TARUN ARORA

AI-Driven Techniques for Noise Filtration in Software Validation Logs

PACIFIC NW SOFTWARE QUALITY CONFERENCE

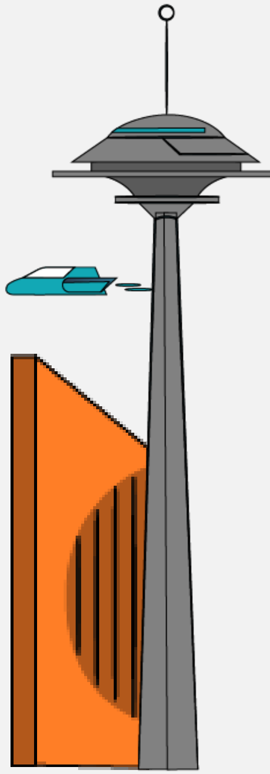
THE FUTURE IS NOW

PNSQC.ORG

OCTOBER 14-16 2024

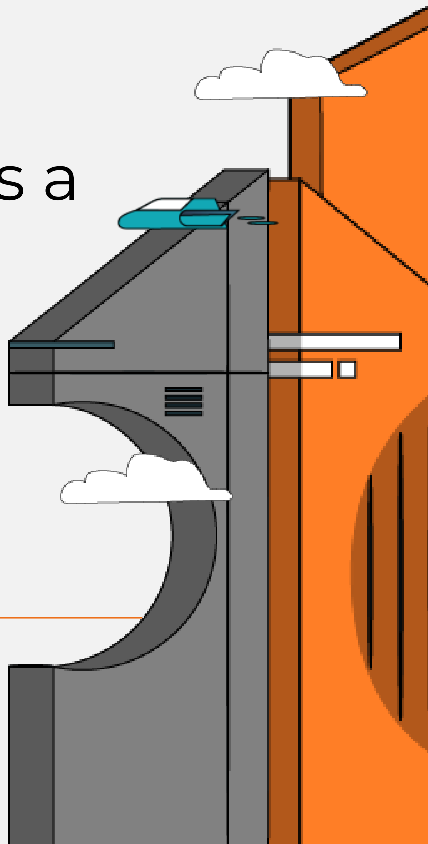
Outline

- Topic Introduction
- Context
- Challenges
- Problem Statement
- Solution
- Machine Learning – End to End Deployment Flow
- Natural Language Processing – Brief Implementation Routine
- Key Learnings and Take Aways



Topic Introduction

- Our paper explores the integration of Software Testing Log Files Diagnostics, Natural Language Processing (NLP) and Ensemble Machine Learning techniques.
- By combining advanced NLP for data interpretation with ensemble classification, we've dramatically improved error and query classification accuracy.
- This method not only reduces manual intervention but also sets a new standard for automated, precise system diagnostics



Context

Bugs Error Triage Flow

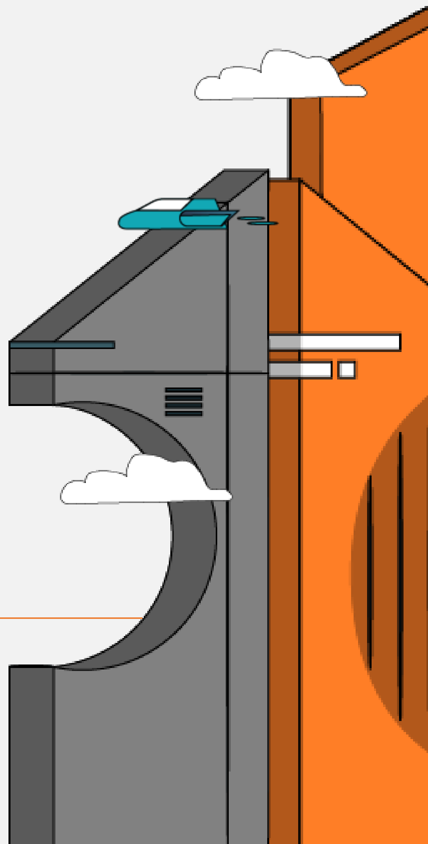
- Software development projects are validated for thousands of use cases by various system integration and validation techniques.
- Each of the tests executed as part of these validation cycles generates validation logs which vary in types and sizes.
- Test types could be basic acceptance test, sanity test, stability, functional, Tape In, pre-integration and many others.

Software Release “Builds” are fed into “Test Environment”

Testing generated logs for each execution

Test or Triage Engineer analyze these logs to identify error

Each part of logs is categorized as noise(info) vs error and error are further segregated for triage to the team



Challenges

- Tests generate various log files, including: .log, .csv etc.
- Each log file has a varied format
  - Even .log file ext. for different tests have different format
- Test cycles generate hundreds of logs weekly
- This scenario exemplifies the Big Data 3 V's:
  - Volume
  - Velocity
  - Variety

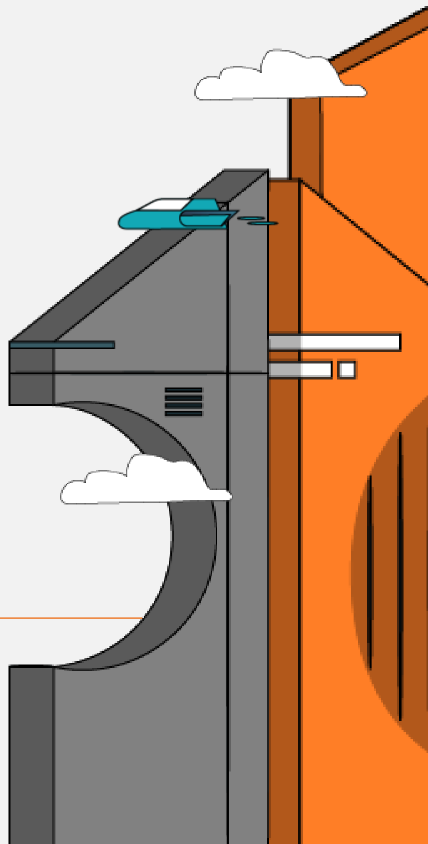
Volume: Amount of data from myriad executions

Velocity: Speed at which thousands of logs generated in a day

Variety: Types of data and log files including varied format, structure, semi structured and unstructured

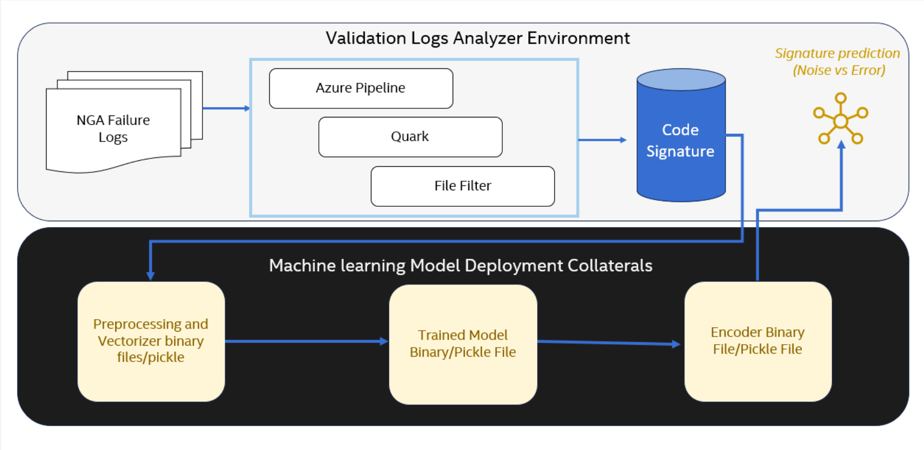
Problem Statement

- Validation logs face the 3V challenge (volume, velocity, variety), making error identification labor-intensive and error-prone,
- With current methods (manual review and Regex) being time-consuming, maintenance-heavy, and prone to human oversight, leading to 1-2 hours of triage per log.

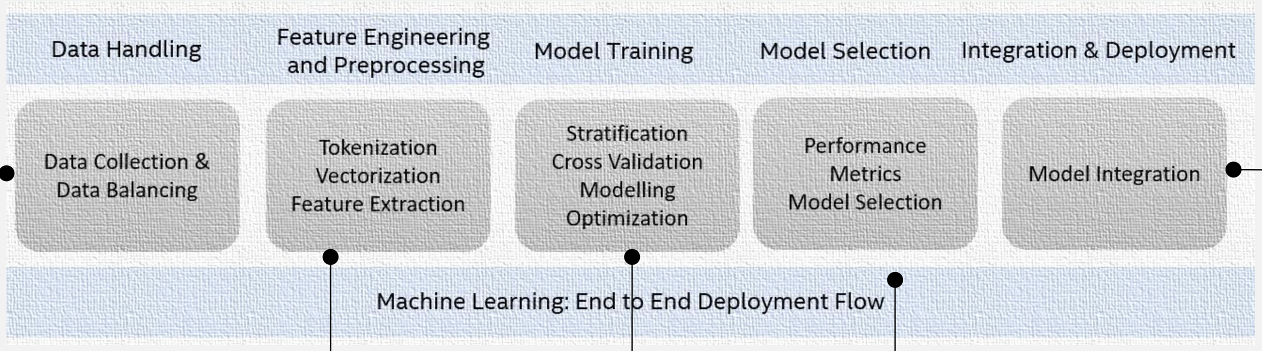


Solution

- Develop a machine learning solution for predicting errors and filtering noise in validation logs.
  - Combining natural language processing (NLP) with classification modeling was the optimal approach.
- Model automates the process of noise filtration and error triage in the logs.
- The image illustrates two workflows:
  ·The upper portion shows the existing process, which involves manual and semi-automated tasks for error identification.
  ·The lower portion represents the proposed integration of a machine learning model for a more efficient and effective end-to-end error detection system.



Machine Learning – End to End Deployment Flow



- Logs dataset comprises essential elements such as log filenames, descriptors of queries, classifications of queries as errors or informational, alongside the substantive text of log signatures.
- Strategic Data Balancing plays a crucial role in addressing the inherent imbalance between 'error' and 'info' query types within our dataset.

- Tokenization involved breaking down the cleaned log entries into individual words or tokens.
- Vectorization is the process of converting data into numerical vectors that can be used as input for machine learning models.
- Feature extraction is the process of transforming raw data into a set of measurable characteristics or features that can be used to improve the performance and accuracy of machine learning models.

- Stratification is a sampling technique used to ensure that different subgroups or strata within a dataset are proportionally represented in the sample.
- Cross-validation evaluates a model by splitting the dataset into multiple subsets, training on some, and validating on others, then averaging the results.
- Hyperparameter tuning optimizes a model's training parameters to improve performance by systematically searching for the best combination.

- Ensemble classification models include Random Forest, Gradient Boosting, AdaBoost, Bagging, Voting Classifier, and Stacking.
- Classification model performance metrics include Accuracy, Precision, Recall, F1 Score, ROC-AUC, Confusion Matrix etc.

- Save the trained model in a format suitable for deployment (e.g., pickle, joblib).
- Develop an API to serve the model predictions.
- Integrate the API with the existing system or application.

TARUN ARORA

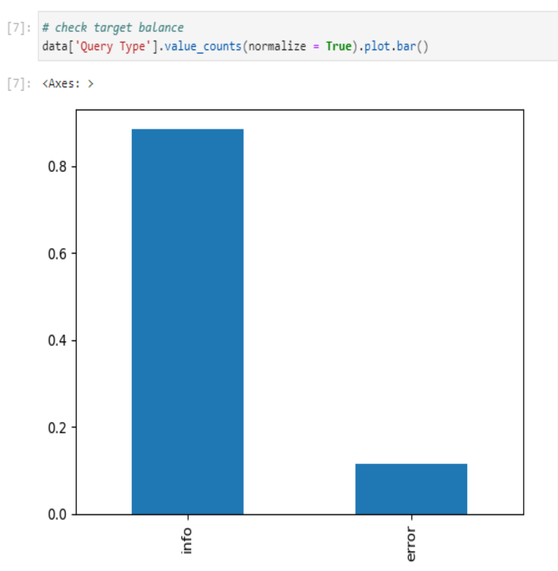
AI-Driven Techniques for Noise Filtration in Software Validation Logs

Natural Language Processing -

Brief Implementation Routine

Signatures are fine print from the logs which test, and debug team analyze to identify the errors and consider in defect triaging.

```
--------------------------------------------------------------
 No Errors Found from Arden Ramless Error Registers
 during the verify or read back of written data.
--------------------------------------------------------------
--------------------------------------------------------------
 No Master Cycle Data Mismatches Found
--------------------------------------------------------------
--------------------------------------------------------------
 No Atomic Cycle Result Mismatches Found
--------------------------------------------------------------
</FAILVECT>

Test Number: (2123230000)
Seed Number: (-2051737296)
+----------------------------+
| Full Interrupt Instance Log |
+-----------------------------+
| Results for Interrupt ID  32 |
+--------------------------------------------------------------------------------+
|Cyc:   ID Type        Vect Inst                                                 |
+--------------------------------------------------------------------------------+
3645:   32 FSB MSI  0x66    0 IRTE High: 0x000000000004b100 Low: 0x0000002300660001 Address: 0xfee00158 Data: 0x       0

Interrupt Status: PASS
```



- Logs signature texts are heavily skewed towards info as expected because logs generally have information which can't be categorized as errors
- As data is imbalanced, hence measures needs to be taken before modeling to balance this data.

Data Extraction and EDA

TARUN ARORA

AI-Driven Techniques for Noise Filtration in Software Validation Logs

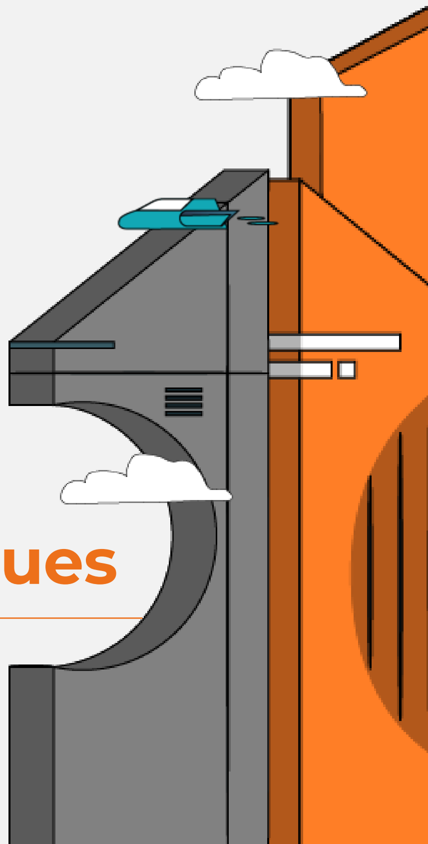
Natural Language Processing -

Brief Implementation Routine

```
def text_preprocessing(text_to_process):
# text preprocessing
    import re
    from nltk.corpus import stopwords
    from nltk.stem import WordNetLemmatizer
    # create a list text
    text = list(text_to_process)

    # preprocessing loop
    lemmatizer = WordNetLemmatizer()
    corpus = []

    for i in range(len(text)):
        r = re.sub('[^a-zA-Z]', ' ', str(text[i]))
        r = r.lower()
        r = r.split()
        r = [word for word in r if word not in stopwords.words('english')]
        r = [lemmatizer.lemmatize(word) for word in r]
        r = ' '.join(r)
        corpus.append(r)

    return corpus
```

- Removal of Stop words
- Lemmatize to reduce the words to its roots
- Removal of Date and Time Stamps
- Some Regex
- Tokenize the words

Preprocessing helps to reduce the noise features from input data for model



Basic NLP Preprocessing Techniques

TARUN ARORA

AI-Driven Techniques for Noise Filtration in Software Validation Logs

Natural Language Processing -

Brief Implementation Routine

```
# Tfid
Tfidf_vect = TfidfVectorizer(ngram_range=(1,2))
Tfidf_vect.fit_transform(df_balanced['Signature Text']).toarray()
Train_X_Tfidf = Tfidf_vect.transform(X_train)
Test_X_Tfidf = Tfidf_vect.transform(X_test)

# encode target variable
Encoder = LabelEncoder()
y_train = Encoder.fit_transform(y_train)
y_test = Encoder.fit_transform(y_test)
```

- Term Frequency-Inverse Document Frequency (TF-IDF)
- Word Embeddings: Word2Vec, GloVe, and FastText
- Doc2Vec: Extends Word2Vec to generate vector representations for entire documents
- Count and Hashing Vectorizer
- BERT: BERT captures the context of words by considering both preceding and succeeding words, leading to more accurate representations.

Vectorization

Natural Language Processing -

Brief Implementation Routine

```
# Classifier - Algorithm - SVM and Tfidf
# fit the training dataset on the classifier
SVM_Balanced_QueryName = svm.SVC(C=10, kernel='linear', degree=3, gamma='auto') #using the tuned parameters
SVM_Balanced_QueryName.fit(Train_X_Tfidf,y_train)

# predict the labels on validation dataset
predictions_SVM_Tfidf = SVM_Balanced_QueryName.predict(Test_X_Tfidf)

#calculating accuracy_score, precision, recall, F-Score and confusion matrix
print("SVM Accuracy Score using Tfidf, balanced data and Query Name-> ",accuracy_score(predictions_SVM_Tfidf, y_tes
print('Precision: ',precision_score(y_test, predictions_SVM_Tfidf)*100)
print('Recall: ',recall_score(y_test, predictions_SVM_Tfidf)*100)
print('F1-Score',f1_score(y_test, predictions_SVM_Tfidf)*100)
df = pd.DataFrame(metrics.confusion_matrix(y_test,predictions_SVM_Tfidf), index=['error','info'], columns=['error',
df
```

Models Examples :

- - Support Vector (SVM)
- - Logistic Regression
- - Random Forest Classifier
- - Gradient Boost/XG Boost
- - Naïve Bayes

Selected SVM and RFC Based on best results for precision and recall. Team wants model which is having high precision and recall for error prediction as they are more costly.

Model Generation

Natural Language Processing -

Brief Implementation Routine

	TFIDF_SVM	TFIDF_Logit	TFIDF_NB	TFIDF_RS_GS_CV Est: 50 , Depth: None	TFIDF_GB Est: 100 , Depth: 10, learningrate: 0.5	TFIDF_SVM	TFIDF_RS_GS_CV Est: 50 , Depth: None
	368	399	816	363	366	346	169
	60988	60949	59318	60997	60991	60993	263069
	263	302	1933	254	260	258	1484
	60860	60829	60412	60865	60857	60882	36946
FPR	0.60%	0.65%	1.36%	0.59%	0.60%	0.56%	3.86%
FNR	0.43%	0.49%	3.10%	0.42%	0.43%	0.42%	0.064%

RFC Performance Insights:

In contrast, the RFC model exhibited a significantly lower false-negative rate for 'error' predictions at 0.05%, albeit with a false positive rate for 'info' at around 3%. This suggests that while RFC is markedly adept at identifying 'error' queries with high coverage, it also presents a tendency to misclassify some 'info' queries as 'errors'.

SVM Performance Analysis:

The SVM model demonstrated a false positive and negative rate of approximately 0.5%, indicating a scenario where 0.5% of cases might be misclassified as 'info' when they are 'error', and vice versa. This rate, while low, underscores a critical challenge in distinguishing between the two query types with high reliability using SVM.

Model Selection

TARUN ARORA

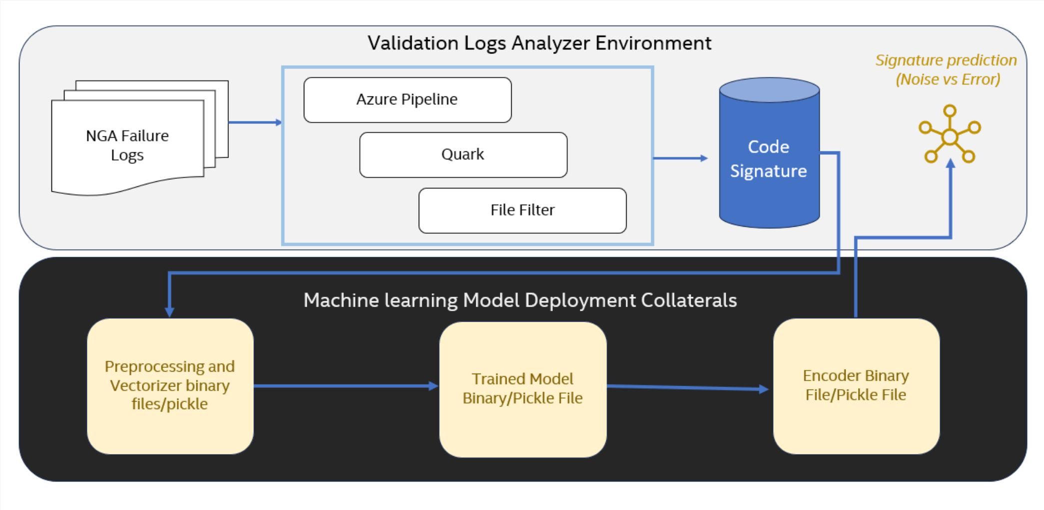
AI-Driven Techniques for Noise Filtration in Software Validation Logs

Natural Language Processing -

Brief Implementation Routine

The integration of the trained RandomForestClassifier (RFC) with Validation Flow enhances automated log analysis by autonomously identifying and categorizing failure signatures using advanced NLP and RFC.

This boosts diagnostic precision and reliability, optimizing maintenance strategies and system reliability. The figure below illustrates the integration flow.

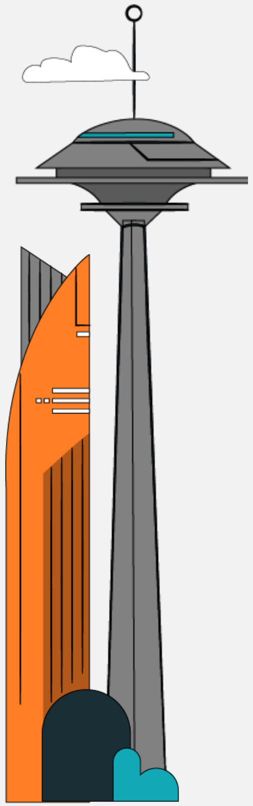


Deployment

TARUN ARORA

AI-Driven Techniques for Noise Filtration in Software Validation Logs

Key Learnings and Take Aways



- Integration Success: Its critical to integrate the model binaries with the engineering flow to leverage the model efficacy
- Enhanced Accuracy: The integration notably increased the accuracy of classifying error and information queries, surpassing traditional diagnostic methods.
- Efficiency Gains: Advanced NLP efficiently processed and interpreted complex log data, while RFC accurately identified and categorized failure signatures.
- Reduced Manual Intervention: The methodology demonstrated potential to greatly reduce manual intervention in system diagnostics.
- Future Potential: The project highlights the significant potential of combining NLP with RFC for log file analysis, setting a foundation for future advancements in software diagnostics.

- Stakeholders Collaboration and continuous interaction is key
- Pre-Processing and Model Generation is incremental Process
- Mere good accuracy may not help – we need to select model which suits the objective – in this case we choose model where precision and recall of Error is better compared to equal precision and recall of info and error

THANK YOU

THE FUTURE IS NOW

PNSQC.ORG OCTOBER 14-16 2024