Mini-Project 3 Report

Task 1:

Task 1.1

We experimented with the data. We found that Task 1 is highly dependend on (1) the seed for random processes and (2) the specific methods chosen from the Failing Methods in the data set.

To combat issue (2) we chose to randomly choose two failing methods (i.e. 'HIT01_8', 'HIT02_24') to include in our holdout sets.

Issue (1) is harder to solve. By experimenting we identified that for most seeds there is no impact of 5% and 10% loss on precision and recall when adding more non-students to the holdout set. This makes answering the question task 1.1 poses more difficult. We hence perform the experiment for a range of seeds. Often times the precision and recall value does not fall below the thresholds regardless of how many non-students are added. This would mean that the average value would tend to infinity if even for one seed the threshold is not reached. This intuitively makes only little sense. We thus calculate the average in two ways: (I) by counting never reaching the threshold as instantly reaching the threshold and (II) just disregarding not reaching the threshold and only averaging the number of non-students added when reaching the threshold if a threshold is reached. For all thresholds we report both values.

Furthermore, we calculate which seeds results in reaching all (i.e., 4) tresholds and which just reach 3 tresholds. We will work with these thresholds in task 1.2. We chose a range between 0 and 250 for the seeds to balance runtime and meaningfulness of an average.

For Task 1.1. we gradually add at random “Non-Students” to the students holdout set and report on the degradation of the classifier’s precision and recall.

Baseline: 'Holdout Set' Metrics per Bug Report:

- overall: Precision=0.67, Recall=0.80
- HIT01_8: Precision=0.78, Recall=0.82
- HIT02_24: Precision=0.48, Recall=0.76

Average Non-Students to add to have an impact of 5% and 10% loss on precision and recall:

(I: by counting never reaching the threshold as instantly reaching the threshold)

Average Number of 'Non-Students' added to the 'Student Holdout Set' before

- Loss of 5% on Precision: 8.293

- Loss of 10% on Precision: 1.06
- Loss of 5% on Recall: 5.753
- Loss of 10% on Recall: 0.42

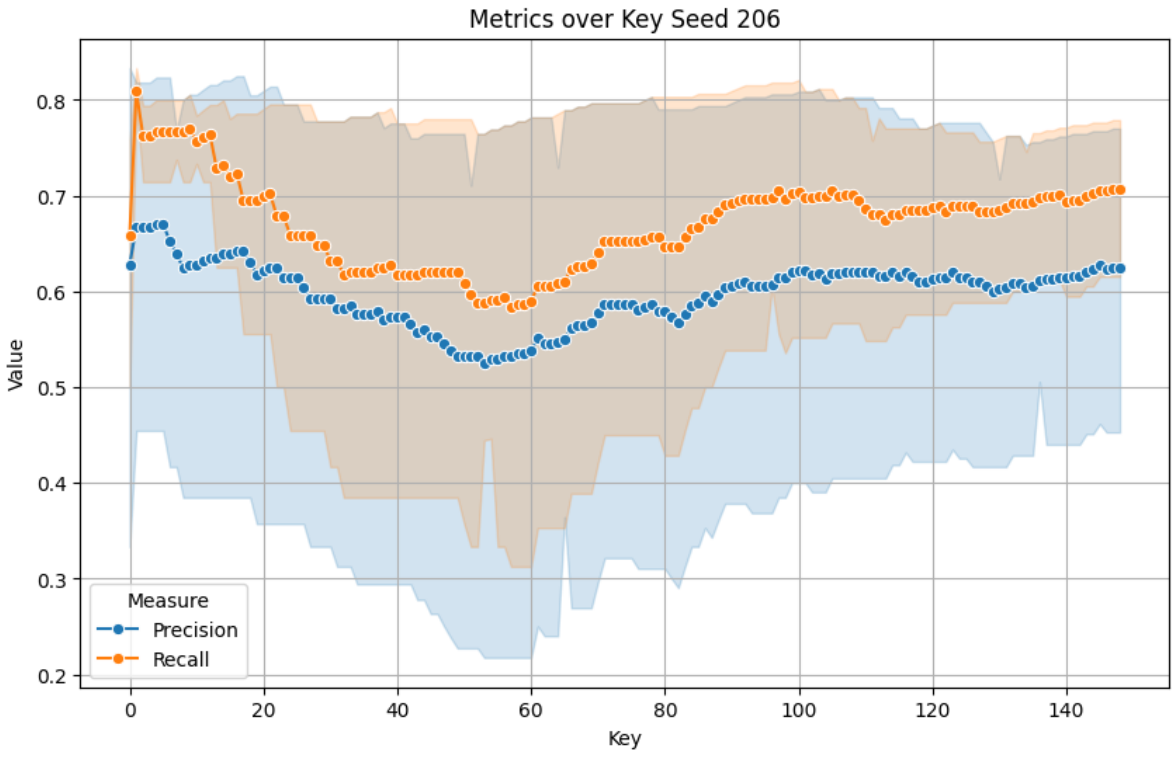
Cleaned: Average Non-Students to add to have an impact of 5% and 10% loss on precision and recall:

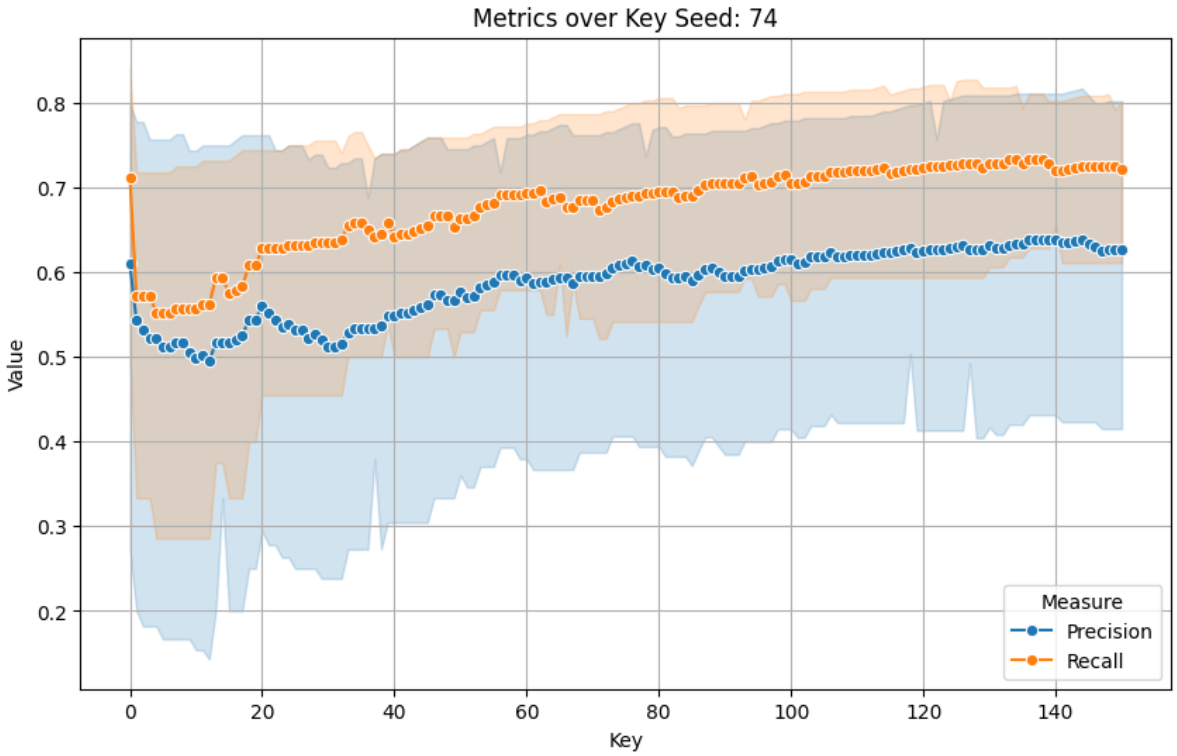
(II: disregarding not reaching the threshold and only averaging the number of non-students added when reaching the threshold if a threshold is reached)

Average Number of 'Non-Students' added to the 'Student Holdout Set' before

- Loss of 5 % on Precision: 22.618
- Loss of 10 % on Precision:: 26.5
- Loss of 5% on Recall: 15.411
- Loss of 10% on Recall: 7.875

Below we visualized the results for the seed 74 and 206. We picked seed 74 since it is the only seed in our sample where all 4 thresholds (i.e. loss of 5% and 10% on precision and recall each) are crossed. We picked 206 because three thresholds are reached only after adding relatively many non-students.





Task 1.2

What is the min number of “Non-Students” to train a model that produces similar outcome to the model trained on mixed data (from mini project 2)?

Due to rather long runtimes of the training process we calculated results only for one seed, even though we suspect this task to also be dependent on the random seed chosen. However, in this case the obtained results make intuitively more sense for this task, since values where obtained that do not correspond to infinity or 0 in comparison to task 1.1. Hence, we deemed the experiment with one seed to be sufficient. We chose seed from task 1.1 which we found to obtain values for all four metrics.

We first establish the baseline. We train the classifier on the full training set and evaluate on the full holdout set.

Baseline: ‘Holdout Set' Metrics per Bug Report:

- overall: Precision=0.66, Recall=0.75
- HIT02_24: Precision=0.42, Recall=0.61
- HIT01_8: Precision=0.80, Recall=0.80

We then start with the student training set and train a classifier and evaluate it on the full holdout set. In the following we gradually add more rows from the non-student training set to the student training set and train a new RF model for each. We perform predictions on the complete holdout set, to see how the model performs as the training data distribution changes.

We save the number of added non-students when various thresholds for the absolute difference between the baseline metrics and metrics for the newly trained models are reached. The number of minimum amount of non-students at various thresholds is finally reported.

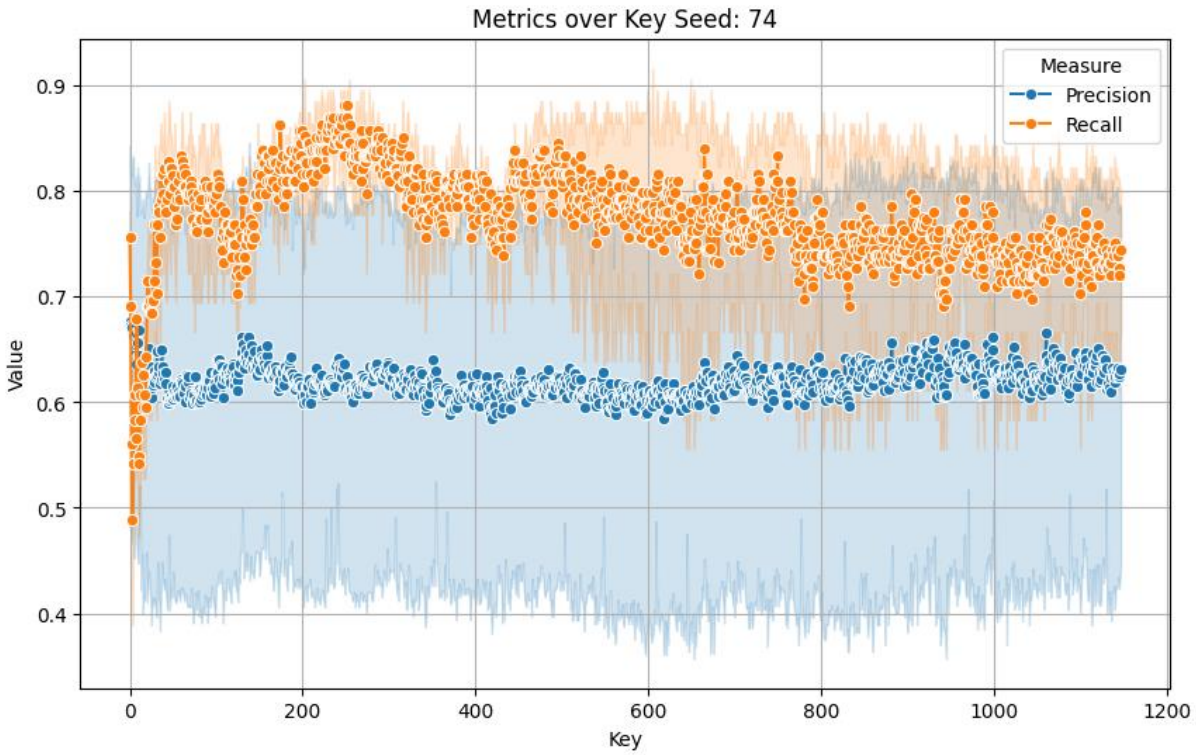
The minimum amount of non-students that need to be added to the 'Student Training Set' before precision and recall are within the following thresholds to the baseline precision and recall is:

Number of 'Non-Students' added to the 'Student Training Set' before

- Precision is within 0.01 absolute difference to baseline: 8
- Precision is within 0.005 absolute difference to baseline: 8
- Precision is within 0.001 absolute difference to baseline: 12
- Recall is within 0.01 absolute difference to baseline: 30
- Recall is within 0.005 absolute difference to baseline: 106
- Recall is within 0.001 absolute difference to baseline: 106

For the metric on all holdout Failing methods.

Finally, we visualize the precision and recall performance on the complete holdout data set with increasingly more non-students.



Task 2: Necessary and Sufficient Explanations

To generate the ground truth explanations, we first experimented with Bart - a model from the NLP domain that can be used for summarization tasks.

While the explanations generated by Bart contain useful information, they are often repetitive and not concise, e.g.:

"HIT08_54": " Lines 115/117 are looking at '_P' and the underscore fails the validation . Ch5 isn't an underscore . The input is in the wrong format and therefore the program correctly throws an error ."

In this task, we aim to generate necessary and sufficient explanations for each bug report, so we decided to try another model:

Llama 3.2 3B is a 3-billion-parameter multilingual large language model, optimized for advanced natural language processing tasks like dialogue generation, reasoning, and summarization. It should therefore be perfect for our task. Furthermore, it is also more recent than Bart.

We use the following prompts to generate our ground truth explanations:

```
{"role": "system",
"content": "You are an expert summarizer. Given detailed bug reports,
provide a single, comprehensive explanation that includes all necessary and
sufficient information to understand and fix the bug.",},

{role": "user",
content": f"Summarize the following bug reports:\n\n{combined_text}"},
```

With minimal post processing this provides us among others with the following summaries:

"HIT06_51": "The provided bug report describes a situation where the `addNumber` method in a Java class is not correctly handling the addition of numbers with decimal points. The issue lies in the comparison of the `exp` variable (which represents the exponent) with the original value of `x` (which is a double). The comparison is incorrect, resulting in an incorrect output.",

"HIT07_33": "The provided bug report describes a Java code snippet that appears to be working correctly, but is causing a `NullPointerException` when attempting to access an element of an array. The bug is caused by a missing null check on the `array` variable, which is being passed as an argument to the `toClass` method.",

"HIT08_54": "The provided bug report describes a scenario where the Java program fails to validate the input locale format correctly. The program expects the input to be in

the format `cc__CCCCC`, where `cc` is a country code and `CCCCC` is a country code followed by three more characters. However, the input `fr__POSIX` does not meet this format, resulting in a `NullPointerException` when the program attempts to access `ch3` and `ch4` in the `if` statement."

While there can still be some repetitions, overall, the generated explanations are of high quality and are more concise than the ones generated by Bart.

We then defined our metrics and thresholds.

For the readability , we will use the Flesch Reading Ease score to measure the readability of the explanation. The Flesch Reading Ease score is a measure of how easy a text is to read. The higher the score, the easier the text is to read. It is based on the idea that shorter words and shorter sentences are easier to read. The maximum score is 121.22. There is no lower limit, negative values are valid.

Most ground truth explanations have a Flesch Reading Ease score between 40 and 70, and are therefore of average readability. Given that the explanation of coding errors is often complex, this range is acceptable.

```
HIT03_6: 64.75
HIT05_35: 58.62
HIT07_33: 53.04
HIT02_24: 51.18
HIT04_7: 50.16
HIT08_54: 46.81
HIT06_51: 43.02
HIT01_8: 28.67
```

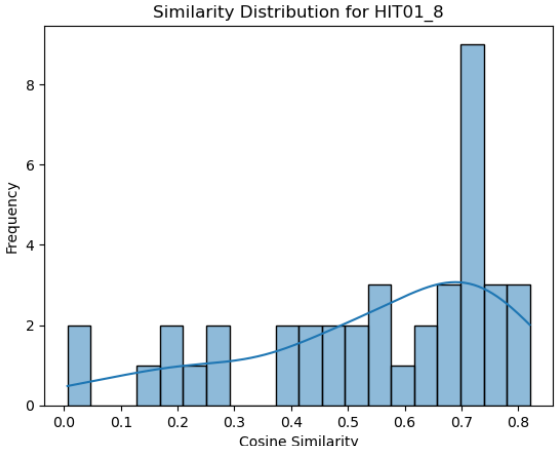
Since there are some deviations between the different explanations, we determine the threshold relatively per bug report. We set the threshold to 80% of the Flesch Reading Ease score of the ground truth explanation. We hypothesize that this threshold will be sufficient to filter out explanations of low quality and readability. Since the average readability score per bug report is slightly higher, this allows for some tolerance for changes in the readability caused by the model.

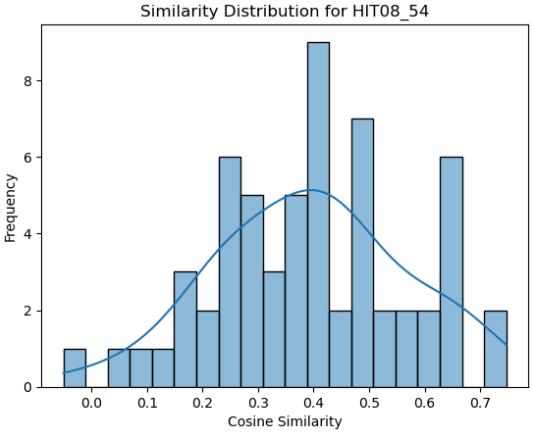
```
average readability of explanations per bug report

HIT01_8: 66.29499999999999
HIT02_24: 66.48525000000001
HIT03_6: 78.65999999999997
HIT04_7: 70.03829999999998
HIT05_35: 74.99766666666666
HIT06_51: 74.76399999999997
HIT07_33: 71.59666666666664
HIT08_54: 75.70616666666666
```

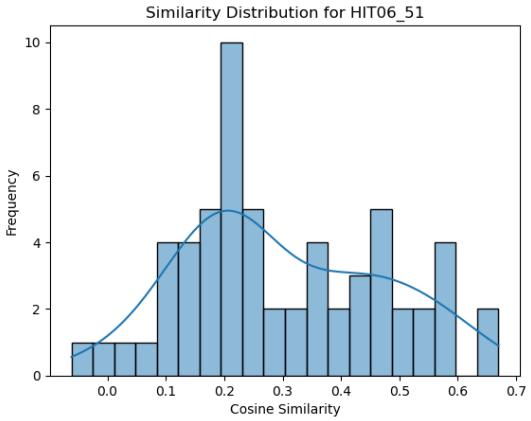
We will use Sentence Transformers to calculate the semantic similarity between the explanation and the bug report. Sentence Transformers is a Python library that allows us to use pre-trained models to calculate sentence embeddings. We hope that this will be more accurate than simple embeddings like Word2Vec or TF-IDF.

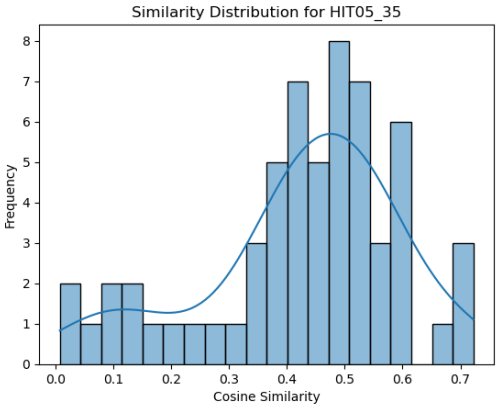
Average similarity per failing method

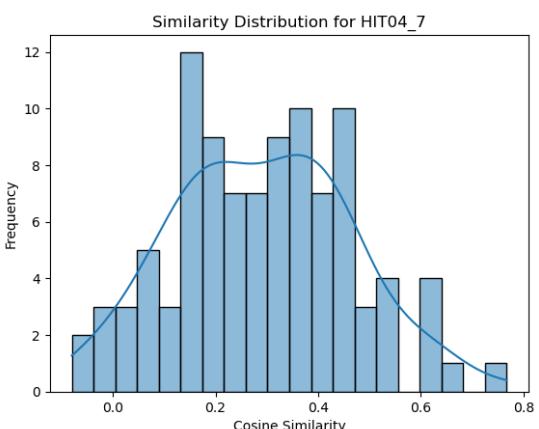
```
HIT01_8: 0.5436896428233012
HIT02_24: 0.3701791672501713
HIT03_6: 0.25586818529409355
HIT04_7: 0.2914702493418008
HIT05_35: 0.42632644800469277
HIT06_51: 0.30135327534129225
HIT07_33: 0.4918408831271032
HIT08_54: 0.3938455417131384
```

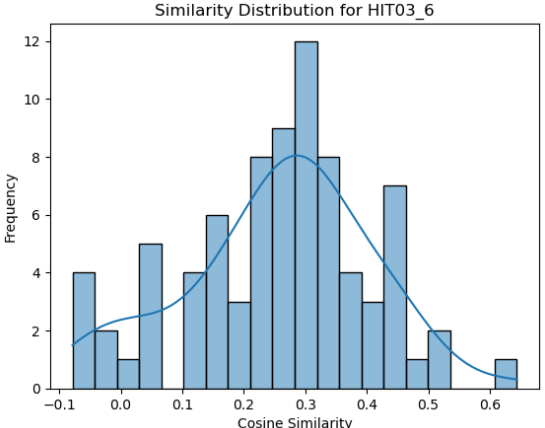


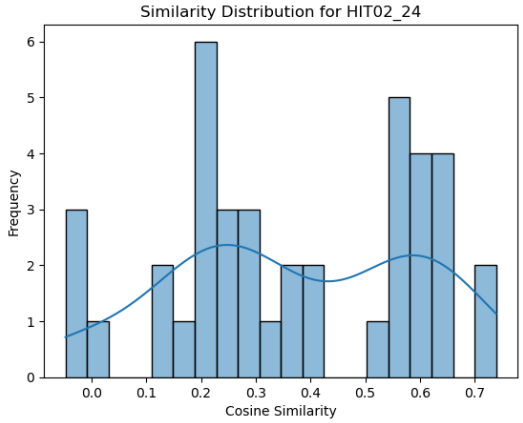


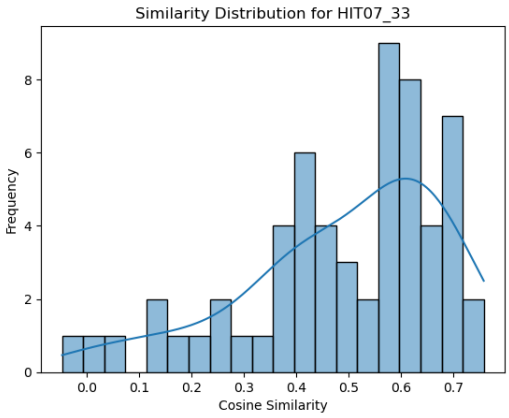












Looking at the average cosine similarity between the explanations and the bug reports, we can see that the cosine similarity is around 0.3-0.5. Because the average similarity per bug report still allows for a lot of difference, we decided to go with the 70th percentile per bug report of the cosine similarity as the threshold. We expect this threshold to be more challenging to reach and more interesting for the task.

We then try to find a minimal number of explanations that satisfy the thresholds after merging. For this we looked at the most readable explanations and checked the similarity score and vice versa.

We came to the conclusion that just because a method is highly readable does not mean that it has to be semantically similar to the ground truth explanation. However, if the explanation is highly similar then it is likely to be readable as well. The given explanations even meet the readability threshold already.

```
Similarity scores of most readable methods
HIT01_8: 0.018344800919294357
HIT02_24: -0.047606274485588074
HIT03_6: 0.031428221613168716
HIT04_7: 0.08298180997371674
HIT05_35: 0.5033749938011169
HIT06_51: 0.10949976742267609
HIT07_33: 0.04427710548043251
HIT08_54: -0.05031628906726837

Deviation from threshold
HIT01_8: -97.4%
HIT02_24: -108.43%
HIT03_6: -90.46%
```

```
HIT04_7: -78.38%
HIT05_35: -2.5%
HIT06_51: -73.68%
HIT07_33: -92.91%
HIT08_54: -110.55%

Readability Score for most similar methods
HIT01_8: 37.3
HIT02_24: 43.06
HIT03_6: 76.42
HIT04_7: 71.78
HIT05_35: 69.07
HIT06_51: 65.52
HIT07_33: 77.23
HIT08_54: 64.1

Deviation from threshold
HIT01_8: +62.63%
HIT02_24: +5.17%
HIT03_6: +47.53%
HIT04_7: +78.88%
HIT05_35: +47.28%
HIT06_51: +90.38%
HIT07_33: +82.01%
HIT08_54: +71.17%
```

We will therefore start with feeding the most similar explanations to the ground truth explanation to the model (Llama again) and then evaluate if we still meet the similarity and readability threshold.

Except for HIT02_24 we easily achieved the thresholds by a large margin with just one explanation:

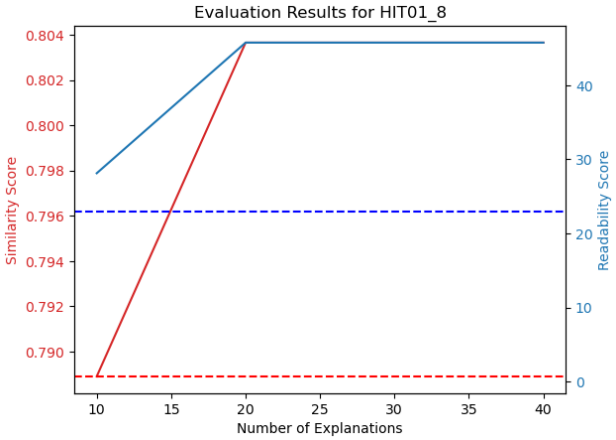
```
Deviation from Similarity Threshold
HIT01_8: +62.63%
HIT02_24: +5.17%
HIT03_6: +47.53%
HIT04_7: +78.88%
HIT05_35: +47.28%
HIT06_51: +90.38%
HIT07_33: +82.01%
HIT08_54: +71.17%

Deviation from Readability Threshold
HIT01_8: +62.63%
HIT02_24: +5.17%
HIT03_6: +47.53%
HIT04_7: +78.88%
HIT05_35: +47.28%
HIT06_51: +90.38%
HIT07_33: +82.01%
```

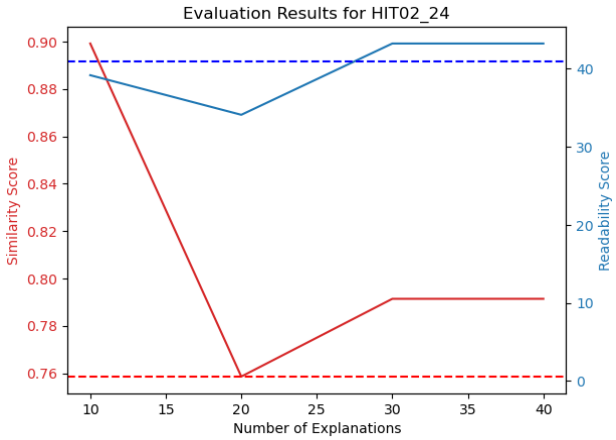
```
HIT08_54: +71.17%
```

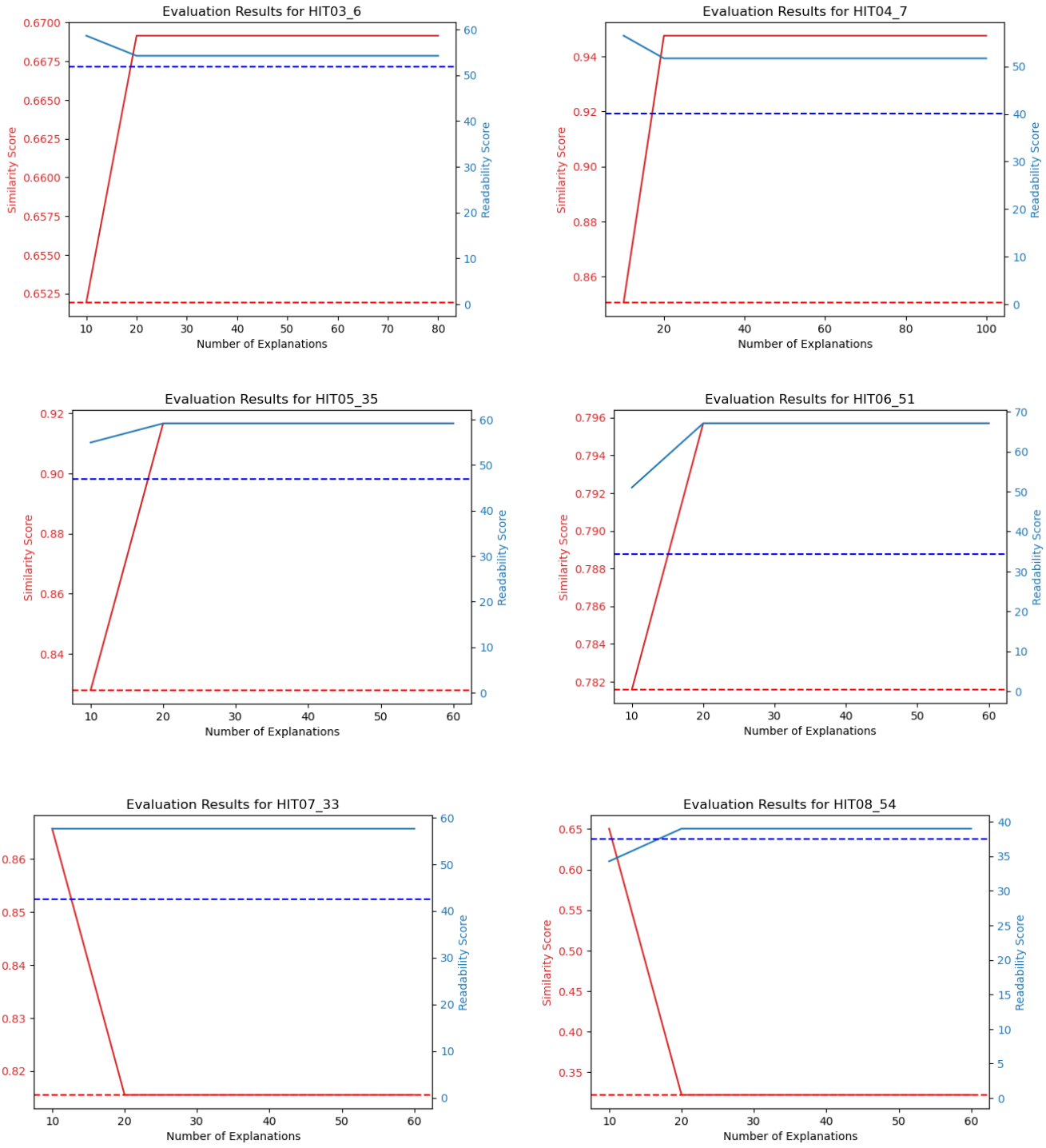
We expect that with more explanations the readability and similarity will go down as the model tries to cover more aspects of the bug report. We think it might therefore be beneficial to evaluate the model for every 10 explanations, starting with the 10 most similar explanations to the ground truth explanation.

Unfortunately, due to the limited amount of allowed input characters, adding more explanations quickly does not change the results as the input just gets cut off. However, some general trends can be observed. The results are very dependent on the bug report and the explanations. For some bug reports, the model is able to generate a good explanation with just one input, while for others it gets better with more inputs. The similarity score is often a lot higher than the threshold but in some cases it decreases a lot after more examples. This could be because compared to our original ground truth inputs, we rely on different explanations during this evaluation. While they are readable, they might not be as similar to the ground truth explanation.

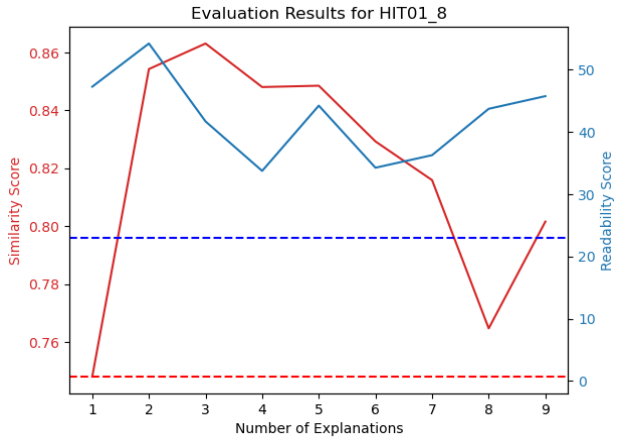
The readability score is overall closer to our defined threshold and only lies below it for 2 bug reports with few explanations. This is likely due to the fact that the model is able to generate a good explanation with just a few inputs, but the explanation may be relying on specific difficult wording from those explanations.

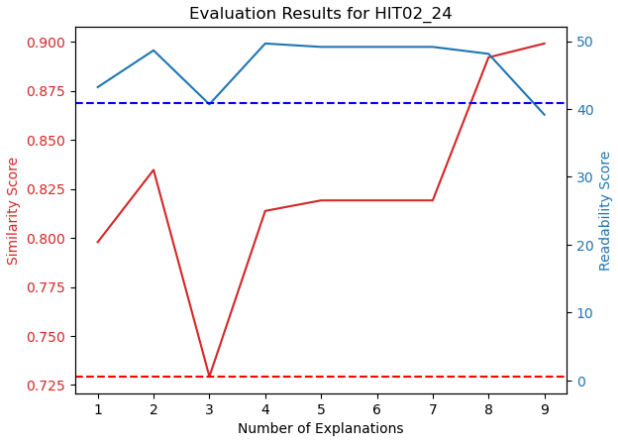


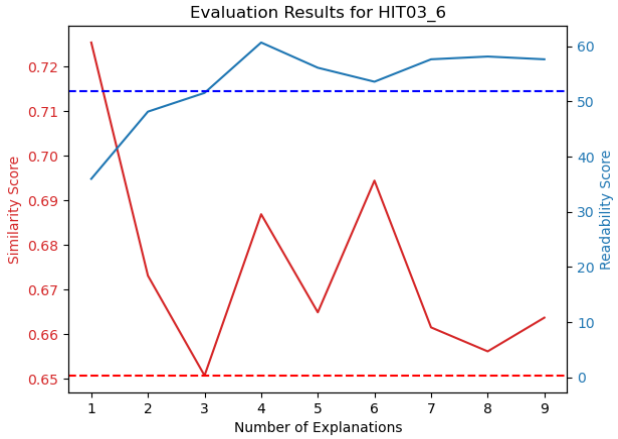


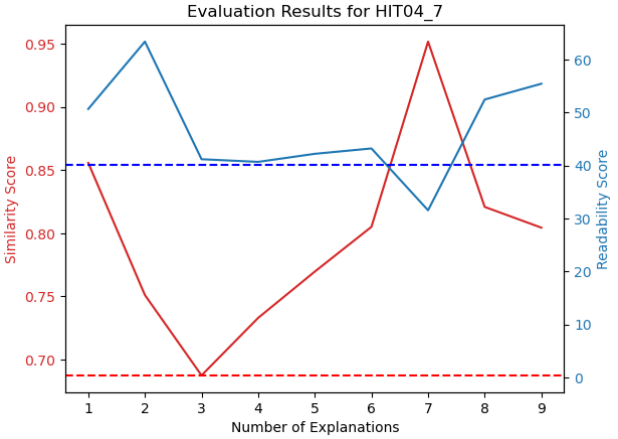


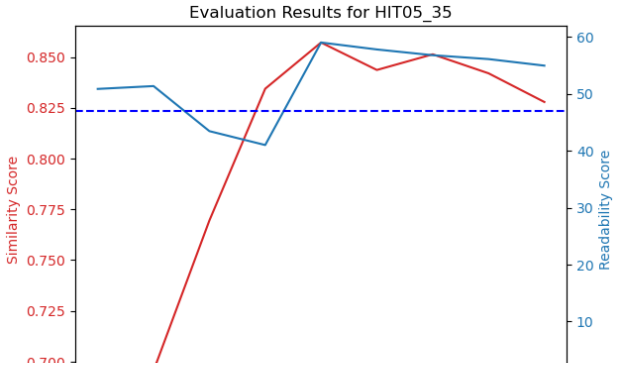
To get a more detailed impression, we now want to look at the top 10 similar explanations for each bug report and evaluate them.

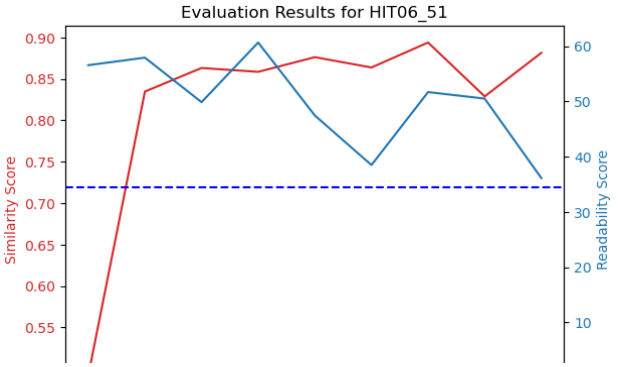

Evaluation Results for HIT01_8
Similarity Score
Readability Score
Number of Explanations

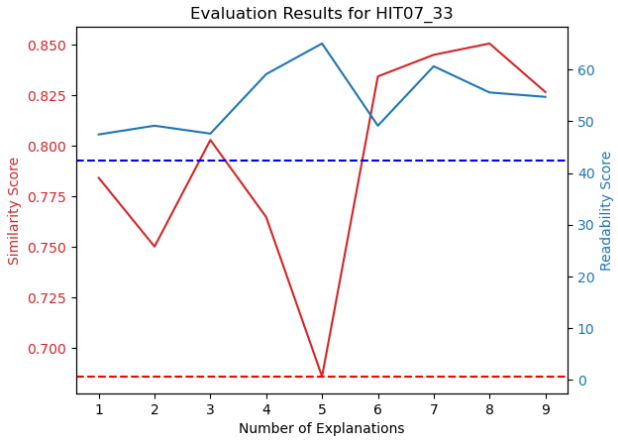

Evaluation Results for HIT02_24
Similarity Score
Readability Score
Number of Explanations

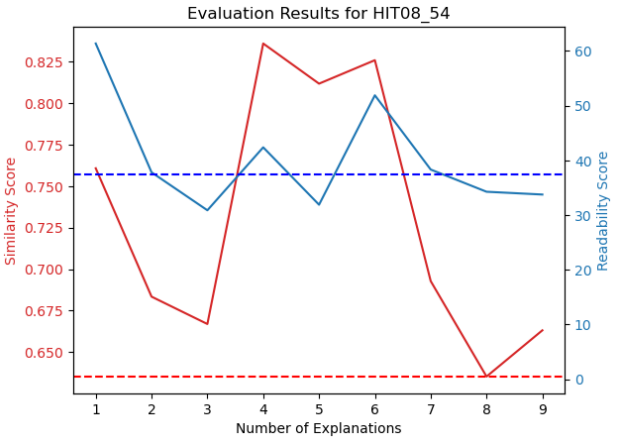

Evaluation Results for HIT03_6
Similarity Score
Readability Score
Number of Explanations


Evaluation Results for HIT04_7
Similarity Score
Readability Score
Number of Explanations


Evaluation Results for HIT05_35
Similarity Score
Readability Score


Evaluation Results for HIT06_51
Similarity Score
Readability Score


Evaluation Results for HIT07_33
Similarity Score
Readability Score
Number of Explanations


Evaluation Results for HIT08_54
Similarity Score
Readability Score
Number of Explanations

We can observe that in this case the results are a lot more variable. This is to be expected as we add a lot of new information each time relative to the input length.

The overall trend of the similarity score is almost a 50:50 splits between the methods: While half of them seem to improve with more explanations, many also have reduced similarity scores. However, when we compare to the more coarser evaluation results for 10 explanation increments, we can see that this does not necessarily reflect the overall trend. We can also see when the model heavily relies on one new explanation. For example, for the method HIT03_6 at 3 or 5 explanations the model seems to heavily rely on the new explanation which seems to be more dissimilar to the ground truth and reduces the similarity score.

Overall, the similarity is rather high which to some extend has to be ascribed to the fact that the model tends to use similar sentence structures and words.

The readability score shows stronger changes. This is in line with looking at the provided results as the exact explanations can also change a lot.

To summarize, it is sufficient to choose one very similar explanation to the ground truth explanation. This is also expected to reach the readability score as we defined it based on the readability of the ground truth. However, the model can still benefit from more explanations in some cases and further improve.

Task 3: Diverse Explanations

For task 3 we use the metrics produced in task 2 for similarity and readability.

(Q 3.1) There are many ways which we could use to measure the diversity of explanations. We could, for example, use the type-token ratio or the number of unique words in the explanations to measure the diversity of words used in the explanations. Since we did not get great results with the TTR in previous experiments, we will not use this approach. Instead, we want to regard the Shannon Entropy of the explanations. The Shannon Entropy is a measure of the uncertainty in a random variable. In our case, the random variable is the choice of words in the explanations. The Shannon Entropy is defined as follows:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \cdot \log_2(p(x_i))$$

where p(x_i) is the probability of the i-th word in the explanation. Shannon entropy measures the randomness or uncertainty in a distribution—in this case, the distribution of words in a text. Higher entropy means more unpredictability, while lower entropy suggests redundancy or repetition. The minimum entropy is 0, which occurs when all words are the same. The maximum entropy is the logarithm of the number of words in the text, which occurs when all words are equally likely. For normal written English, entropy usually ranges between 4 and 8 bits per word, depending on vocabulary richness.

We could also measure diversity by looking at the semantic diversity of explanations using the embedding distance (mean pairwise cosine similarity) between the explanations. Since this directly contradicts a high similarity between explanations, we decided against this approach.

In the broader range of diversity, we could look at more features than the explanation itself, e.g. at associated categorical features like the bug reporters demographic data, their experience level or the country that they are from. However, this makes it harder to define exact thresholds for diversity.

As the Shannon Entropy seems like the most straight forward approach, we will use this to measure the diversity of explanations in the following.

```
from collections import Counter
import math

def shannon_entropy(text):
    words = text.split()
    word_counts = Counter(words)
    total_words = len(words)
    entropy = -sum((count/total_words) * math.log2(count/total_words) for
count in word_counts.values())
    return entropy
```

As a baseline, we look at the Shannon entropy for the ground truth explanations:

```
HIT08_54: 5.358714497742255
HIT07_33: 5.25585347326784
HIT06_51: 5.115114023681427
HIT01_8: 5.10341455748809
HIT03_6: 4.999664476749764
HIT05_35: 4.898153434632013
HIT02_24: 4.8219280948873635
HIT04_7: 4.812209613812088
```

We can see that all our ground truth explanations achieve a Shannon Entropy of 4-6 bits per word. We will use this as a reference point for the diversity of explanations. This

moderate Entropy is in general desirable, as it indicates a good balance between clarity and lexical variety.

(Q 3.2) The max readability and max similarity values independent of the diversity per bug report are as follows:

```
HIT01_8: 0.820572555065155, 99.23
HIT02_24: 0.7392517328262329, 100.24
HIT03_6: 0.6439655423164368, 118.18
HIT04_7: 0.7673725485801697, 118.18
HIT05_35: 0.7223565578460693, 104.64
HIT06_51: 0.6698489785194397, 116.15
HIT07_33: 0.7574557662010193, 118.18
HIT08_54: 0.7468466758728027, 110.06
```

The readability scores are quite high because some people put very simple "explanations" like "yes" or "no" as their answer. To avoid artificially inflated scores, we filtered out explanations that did not contain English words or were shorter than 4 words.

However, the most readable explanations still tend to be very simple answers, indicating that readability alone might not be the best metric to optimize for.

Examples:
i++ should be i+
I am not sure.
type will be object when it should be string

(Q 3.3) To find the max diversity for the max similarity (compromising readability) we first look directly at the diversity of the methods with the highest similarity and compare it to the entropies of the ground truth:

```
HIT01_8: 4.829966150010236 vs 5.10341455748809 (GT), deviation =
0.27344840747785426

HIT02_24: 4.720049960644813 vs 4.8219280948873635 (GT), deviation =
0.10187813424255054

HIT03_6: 6.022673958060609 vs 4.999664476749764 (GT), deviation = -
1.0230094813108446

HIT04_7: 5.519999987133631 vs 4.812209613812088 (GT), deviation = -
0.7077903733215436

HIT05_35: 6.443530320573129 vs 4.898153434632013 (GT), deviation = -
1.545376885941116

HIT06_51: 5.576098444107486 vs 5.115114023681427 (GT), deviation = -
0.46098442042605914
```

```
HIT07_33: 4.004886164091841 vs 5.25585347326784 (GT), deviation =
1.2509673091759987

HIT08_54: 5.490861597695111 vs 5.358714497742255 (GT), deviation = -
0.13214709995285556
```

As we can see, for all of these cases the diversity is acceptable, although it is mostly slightly lower than our ground truth. Because of this, we decided to also look at the top 10 most similar explanations for each bug report and calculate the Shannon Entropy for each of them. We also regard combinations of these explanations to find the highest diversity for the highest similarities, i.e. the highest diversity may be achieved by combining the top 5 most similar explanations or by only taking one of the top 10 most similar explanations.

Example output (bug report, entropy of joined explanations, explanations)

HIT02_24: 6.538425914613637, explanations (8): ('Since the exception seems to be thrown up by Color constructor (seeing message - color parameter outside of expected range); there is a problem with the value of g.', 'depending on the upperBound and lowerBound values; "g" might exceed -255 or 255 which is not a valid value for the Color object.', 'variable "g" might be outside the range of the Color class acceptable range values.', 'You are calling the Color constructor with three float parameters so the values are allowed to be between 0.0 and 1.0.', 'The variable is being defined correctly as a parameter of the getPaint method.', "value really only has to be an int since valid values are forced 0 to 255 later. I imagine it is a double for consistency with other color schemes used in other methods that may use the full range. but by itself shouldn't cause any problem with any functions used in getPaint", 'The exception is coming from Color; so it must be g that has a bad value. I need to see the definition of this.lowerBound and this.upperBound to know what is wrong though.', 'there is an issue with colors in that programme ')

This approach seems to favour multiple and thus slightly longer explanations. With this, we reach rather high entropy scores. This is a good sign, as it indicates that the explanations are diverse and not just repetitions of the same information. The average readability is a bit low, however, this is a trade-off that we are willing to make to ensure that the explanations are diverse. We would also expect this value to increase when feeding the explanations to the LLM.

Conclusion

Combining these insights with our previous results for task 2, it seems logical to curate a set of ~10 explanations for each bug report, which are the most similar to the ground truth and have a high diversity. This way, we can ensure that the explanations are both accurate and diverse. Since readability already comes to some extend with the similarity, it can be neglected or only slightly optimized for. We can also imagine that the readability can easily be tweaked by prompting the LLM to generate more readable explanations, while similar approaches to similarity and diversity are not as straight forward.