Mini Project 3

Cedric Lorenz, Leonard Dreessen, Oliver Hess, and Raphael Reimann

Task 1: Distribution Shifts

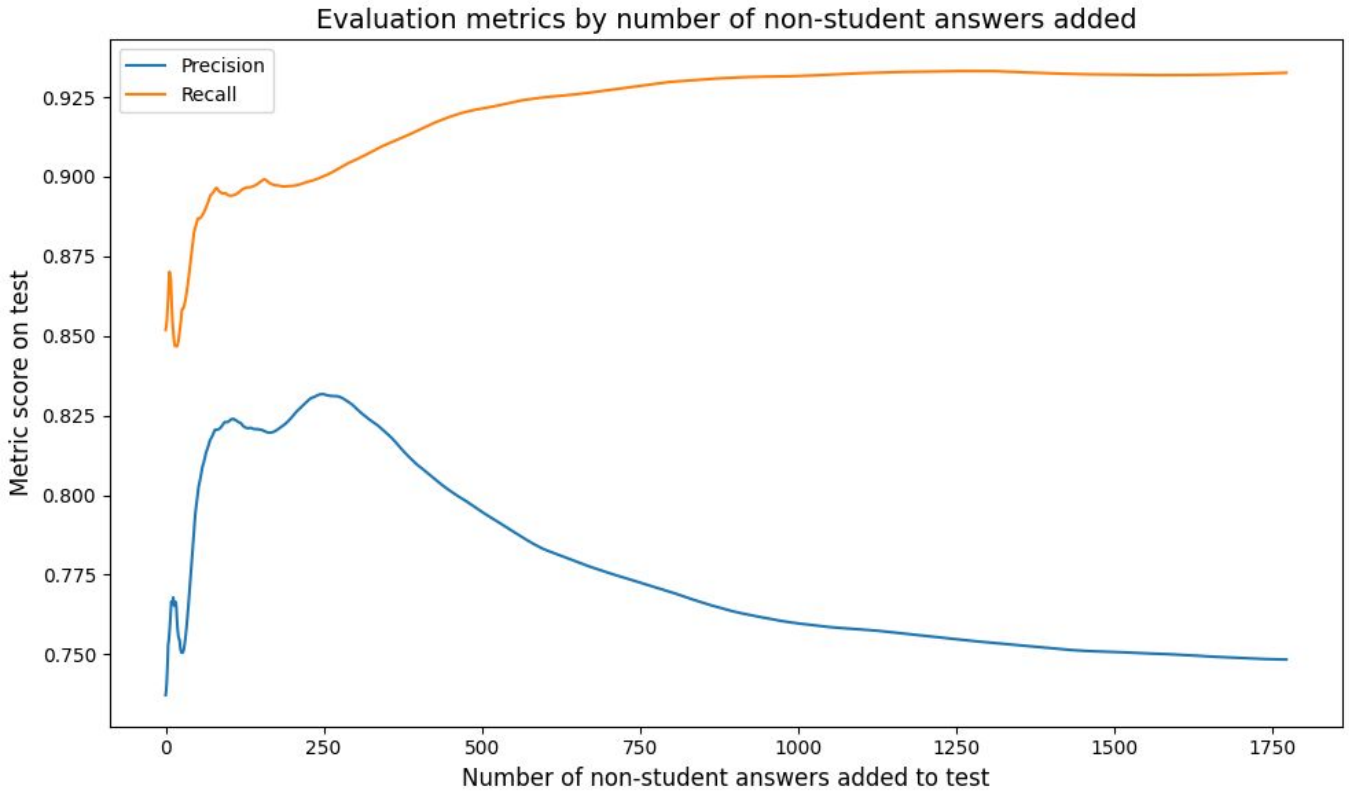
Goal

- The goal of this task is to measure the impact of changes (distribution shifts) in one input feature (”Participant’s Profession”).

Implementation Details

- Random Forest Classifier model with similar setup to Mini Project 2
  - Only based on student answers (807 out of 2580 total answers in data set)
  - FailingMethods 0-5 used for training, 6 & 7 fixed in test/holdout set
- Gradually adding a random non-student answer to the test set
  - No multiple choice of a non-student answer possible.
  - So at the end, the test set consisted of the student answers for FailingMethods 6 & 7 and all non-students answers.

Task 1: Distribution Shifts



Task 1: Distribution Shifts

Question 1.1: For the impact of 5% and 10% loss on precision and recall, what is the min number of "Non-Students" added on average to the holdout set?

- Compared to the baseline where the holdout set consists of student answers only, there is no negative impact on recall or precision regardless the number of added non-student answers (apart from a tiny loss on recall after adding around 20 non-students)
- Recall: with more non-students in the holdout set the recall increases by a log shaped curve
- Precision: increases as well until 275 added non-students and then drops logarithmically
- It's not surprising that precision drops at a certain point of recall increase because of the trade off between both metrics (see https://medium.com/analytics-vidhya/precision-recall-tradeoff-79e892d43134)
- Apparently, it is easier for the model to classify non-student samples than student samples which is surprising, but can have arbitrary reasons that need to be further investigated (next slide).

Task 1: Distribution Shifts

Question 1.1: Next steps to investigate

- Rerun experiment with different subsets of non-student samples added to test (professional, programmer etc.) and check if similar trend is observed
- Check the similarity of explanations between different groups:
  - some groups might be very similar to students
  - some groups just might be easier to classify for the model

  → that might be linked to the participants skill level or the motivation during participation in the study

Task 1: Distribution Shifts

Question 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)?

- We compare to our results from the “one test set permutation” experiment from Mini Project 2, which is also based on the same FailingMethods (6 & 7) in the test set: **Precision**: ~0.763 **Recall**: ~0.931
- Around ~900 non-student answers in context of Mini Project 3 added in order to achieve similar values for both of these metrics. But…
    - When only considering Precision: in the range of ~35-900 non-student answers added, actually the student model performs better than our mixed-data model from Mini Project 2. After this range, we definitely see a loss in precision.
    - When only considering Recall: as Recall overall increases with more non-student-samples seen, up from the mentioned ~900 non-student answers added, the student model starts outperforming the model from our prior project.
- Further investigation notes from Task 1.1 also apply here, rerunning the experiment with different random-states and subsets might be useful.

Task 2: Necessary & Sufficient Explanations

Goal

- Identify a minimal subset of human-provided bug explanations such that, when consolidated by an LLM, the result is both readable and semantically aligned to a “gold standard” explanation.

Data Sources

- To create the golden dataset, we take all explanations for a subset of the Question IDs (n=8, one for each code file). We manually create each golden explanation by including all information from the explanations. That way we have a singular explanation that includes everything the respondents have answered
- We filter the explanations for true positives and true negatives and join them on the golden explanations

Metrics

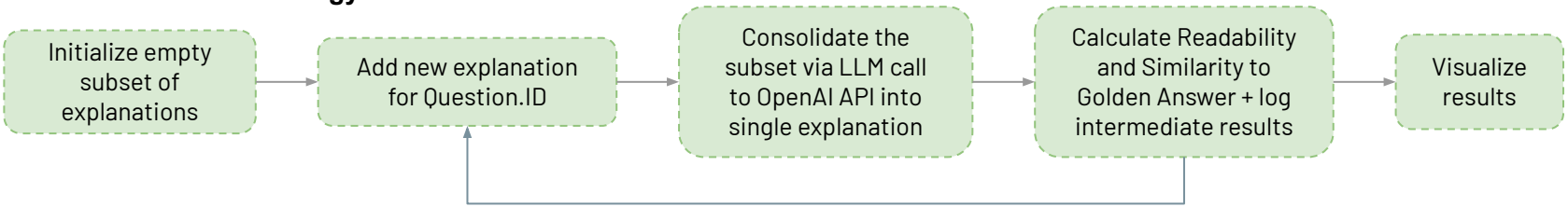
- Readability: Flesch-Reading-Ease (>= 60 considered understandable, `textstat` library)
- Semantic Similarity: Cosine similarity (>= 0.85 considered similar, `sentence-transformer` using the `all-MiniLM-L6-v2` embedding model)

> The issue is `value`, which results in `g` being outside the valid range for a color value. Color components are expected to be within 0.0 and 1.0 (for floats) or 0 to 255 (for integers), but `value` is -0.5 which yields an invalid negative result. `value` is checked against `lowerBound` and `upperBound`, but the resulting variable `v` is not used. The original `value` is used incorrectly. The calculation using `value` produces a `g` value that is negative and invalid for a color object. Ensure that the variable `v` is used instead of `value`. Correct the code to properly clamp `value` within the range of `lowerBound` and `upperBound`.

Example Golden Explanation for Question.ID == 10

Task 2: Necessary & Sufficient Explanations

Our Naive Subset Strategy



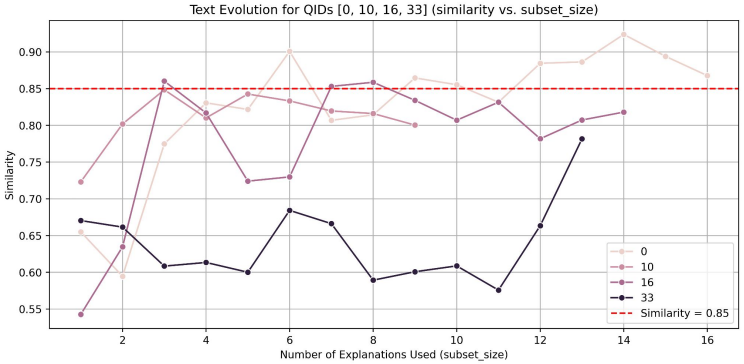
- We start from a single explanation and then naively add more explanations to the prompt so that the LLM has more information to generate a consolidated answer that is close to the golden answer
- Limitation: We add explanations in an arbitrary order. That means that adding a new explanation with a lot of information early in the iterative process might lead to an early spike in similarity
- Limitation: We use the first Question.ID for each of the eight code files. It was not clear from the task description that not every Question.ID discusses a bug. We now essentially look at at how many explanations are needed to be similar to a golden explanation that is a manually crafted digest of all explanation of a given Question.ID

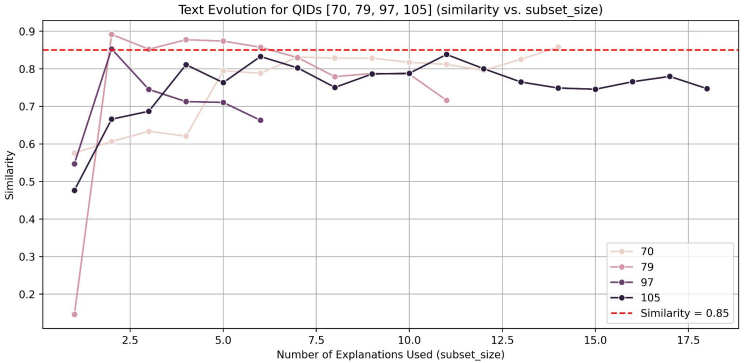
Note: The exercise says to stop adding explanations once the threshold is reached. We visualize the threshold but keep adding explanations to see how the metrics change after exhausting all available explanations

```
You have multiple bug explanations from
different programmers.
Your task is to merge them into a single
explanation that:
1) Contains all necessary details to understand
and fix the bug,
2) Avoids redundancy and repeats,
3) Is concise and readable.

Here are the explanations each separated by a
newline:

{joined_explanations}
```

Prompt Template

Similarity Scores + Reflection

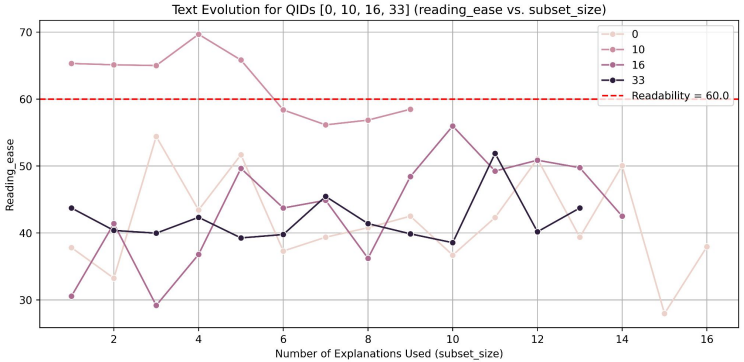


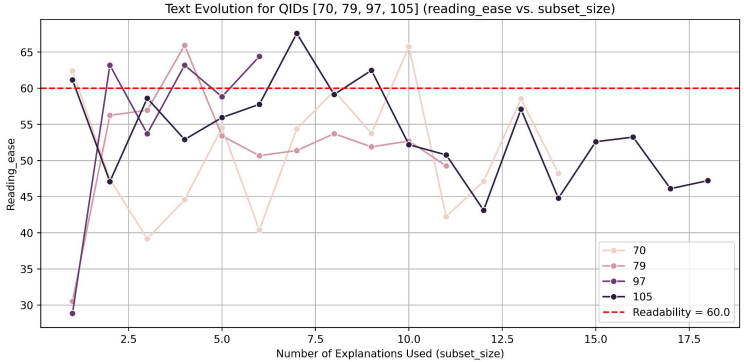


- For most questions, the LLM gets close to the golden answer after adding 2-3 answers
- Note: Order of explanations is important, adding many explanations with little information leads to consolidated explanation with low similarity for a long time, e.g. Question.ID 33
- In general, adding more examples after initial few does not improve semantic similarity much or even declines
- For most Question.IDs the LLM-generated consolidated explanation is similar to the golden answer

LLMs are a good tool to produce bug explanations with a minimal set of existing explanations.

Reading Ease Scores + Reflection





- The Question.IDs have a different number of explanations, which is why not all lines have the same length
- Most consolidated explanations have a low reading ease score since they are highly technical with long sentences and dense explanations
- The explanations with higher scores, e.g. Question.ID 10, are the ones that feature less complex variable names
- Adding examples does **not improve** the **reading ease** score.
  -> Adding more examples should not have an impact on the LLMs ability to generate readable text
- Adding explanations forces the LLM to include complex variable names or programming concepts in their consolidated answer which leads to decreased reading ease with more explanations added

Not many examples are needed to consolidate them to a reasonable answer with an LLM.

Task 3: Diverse Explanations

Goal

- Select a set of explanations which are highly readable, semantically similar to a ground truth, and have high inter-explanation diversity
- Solution: **Brute-force** find subset of size **N=3** which optimizes the metrics to variable degrees

Data Sources

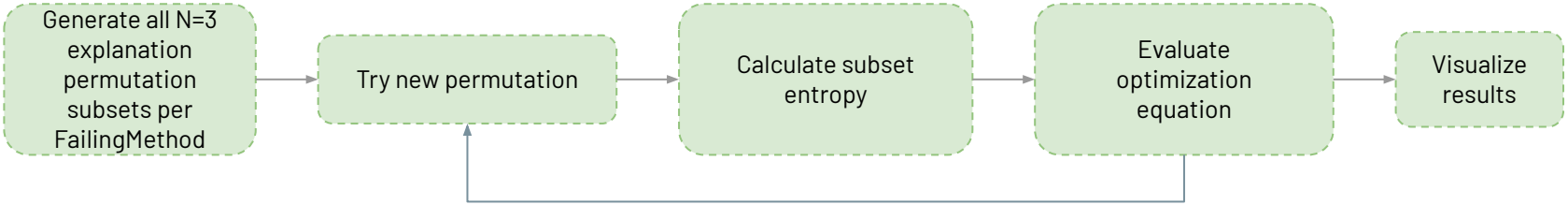
- User explanations (answerData) where TP == 1 (rows which correctly identify the bug)
- Ground truth explanations for comparing similarity (n=8, one for each code file)

Metrics

- Shannon Entropy (base=2)
    - Categorical Variables: Use them as is and feed entropy formula with discrete probability dist.
    - Numerical Variables: Convert numerical range into 10 bins and use as discrete distribution

Task 3: Diverse Explanations

Brute-Force Optimization Strategy per FailingMethod

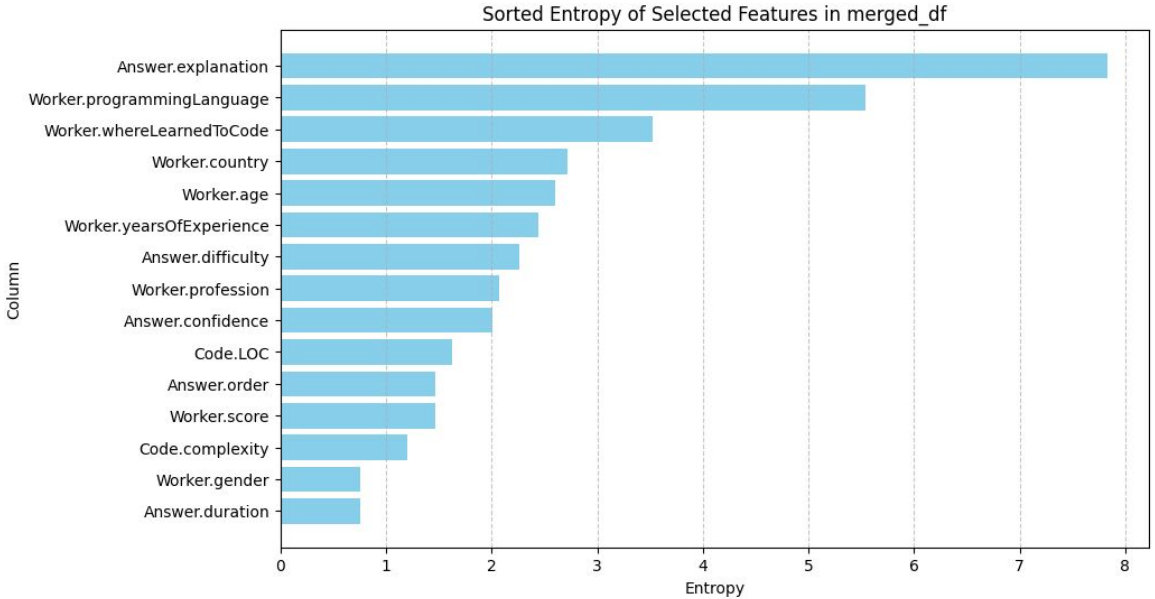


Optimization Equation

$$score(x) = \sum_{i=1}^{N=3} \alpha \times readability(x_i) + \beta \times similarity(x_i) \times entropy(x_i)$$

where x is a permutation consisting of three explanations and alpha, beta and gamma are weights regarding readability, similarity and entropy. Importantly, all metrics are normalized to have a range between 0 and 1 in order to ensure fair weighting.

3.1 Diversity Measurement



Entropy over all rows and features

- Calculate Shannon Entropy for each column and sum together for final subset value
- Use answer, code and worker features
- Not using Worker.ID because entropy is high, but for each Question.ID there is only one answer per worker
- Explanation itself and programming language have highest entropy
- Will recalculate entropy for each subset as indicator for diversity

3.2 Max Similarity and Readability

Max reading ease score: 121.22

Max reading ease score entry:

‘Yes’ 

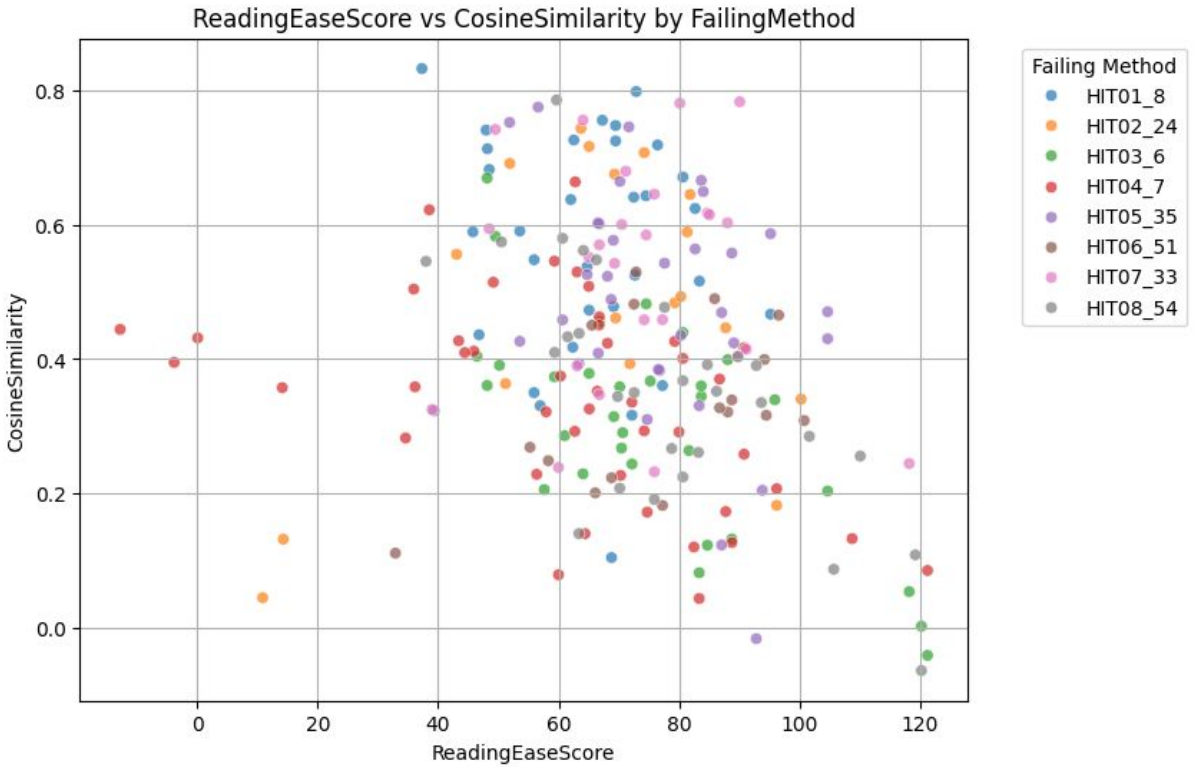
Clearly, reading ease should not be the only metric to optimize.

Max cosine similarity: 0.8326233625411987

Max cosine similarity entry:

‘The variable "minutesOffset" is checked incorrectly by the IF statement on line 279. Any negative value for "minutesOffset" will throw this exception; while the documentation states that "minutesOffset" can be negative in some cases.’

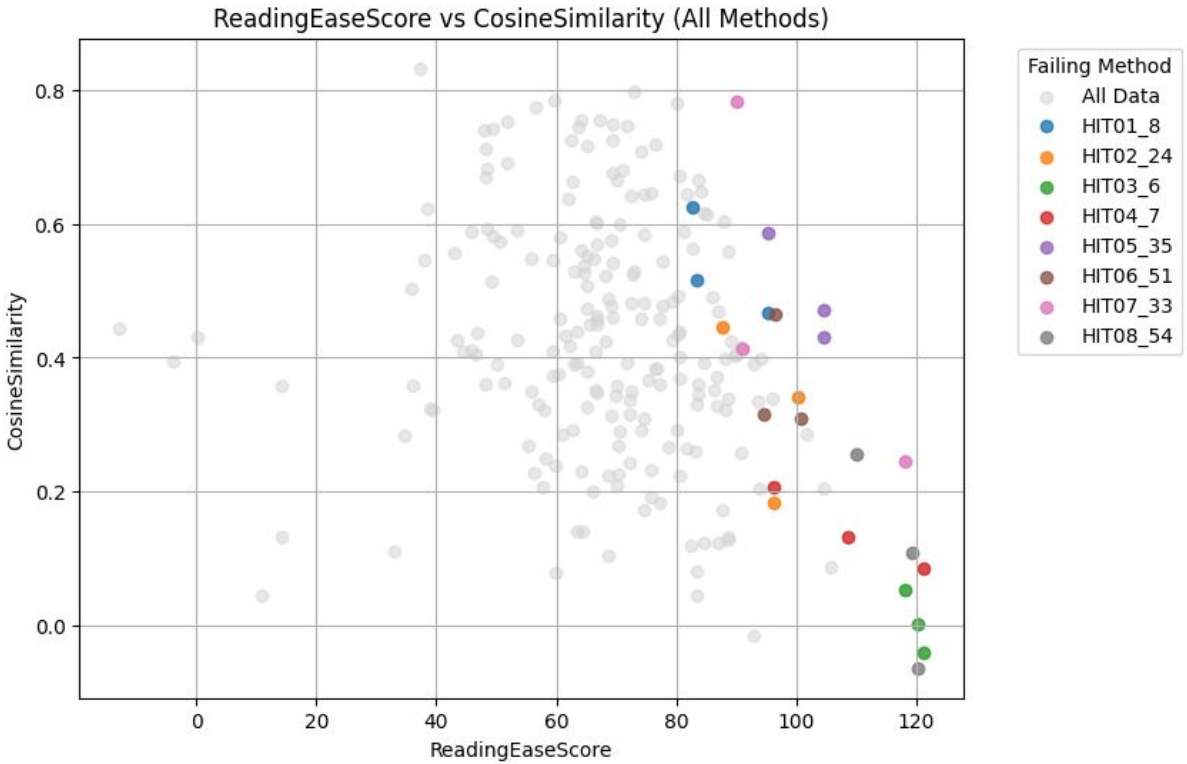
Distribution of Reading Ease and Similarity



- There is a wide distribution of readability vs. similarity
- Notably, there are no non-readable entries which have high similarity, indicating that the golden explanations must be readable.

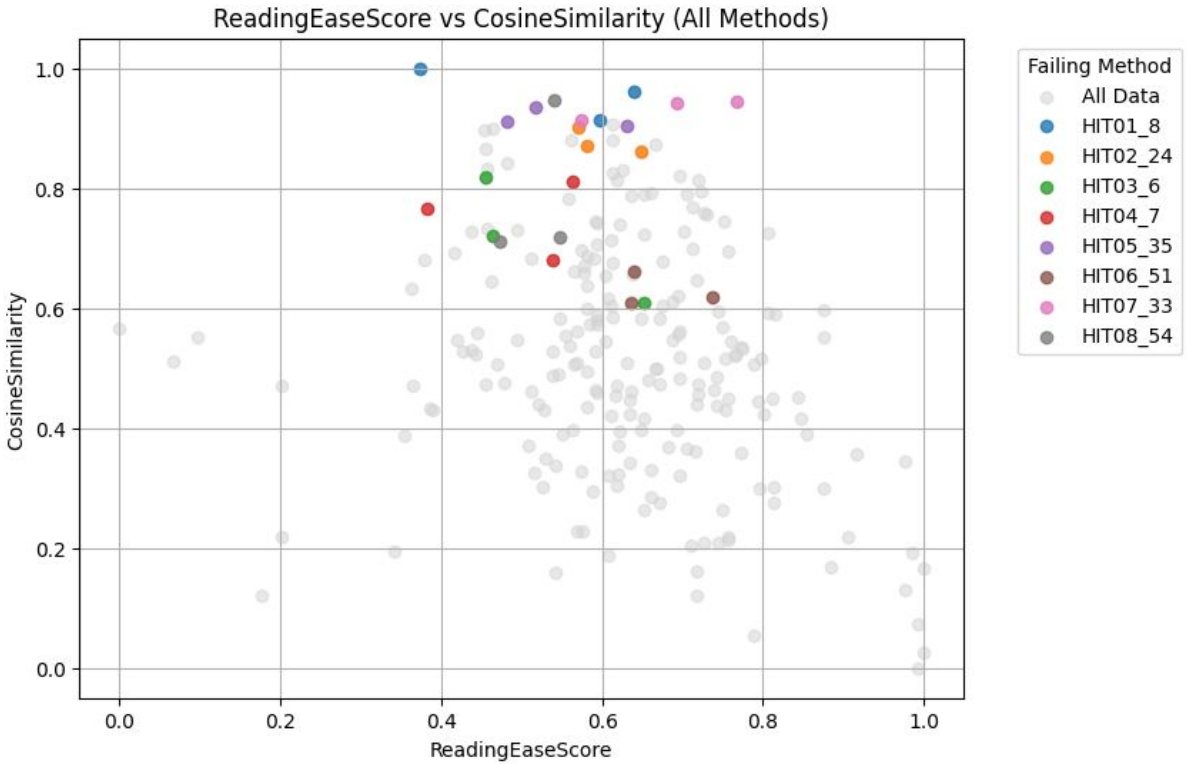
- In the following, we will apply different optimization configs and compare the subset allocations in this plot.

Max Readability only



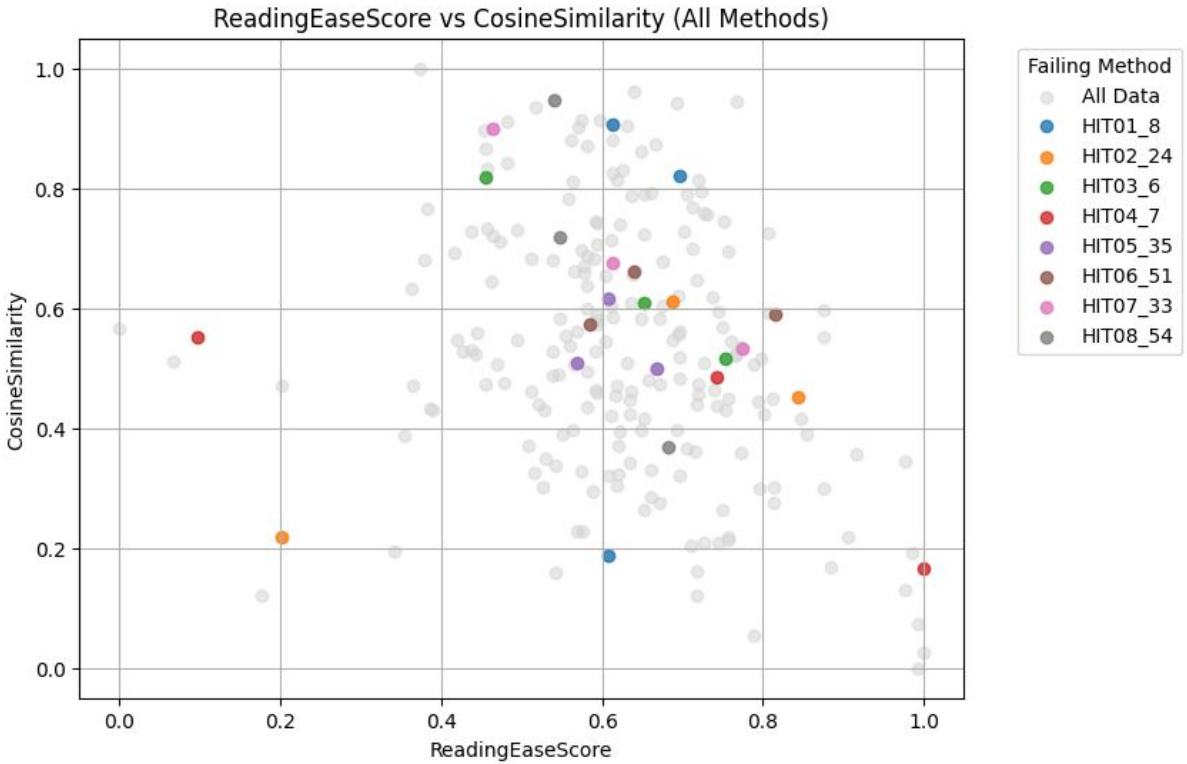
- alpha = 1.0 in order to only optimize for readability
- Dots are allocated to the right of the diagram, indicating correct optimization for readability.

Max Similarity only



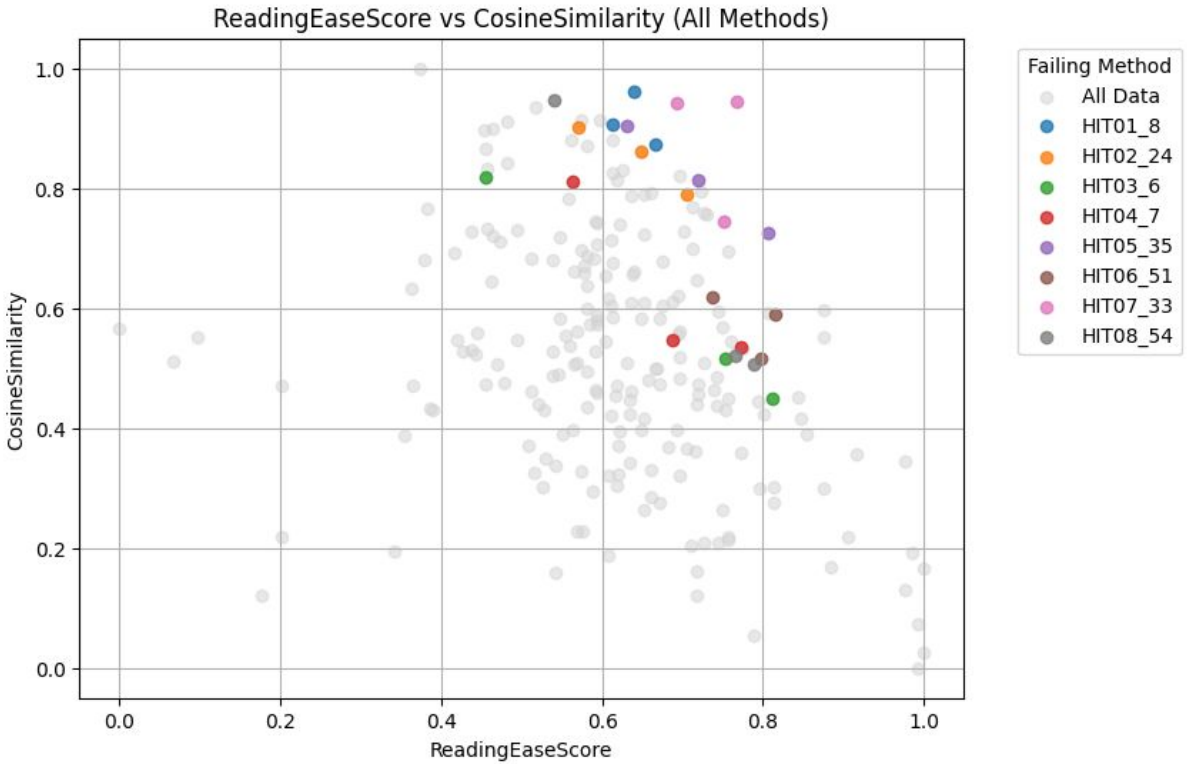
- beta = 1.0 in order to only optimize for similarity
- dots are allocated to the upper section of the diagram, also indicating correct optimization for similarity.

Max Diversity only



- gamma = 1.0 in order to only optimize for entropy
- Distribution of dots is somewhat random. This makes sense because entropy is not plotted in this diagram and can only be calculated per subset.
- A bar chart comparison of entropies of different optimizations can be found on a later page

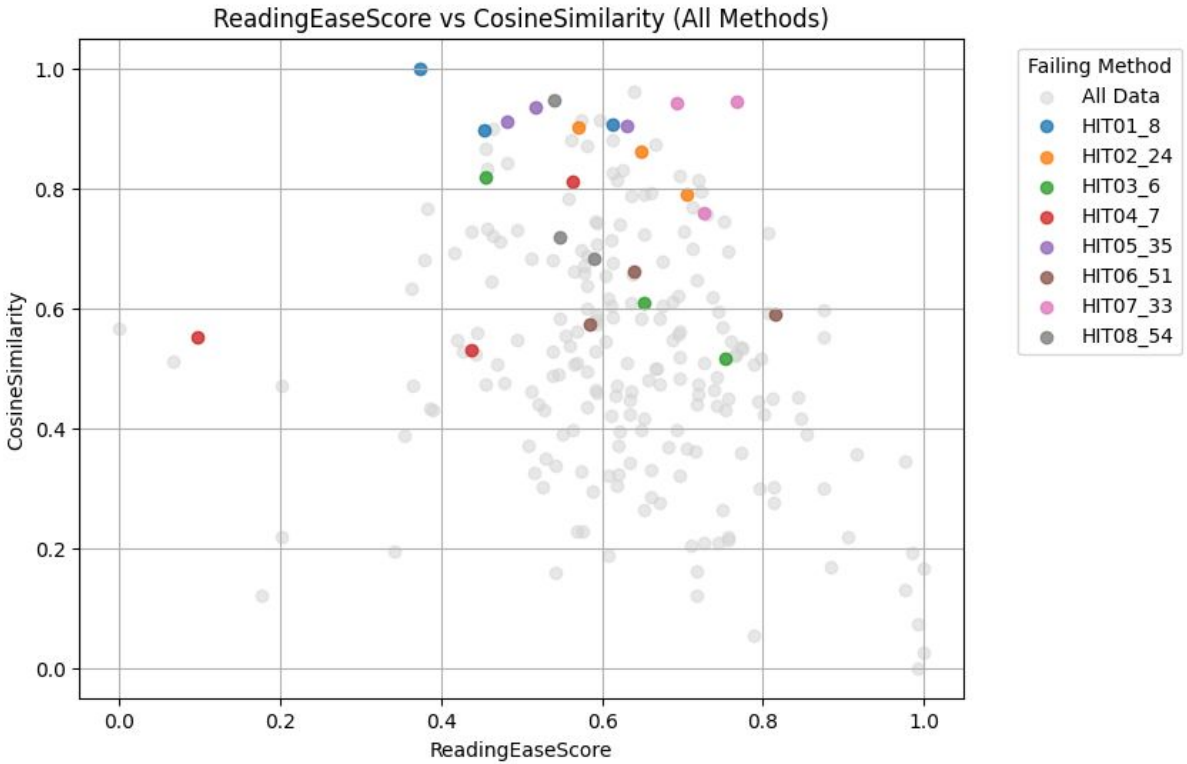
3.2 Max Similarity and Readability



- alpha = 0.5 and beta = 0.5 in order to not optimize for entropy but for readability and similarity equally.

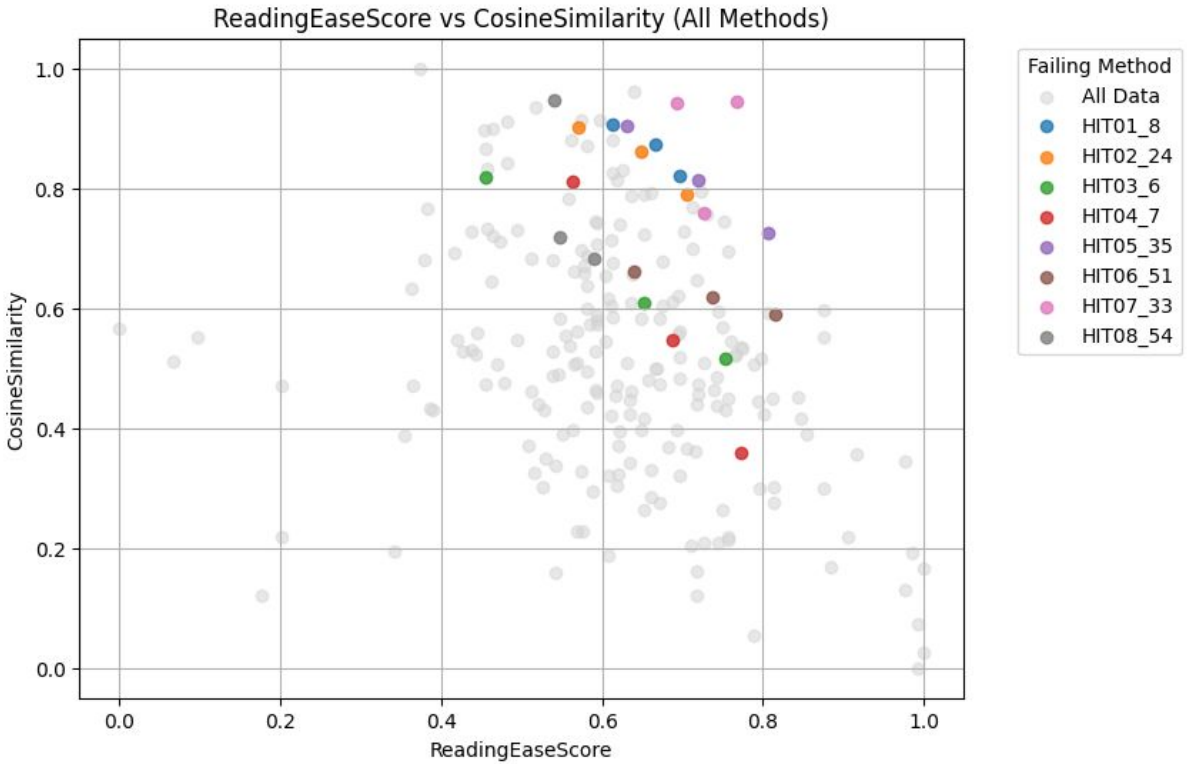
- The dots are distributed to the upper right in order to both satisfy readability and similarity.
- Naturally, entropy of subsets will be lower.

3.3 Max Similarity and Diversity (not Readability)



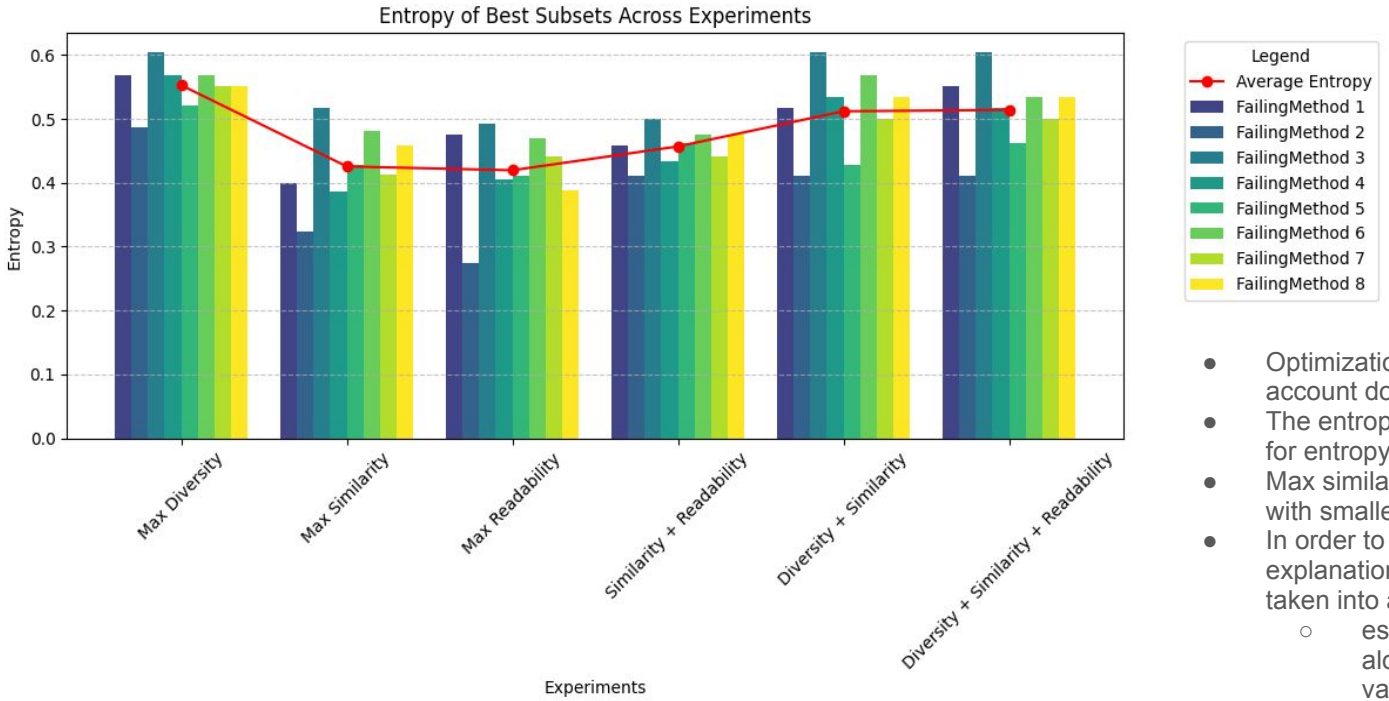
- gamma = 0.5 and beta = 0.5 in order to not optimize for readability but for entropy and similarity equally.
- The dots are tending upwards but are still somewhat randomly distributed due to the entropy optimization.

3.3 Max Similarity and Diversity and Readability



- alpha = 0.33, beta = 0.33 and gamma = 0.33
- The dots are tending towards the upper right but are still somewhat randomly distributed due to entropy.

Entropy Comparison of all Optimizations



- Optimizations where entropy was taken into account do have higher summed entropies.
- The entropy is highest when only optimized for entropy (as expected)
- Max similarity and readability both correlate with smaller entropy
- In order to receive a well-rounded subset of explanations, all three metrics should be taken into account
    - especially since high readability alone does not have high semantic value

Task 3 Further Reflections

- The optimization method is brute-force, has exponential runtime and does not scale to higher subset sizes. Therefore, we needed to restrict it to N=3.
- In turn, we were able to find optimal combinations for our scoring function.
- However, more elaborate scoring functions should be researched in the future.
- There could have been different approaches, e.g. having a variable amount of N per FailingMethod to make it more flexible
    - For example one could have chosen all explanations above the similarity and readability thresholds defined in 3.2 and then optimized entropy.