SOFTWARE TESTING

MTE PROJECT

BUG TRIAGING

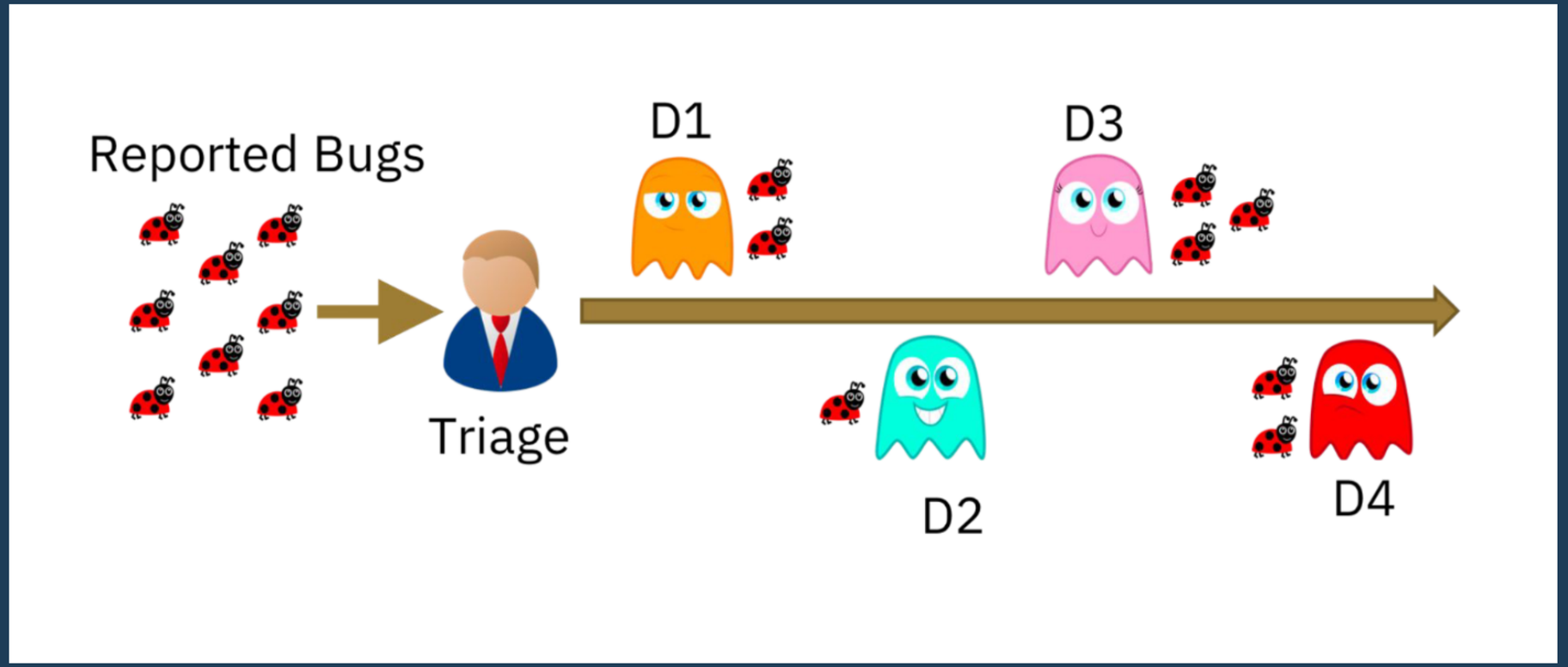
DEEPANSHU SINGHANIYA (2K19/SE/033)
DEVANSHI SINGH (2K19/SE/035)

INTRODUCTION

What is bug triaging?

Triaging is the process of reviewing bugs to ensure they are valid, reproducible, and have accurate information that allows them to be resolved and tested.
Identifying an appropriate developer who could potentially fix the bug is the primary task of a bug triaging process.

Bug Triaging process



Our Model

Technology around the globe

A novel bug report representation approach is proposed using DBRNN-A: Deep Bidirectional Recurrent Neural Network with Attention mechanism and with Long Short-Term Memory units (LSTM)

The proposed deep algorithm is capable of remembering the context over a long sequence of words.

We have used an attention based deep bidirectional recurrent neural network (DBRNN-A) model that learns a syntactic and semantic feature from long word sequences in an unsupervised manner.

In large scale systems, with a huge number of incoming bugs, manually analyzing and triaging a bug report is a laborious process. Manual bug triaging is usually performed using the bug report content, primarily consisting of the summary and description

Dataset

A total of 383,104 bugs where collected with the bug title, description, the bug owner, and the reported time. The developer in the “owner" field is considered as the ground truth triage class for the given bug .
Bugs with status as Verified or Fixed, and type as bug, and has a valid ground truth bug owner are used for training and testing the classifier while rest of the bugs are used for learning a bug representation

Property	Chromium	Core
Total bugs	383,104	314,388
Bugs for learning feature	263,936	186,173
Bugs for classifier	118,643	128,215
Vocabulary size $\|V\|$	71,575	122,578

Issue 35980: window size and position set to 0 right after window.open call

Reported by chr...@netscript.com on Wed, Feb 17, 2010, 12:41 PM GMT+5:30

Chrome Version : 5.0.307.7 (Official Build 38400) beta (Mac)
URLs (if applicable) :
Other browsers tested:
Add OK or FAIL after other browsers where you have tested this issue:
Safari 4: ok
Firefox 3.x: ok
IE 7:
IE 8:

What steps will reproduce the problem?
1. w=window.open(...)
2. alert(w.innerHeight)
3.

What is the expected result?
Some valid values for innerHeight, innerWidth, screenX and screenY.

A sample bug report

Json format-Dataset

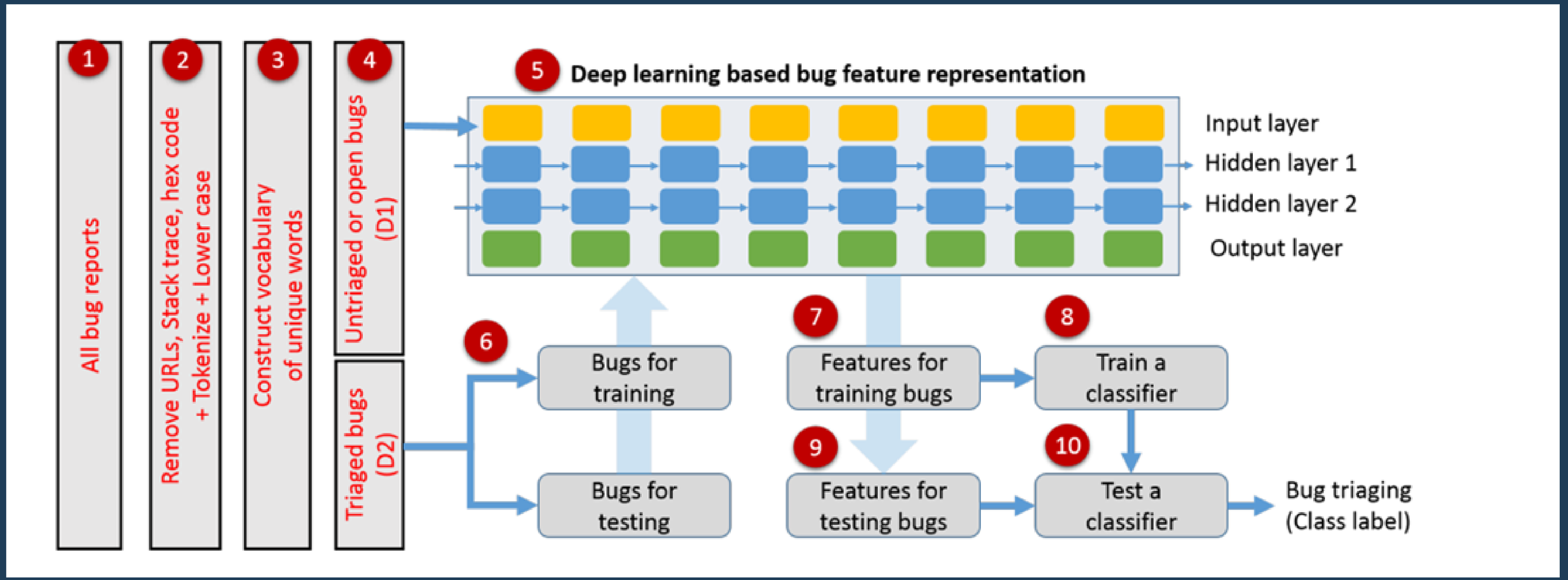

```
{
    "id" : 204503,
    "issue_id" : 35980,
    "issue_title" : "window size and position set to 0 right after window.o
    "reported_time" : "2010-02-17 07:11:21",
    "owner" : "gavinp@chromium.org",
    "description" : "\nChrome Version       : 5.0.307.7 (Official Build 384
    "status" : "Available",
    "type" : "Bug"
},
```

APPROACH

The problem of automated bug triaging of software bug reports is formulated as a supervised classification approach with the input data being the bug summary (title) and the bug description.

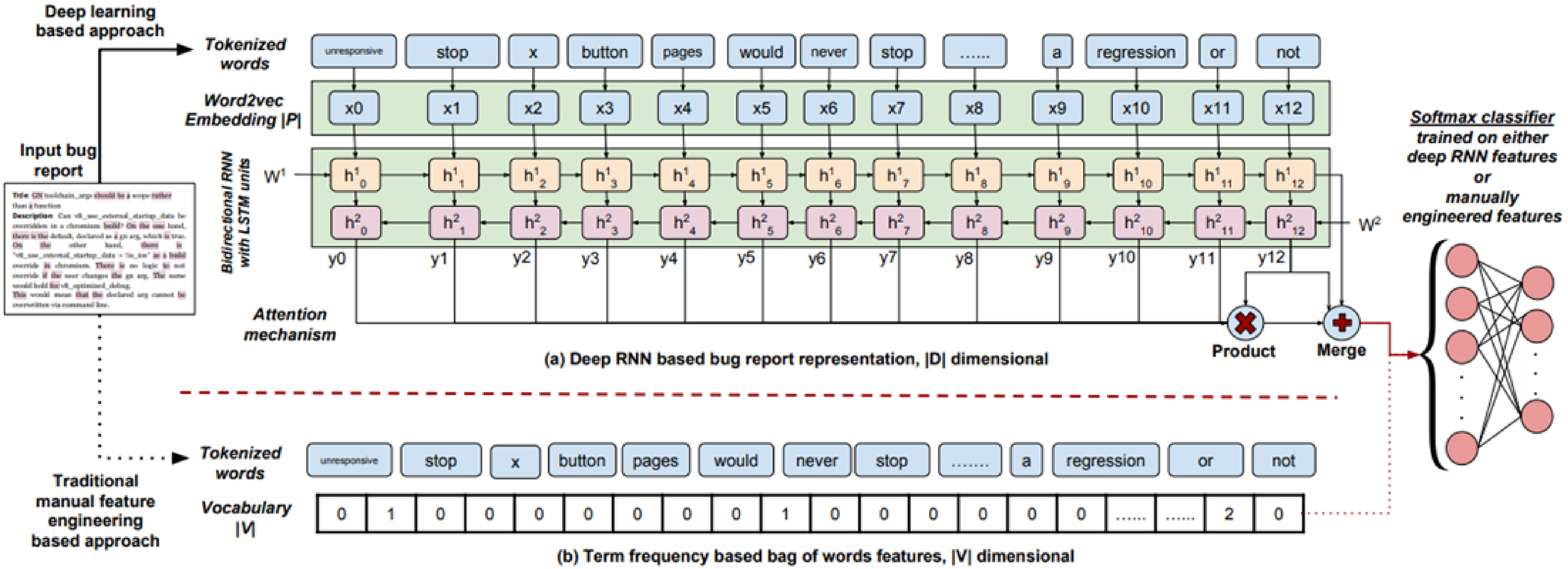
- A bug corpus having title, description, reported time, status, and owner is extracted from an open-source bug tracking system,
- Handling the URLs, stack trace, hex code, and the code snippets in the unstructured description requires specialized training of the deep learning model, and hence in this research work, those contents are removed in the preprocessing stage,
- A set of unique words that occurred for at least k-times in the corpus is extracted as the vocabulary,
- The triaged bugs (D2) are used for classifier training and test, while all the untriaged/ open bugs (D1) are used to learn a deep learning model,

- A deep bidirectional recurrent neural network with attention mechanism technique learns a bug representation considering the combined bug title and description as a sequence of word tokens,
- The triaged bugs (D2) are split into train and test data with a 10 fold cross validation to remove training bias,
- Feature representation for the training bug reports are extracted using the learnt DB-RNN algorithm,
- A supervised classifier is trained for performing developer assignment as a part of bug triaging process,
- Feature representation of the testing bugs are then extracted using the learnt deep learning algorithm,
- Using the extracted features and the learnt classifier, a probability score for every potential developer is predicted and the accuracy is computed in the test set.


1
All bug reports
2
Remove URLs, Stack trace, hex code + Tokenize + Lower case
3
Construct vocabulary of unique words
4
Untriaged or open bugs (D1)
Triaged bugs (D2)
5
Deep learning based bug feature representation
Input layer
Hidden layer 1
Hidden layer 2
Output layer
6
Bugs for training
Bugs for testing
7
Features for training bugs
Features for testing bugs
8
Train a classifier
9
10
Test a classifier
Bug triaging (Class label)

Deep Bidirectional Recurrent Neural Network with Attention (DBRNN-A)

DBRNN-A can learn sentence representation preserving the order and syntax of words, as well as, retaining the semantic relationship. Long short-term memory (LSTM) cells are used in the hidden layer which have a memory unit that can remember longer word sequences and also can solve the vanishing gradient problem


Deep learning based approach
Tokenized words
Word2vec Embedding |P|
Input bug report
Bidirectional RNN with LSTM units
W1
W2
Attention mechanism
unresponsive
stop
x
button
pages
would
never
stop
.....
a
regression
or
not
x0
x1
x2
x3
x4
x5
x6
x7
x8
x9
x10
x11
x12
y0
y1
y2
y3
y4
y5
y6
y7
y8
y9
y10
y11
y12
Product
Merge
Softmax classifier trained on either deep RNN features or manually engineered features
(a) Deep RNN based bug report representation, |D| dimensional
Traditional manual feature engineering based approach
Tokenized words
Vocabulary |V|
0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 .... 2 0
(b) Term frequency based bag of words features, |V| dimensional

Classifying (Triaging) a Bug Report

We use a Softmax classifier, a popular choice of classifier along with deep learning

- Softmax classifier is a generalization of logistic regression for multiclass classification, taking the features and providing a vector of scores with length equal to the number of the classes.
- A softmax classifier normalizes these score values and provides an interpretable probability value of the i-th bug report belonging to the class

Data Pre-processing

For every bug report, the title and description text content of the bug are combined.

Preprocessing of the unstructured textual content involves removing URLs, hex code, and stack trace information, and converting all text to lower case letters.

Tokenization of words is performed using Stanford's NLTK package6. A vocabulary of all words is constructed using the entire corpus. To remove rarely occurring words and reduce the vocabulary size, usually the top-F frequent words are considered or only those words occurring with a minimum frequency are considered.

TRAINING DATA FOR DEEP LEARNING

Only the untriaged bugs (explained in the data extraction subsection) is used for training the deep learning model.

TRAINING DATA FOR CLASSIFICATION

For training and testing the supervised classifier, a 10-fold cross validation model is followed. All the fixed bug reports are arranged in chronological order and split into 11 sets. Starting from the second fold, every fold is used as a test set, with the cumulation of previous folds for training.

Performance Measures for Google Chromium

Threshhold	CV#1	CV#2	CV#3	CV#4	CV#5	CV#6	CV#7	CV#8	CV#9	CV#10	Average
Min train samples per class = 0	34.9	36.0	39.6	35.1	36.2	39.5	39.2	39.1	39.4	39.7	37.9 ± 1.9
Min train samples per class = 5	32.2	33.2	37.0	36.4	37.1	37.2	38.3	39.0	39.1	38.2	36.8 ± 2.2
Min train samples per class = 10	36.2	37.1	40.45	42.2	41.2	41.3	44.0	44.3	45.3	46.0	41.8 ± 3.1
Min train samples per class = 20	36.7	37.4	41.4	42.5	41.8	42.6	44.7	46.8	46.5	47.0	42.7 ± 3.5

Performance Metrices for Mozila Firefox

Threshhold	CV#1	CV#2	CV#3	CV#4	CV#5	CV#6	CV#7	CV#8	CV#9	CV#10	Average
Min train samples per class = 0	33.6	34.2	34.7	36.1	38.0	37.3	38.9	36.3	37.4	38.1	36.5 ± 1.7
Min train samples per class = 5	27.6	34.9	37.9	38.7	40.1	42.3	45.2	44.9	45.0	44.5	40.1 ± 5.3
Min train samples per class = 10	35.1	36.4	40.5	42.5	45.4	47.4	48.9	49.1	51.1	51.4	44.8 ± 5.6
Min train samples per class = 20	36.8	37.4	39.5	43.9	45.0	47.1	50.5	53.3	54.3	55.8	46.6 ± 6.4