Topic: Project Management and Data Science - Defects Triaging made faster

Many of us as project managers manage maintenance queues/problem management systems and QA bugs. Assignment of new defect/issue to right person with available bandwidth and expertise to resolve the issue must be ensured. There are critical times where even success of projects and programs depends upon how this activity is managed.

Triaging is generally done in a meeting or offline with a group of Tech Architects and leaders reading bugs and assigning the right area. Then project managers identifying right person with available bandwidth. It is a regular and time-consuming activity but important one.

A lot of rule-based automation is already done by organizations in these areas, but I am talking about the point where people must read the text and understand. That is where AI and Machine learning help.

The bug triaging AI program I have written takes bug summary, description, and comments as inputs. Bug Data is taken from freely available online Jira dataset. The component where bugs can be categorized generally do not change within organizations, so I have used supervised learning method LinearSVC which categorizes Bugs into their component/category area that requires a fix to resolve the issue.

Once the area to be fixed is identified, rule-based system can be used to assign it to right team or right people automatically.

This program also needs to have a feedback loop. To start with, model can be refreshed weekly to cater to new data points and exception corrections. However, this feedback loop can also be automated.

Project Management Note: The development of this work took 40 hrs. Actual production deployment needs the AI program to be integrated automatically with problem management systems or bug management systems like TFS/Jira. One also needs to work with tech SMEs closely initially to verify new assignments done by the program. This whole process can take about 80 hrs more. So first ready to use program can be available in 3 weeks of time. 2 hrs of developer time per week is required initially for monitoring, exception correction and feedback implementation. Soon, model will become robust and hopefully less time will be needed for maintenance in future

Dataset Citation :

Rath, Michael; Mäder, Patrick, 2019, "The SEOSS Dataset - Requirements, Bug Reports, Code History, and Trace Links for Entire Projects", <https://doi.org/10.7910/DVN/PDDZ4Q>, Harvard Dataverse, V1

Code inspiration:

<https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/>

Supervised Bug Triaging - Pseudocode

-----------------------------------------------------

1. Import all important libraries

2. issue comments summary and description are important text columns

3. concatenate multiple issue rows in to one based on issue id

4. Join different issues table for components text columns

5. Select important columns into a dataframe

6.Summary column is dataframe should not have nulls so drop all such rows

7. Then drop all rows which have components representing lesser number of rows

8.See if few component can be combined into one and combine those using regular data

9. Create a category id column in df\_is\_wth\_commf df and assing a numbr value to each category

10. Create a mapping table category\_id\_df to maintain mapping between categories and dictionary

11. Also create category to id and id to categoty dictionary to be used later on

12. Plot the graph to show number of occcurrence against each componetn on x axis

13. Data Cleaning - it is a very important step. First conver summary column into a list called data

I have categorized them into two

steps I performed

a. removed all URLs

b. removed all non alphabatic characters

c. removed multiple spaces, tabs and new lines

steps I have code for but have commented them

a. remove all stopwords

b. remove duplicate values

14. Now combline data list with original df\_is\_eth\_commf df and create new dataframe df\_data

15. Select important columns and drop nan values

16, Create features using tfidVectorizer and create label using category\_id

17.chi2 - chisquare test is used to find the features that are most relevant for comonent category

a. for eac category chi- squared stats is calculated

b. sort the score

c get the feature names from tfidf array

d. identify top 3 unigrams and bigrams and print

18. Split the data into train and test.Pass indices\_test and df\_data.index to refer to index of train, test and predict data laster and to join to original dataframe

19. Create list of models functions that are part of sklearn

20. Use cross\_val\_score to get model accuracies

21. For our purpose print only mean accuracy and standard deviation

22. As linearSVC returns best result Split data set again. Run the model. Predict values on test data as well

23. Print precision, recall, F1 score of the model using metrics.classification\_report function

24. Create confusion matrix on test values using confusion\_matrix function

25. Use sns.heatmap function to print confusion matric in heatmap format