0.1 Determine the Robustness given Sequence Length

So far, the experiments were conducted on a maximal sequence length of [25]. Also, we conducted each experiment[FOR XIXI: In terms of wording, should I change 'experiment' to 'simulation'?] on the same dataset. [MENTION the RQ to have an approach which is *process model agnositc*] With these result, we are not able to to claim, that thew model consistently outperforms the other approaches. Therefore, this section will explore the results on different data-set sizes and types.

0.1.1 Experimental Setup

For this experiment, we run the same experiment of ?? on all the datasets, that where introduced in ??.

0.1.2 Results

Again, Figure 1 shows a clear dominance of the evolutionary model ([Model Name]) across all datasets.
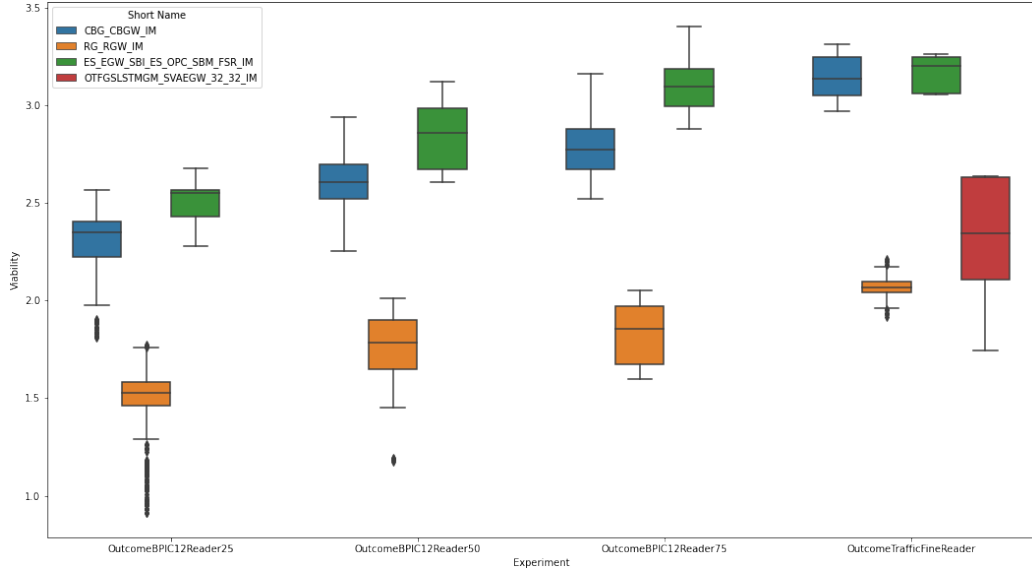

Figure 1: This figure shows boxplots of the viability of each models' generated counterfactual.

The results show that the performance remains consistent for [for a specific model]. Here, [a specific model] returns an avarage viability of [some value]. Hence, we can savely assume that the model outperforms the baseline approaches in general.

However, a notable observation occurs in the processing time. Although all models require more time to produce their results, if we increase the maximum length of sequences, the evolutionary algorithms require substantially more time. **??** shows, the processing time is much higher than the baseline models. On average, [evo model] require [duration in seconds], while [other models] require [XX], [XX] and [XX], respectively.

0.1.3   Discussion

The results show that [... TBD]. The increase of generation time can be explained by the viability measure. More specifically, the current implementation of the Semi-strucured Damerau-Levenshtein distance (SSDLD) within the sparsity and similarity measure have a quadratic time complexity. The time complexity primarily depends on the maximal sequence length. The number of cases that are compared is less of a factor as we use a highly vectorized implementation of the distance using numpy.