based words, nouns and verbs identified from the title are passed to ElasticSearch query and searched in the index that is created. Higher priority is given to those words that have their entities defined.
- Top ten similar bugs for each bug are listed with their details.
- ElasticSearch uses TF-IDF scoring formula and so, finds the similarity between words from the titles and displays a score for each of the similar bug.
- Normalization of scores produced by elastic is done to bring all the scores in the same range.50% of normalized elastic score is considered.
- Platform distance score is calculated through an algorithm based on the predecessor matrix built. This matrix specifies the distance between platforms. The lesser the distance,higher the score.20% weightage is considered to calculate the platform distance score.
- The versions are identified from the bug filed by the user and similar bugs. Both the versions are compared and scoring is done based on how older or newer they are.30% weightage is considered to calculate the driver version score.
- The total score is calculated by adding the platform score, driver version score and normalized elastic score. The top four bugs are displayed based on the total scores obtained.[3]

The flow is described in the figure below:

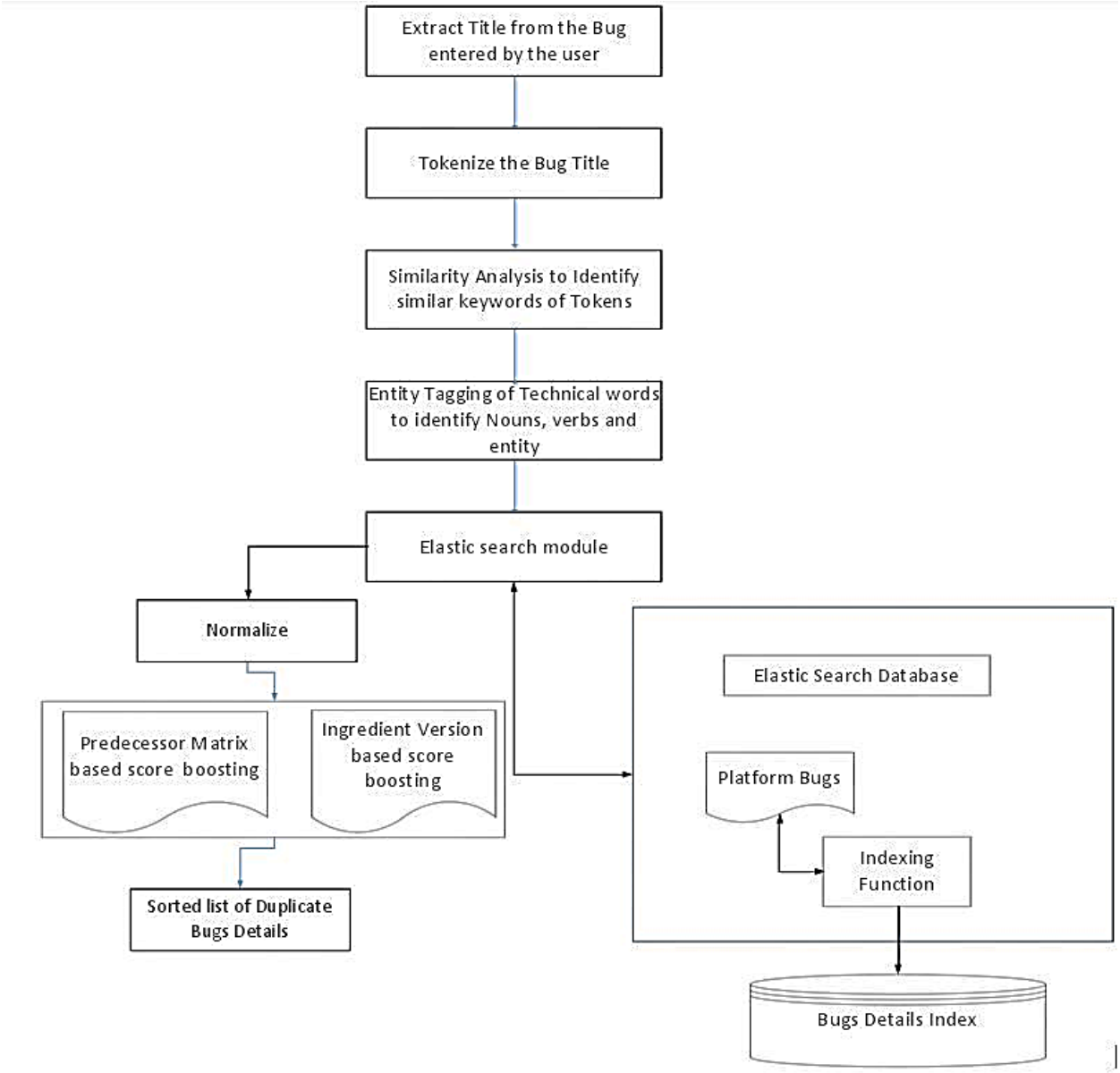

Figure 3.1 Working Flow using ElasticSearch

3.1 Platform Scoring Logic

Platforms are versions of next generation of devices. Normalization of score is done in order to bring all the scores in the same range.20 % platform weightage is given after similar bugs for a particular bug are found out. A predecessor matrix is built which specifies the distance between the platforms of bug filed and the similar bug. The algorithm results a platform distance score based on the predecessor distance specified in the matrix. For example, if the distance between the platform of duplicate bug and platform of the bug filed by the user is 0, highest weightage is given. If the distance between the platform of duplicate bug

and platform of the bug filed by the user is 3, the duplicate bug belongs to oldest platform. So, least weightage is given [3]. The predecessor matrix used for this algorithm is shown below:

Table 3.2 Platform Scoring Predecessor Matrix

Platform	Platform1	Platform2	Platform3	Platform4	Platform5	Platform6
Platform1	0					
Platform2	1	0				
Platform3	1	1	0			
Platform4	1	1		0		
Platform5	2	1			0	
Platform6	3	2			1	0

The calculation for this logic is as follows:

- Normalization of scores is done to bring all the scores in the same range, that is, from 1 to 10.
- Normalization is given by the following formula:

$$Norm\ Score = 9 * \frac{current\ defect\ score - minimum\ elastic\ score}{maximum\ elastic\ score - minimum\ elastic\ score} + 1$$

- 50 % of normalized elastic score is considered, as term search is equally important.
- 4 levels of platform are considered, that is, N, N-1, N-2, N-3
- The platform distance score is calculated as follows:

$$x = 20\% * range$$

$$x = 20\ \% * 10$$

$$x = 2$$

$$increment = \frac{x}{4} = 0.5$$

The range of score is assumed to be 10, and platform weightage is taken as 20%. The value of x is 2 and is divided by number of levels ,that is 4, thereby resulting 0.5.

- If the duplicate bug and bug filed by the user belong to the same platform, the platform distance calculated from the matrix is 0. Highest weightage is given and so, the platform distance score is 2.

$$platform\ distance\ score = 2$$

- If the bug filed by the user belongs to N_{th} platform and respective duplicate bug belongs to $N\text{-}1_{th}$ platform, the platform distance calculated from the matrix is 1. Lesser weightage is given as compared to earlier and so, the platform distance score is 1.5.

$$platform\ distance\ score = 2 - 0.5 = 1.5$$

- If the bug filed by the user belongs to N_{th} platform and respective duplicate bug belongs to $N\text{-}2_{th}$ platform, the platform distance calculated from the matrix is 2. Lesser weightage is given as compared to earlier and so, the platform distance score is 1.0.

$$platform\ distance\ score = 2 - (0.5 * 2) = 1.0$$

- If the bug filed by the user belongs to N_{th} platform and respective duplicate bug belongs to $N\text{-}3_{th}$ platform, the platform distance calculated from the matrix is 3. Least weightage is given as duplicate bug belongs to an oldest platform.So,the platform distance score is 0.5.[3]

$$platform\ distance\ score = 2 - (0.5 * 3) = 0.5$$

3.2 Version Scoring Logic

A source code management system or a repository is present where the versions of each driver and its details are updated and released.Higher priority is given to the most newest version, and hence bugs having older versions are penalized.The versions are identified from the bug filed by the user and similar bugs.Both the versions are compared and scoring is done based on how older or newer they are.30 % weightage is given for the versions scoring algorithm.Latest drivers of respective platforms are taken from the repository depending upon the date when they are released.The flow for calculation is explained below :[3]

- The platform, workweek and other details from the duplicate bugs are extracted and searched in the repository.
- The drivers and their versions are extracted from the repository and stored in a dictionary1.
- The domains are identified from the bug details provided by the user through NER (Named Entity Recognition).
- Domains identified are mapped to driver details through a domain driver mapping table. After mapping, drivers and their versions are extracted and stored in dictionary 2.
- The versions present in these two dictionaries are compared through an algorithm which will be discussed below.[3]0

The respective digits in both the versions are compared. The scoring algorithm for comparing versions is discussed below:

- Four levels of version comparison are considered as there are four floating-point digits.
- The amount of score to be incremented is shown below:

$$x = 30\% * range$$
$$x = 30\% * 10$$
$$x = 3$$
$$increment = \frac{x}{4} = 0.75$$

The range of elastic scores is assumed to be 10, and driver version weightage is taken as 30%. The value of x is 3 and is divided by number of levels, that is, 4, thereby, resulting 0.75.

- If rightmost(last) digit changes between 2 driver versions, highest weightage is given to the driver version score which is 3. This is because the driver being compared is the most recent version of the same driver.

$$driver\ version\ score = 3$$

- If the third digit changes between 2 driver versions, lesser weightage is given to the driver version score which is 2.25. This is because the driver being compared is a slightly older version of the same driver.

$$driver\ version\ score = x - increment = 3 - 0.75 = 2.25$$

- If the second digit changes between 2 driver versions, lesser weightage is given to the driver version score which is 1.50. This is because the driver being compared is an older version of the same driver.

$$driver\ version\ score = x - (increment * 2) = 3 - (0.75 * 2) = 1.50$$

- If the leftmost(first) digit changes between 2 driver versions, least weightage is given to the driver version score which is 0.75. This is because the driver being compared is an oldest version of the same driver.

$$driver\ version\ score = x - (increment * 3) = 3 - (0.75 * 3) = 0.75$$

Finally, the total score is calculated by adding the platform score, driver version score and normalized elastic score.[4]

$$Total\ Score = 20\%\ platform\ distance\ score + 30\%\ driver\ version\ score + 50\%\ normalized\ score$$

4 Results

The results produced after applying scoring algorithms are shown in the below figure. Details of duplicate bugs for a specific bug and their scores are also shown.The Bug Title represents the bug passed by the user.The Platform and Driver field on the left side represents the details of the bug passed by the user.The Terms and Domains Identified field represents the terms ,entities and domains for each term.The Normalized Score field represents the score produced by elastic and squeezed in the range 1 to 10.The Platform score represents the score produced by the platform scoring algorithm,which is calculated based on the predecessor matrix of platforms.The Driver version score represents the score produced by version scoring algorithm,which compares each digit in the version.The Total Score represents the score produced by adding the Platform score,Normalized Elastic score and the Driver version score.

Table 4.1 Results through driver and platform scoring algorithms using ElasicSearch

Bug Title	Platfo rm	Driver	Terms & Domains Identified	Duplicate Bug	Platform	Platform Score	Version Score	Normalized Score	Total Score
Title1	N	Driver1	Term1:Entity1-Domain1,Domain2; Term2:Entity1-Domain1,Domain2, Domain3	Title1	N	3.453731	3	4	10.45373
				Title2	N	3.453731	3	3.263968	9.7177
				Title3	N-1	0	3	3.739668	6.739668
Title2	N	Driver2	Term1:Entity1-Domain1,Domain2 ,Domain3 ,Domain4; Term2:Entity2-Domain1; Term3 :Entity1-Domain1,Domain2, Domain3,Domain5, Domain6	Title1	N	4.1351	3	5	12.1351
				Title2	N	4.1351	3	0.6164	7.751499
				Title3	N-2	2.06755	3	1.776423	6.843973

5 Conclusion

The accuracy of similar bugs found using ElasticSearch is around 60%. The duplicate bugs found related to closest platform and closest version are given a higher score than the other bugs. The total scores found are sorted in descending order to get the top similar bugs. The limitation of this algorithm is the context of the sentence is not considered. LSTM based neural network, which is based on context is being studied in order to improve the accuracy. The main problem that was faced using ElasticSearch was that the dictionary is manually made containing the terms,entities and respective domains and hence, new words need to be manually added.Instead, neural networks use Word2Vec models which bring all similar words with similar contexts together from a vocabulary of words.The other problem was that various stop words were removed before searching for defects.Some examples of these words would be 'after','before',etc. were removed, and hence context of the sentence got disappeared.So,neural networks like LSTM can be used to overcome this problem which has the tendency to learn long-term dependencies.

6 Competing Interests

The author declared that no Conflict of Interest exist.

References

[1] Giorgio Maria Di Nunzio and Alexandro, “A Study on Query Expansion with MeSH Terms and Elasticsearch”. http://ceur-ws.org/Vol-2125/paper_200.pdf

[2] Jasper Sneek, Hugo Larochelle and Ryan P.Adams ,”Practical Bayesian Optimization of Machine Learning Algorithms”,Advances in Neural Information Processing Systems 2 5 (NIPS 2012).

[3] Nilanjana Dev Nath, Shree Kant Jha,Janki Meena M ,”A Survey Paper on Elastic Search Similarity Algorithm”,Asian Journal of Pharmaceutical And Clinical Research. April 2017. https://doi.org/10.22159/ajpcr.2017.v10s1.19757

[4] Darshita Kalyani, Dr. Devarshi Mehta.,”Paper on Searching and Indexing Using ElasticSearch”, International Journal of Engineering and Computer Science, Volume 6 Issue 6 June 2017.