Documentation of automation logics for the IC Fields:

**How Custom Jacard Similarity metric is calculated:**

🡪Inputs: Two texts: pred\_text, prod\_text

🡪Clean the two texts. Cleaning involves removal of stop words (both generic and custom stopwords), lemmatization of each word. Now take the intersection of words present in pred\_text and prod\_text.

🡪The Custom Jacard Similarity is (**no of common words between pred and prod texts/no of unique words in prod\_text)**

**Symptoms Field:**

Input: BugID

CDETS Enclosures used: Headline and RNE-symptoms

Logic:

🡪Concatenation of the above two enclosures into single text.

🡪Perform extractive summarization of above concatenated text using python **spacy** based **pytextrank**  module which gives scores to the given text by dividing it into phrases. Now rank the sentences in the given text by using those scores.

Hyperparameters: limit\_sentences=2

Accuracy:

Out of 25.5k bugs,

--->Percentage of bugs satisfying our Custom Metric (Custom Jaccard similarity>=0.5) is **77.1%** i.e **19.66k** bugs.

Remarks:

In addition to pytextrank based extractive summarization, we have tried out two more approaches, they are**: tf-idf** based extractive summarization and **word-embedding** based extractive summarization. But pytextrank is giving better results compared to them.

**Conditions Field:**

Input: BugID

CDETS Enclosures used: Rne-Conditions, Headline , RNE-symptoms, Description

Logic:

🡪If Rne-conditions is not empty, directly fetch it as the output.

🡪Elif check if Headline contains keywords such as ["when","upon","issue is seen","issue was seen","issue seen","issue is observed",”this vulnerability is seen in”] etc. and if yes, output the Headline as Output.

🡪Elif do the similar exercise as above using Rne-Symptoms.

🡪 Elif do the similar exercise as above using Description.

🡪Else output as “No specific condition”

Accuracy:

Out of **25.5k** bugs,

--->Percentage of bugs satisfying our Custom Metric(Custom Jaccard similarity>=0.5) is **65.8%** i.e **16.78k** bugs.

**Remarks:**

In conditions field, 10% of bugs have “no specific condition”. But the current logic gave us some conditions result. In that case, we believed to **not penalize** the model as we are providing extra information of conditions compared to production data.

**Impact description:**

Input: BugID

CDETS Enclosures used: RNE-symptoms, Description, Headline

Logic:

🡪 Concatenation of the above enclosures into single text.

🡪 Perform extractive summarization of above concatenated text using python **spacy** based **pytextrank**  module which gives scores to the given text by dividing it into phrases. Now rank the each **para** in the given text by using those scores.

🡪The para with highest score as well as satisfies few regex conditions such as( a)cosine similarity with Headline is above some threshold value, b)minimal alpha-numeric or special character based words within the paragraph, c)words count above certain threshold etc) is selected.

🡪The above selected para is outputted as model prediction.

Hyperparameters: limit\_para=1, limit\_phrases = 12

Accuracy:

Out of 25.5k bugs,

--->Percentage of bugs satisfying our Custom Metric (Custom Jaccard similarity>=**0.4**) is **65%** i.e **16.59k** bugs.

**Remarks:**

Here we considered threshold similarity to be **0.4** because we observed that within this threshold, majority of important **network related keywords** are getting captured.

**Steps-to-Repro Field:**

Input: BugID

CDETS Enclosures used: RNE-symptoms, Description, **Entire-Cdets-Notes**

Problems with available data: **23.3k out of 25.5k bugs are blank values.** So we end up with only 2184 bugs.

Logic:

🡪Concatenate the above mentioned Enclosures into single text document and search for different possible keywords like: ["steps-to-repro","recreate steps:","steps to reproduce","steps to re-produce:","steps to repro","repro-steps"] etc.

🡪If any of such keywords are found in the text, output the very next paragraph following the particular keyword.(An automated script is written to validate the output if it is valid steps-to-repro or not using regex conditions like numbering of sentences in paragraph etc)

🡪If none of the keywords are found in the text, output as “there is no repro”.

**Results:**

Out of **2184** bugs,

--->Percentage of bugs satisfying our Custom Metric(Custom Jaccard similarity>=**0.4**) with **non-blank** is **49.72%** i.e **1086** bugs.

Note: Even for missing values our model gives steps to repro for **3994** bugs, which is **rewarding**.

Remarks:

**Scope for improvement**: Currently attachment files in cdets are **not** accessible through cdets API, if we can able to procure that data, there is much scope to improve the results with the current logic.

**Trigger Field:**

Input: BugID

CDETS Enclosures used: RNE-symptoms, Description, **Entire-Cdets-Notes**

Problems with available data: **19.96k out of 25.5k bugs are blank values.** So we end up with only **5522** bugs.

Logic:

🡪If Rne-conditions contains the keyword “trigger” output it as required trigger field.

🡪Else Concatenate the above mentioned Enclosures into single text document. Now search for some custom keywords like: [“root cause”,"vulnerability is due","vulnerabilities are due","issue may be trigger","crash is due"] etc.

🡪if any of such keywords are found in the text document, output the very next lines following that particular keyword. (An regex script is written to check if it is valid output and discard invalid outputs.)

🡪Elif conditions field is not empty, output it as the required trigger field.

🡪Else output, “there is no specific trigger”

**Results:**

Out of **5522** bugs,

--->Percentage of bugs satisfying our Custom Metric(Custom Jaccard similarity>=**0.4**) with **non-blank** values is **21.27%** i.e **1175** bugs.

Note: Even for missing values, our model gives trigger field for **6554** bugs, which is **rewarding**.

**Remarks:**

As we have major values blank, still an extensive testing is required from SME’s for trigger field, to check if the prediction is true or not.

**Workaround Field:**

Input: BugID

CDETS Enclosures used: RNE-Workarounds, Headline

Logic:

🡪 Rne-workarounds + Headline

Accuracy:

🡪Metric used is cosine similarity.

Out of **25.5k** bugs,

--->Percentage of bugs satisfying our Custom **cosine** similarity Metric(Custom cosine similarity>=0.5) is **67.8%** i.e **17.3k** bugs.

**Other Remarks:**

We have also tried other metric called cosine similarity for Workaround.

**How cosine similarity is calculated:**

🡪Inputs: Two texts: text1, text2

🡪Clean the two texts. Cleaning involves removal of stopwords (both generic and custom stopwords), lemmatization of each word and finally vectorization of each text using sklearn’s Countvectorizer (with frequency).

🡪Calculate the cosine similarity between the two vectors i.e (**dot product of A and B**)/|A||B|