Given a dataset D with an original feature set F_0, a downstream machine learning model M, and an operator set O, a new optimal feature set F^* can be constructed from F_0 and O. In an ideal scenario, if the representational power of the operator set O is unlimited, this set may contain an optimal feature f^*, which directly describes the relationship between the original features F_0 and the target label.

In this simplified scenario, given a new feature f constructed from F_0 and O, the optimization goal for feature f is to maximize its similarity to the optimal feature f^*. For convenience, we measure similarity by comparing the syntactic structures of f and f^*. Specifically, the optimization objective is to maximize their largest common subtree size. If f and f^* are identical, their similarity S reaches its maximum value. (We also need to minimize the size of f, because if f is too large, the model needs to reverse-engineer the effective part from it.) When $S(f, f^*) = \mathrm{order}(f^*)$ and $\mathrm{order}(f) = \mathrm{order}(f^*)$, then $f = f^*$, and we have obtained the optimal feature.

Assume there exists a feature validator V that can evaluate the similarity between feature f and the optimal feature f^* by inspecting the downstream model score. This validator can assess the similarity with a certain probability and guide us during the optimization process.

In practical optimization, if we improve features by locally optimizing the syntax tree and use an LLM (Large Language Model) to infer from semantic knowledge, we can effectively expand the optimal feature’s subtree and increase its size. We limit the maximum edit distance during optimization and use the validator to evaluate the effectiveness of each improvement. Thus, the LLM can guide us to effectively grow the optimal feature.

In contrast, if we use a black-box evolutionary algorithm for optimization, when two features' syntax trees $T(f_1)$ and $T(f_2)$ are crossed over, the lack of effective LLM guidance often leads to ineffective improvements. In this case, erroneous changes may be fixed by the validator, resulting in poor optimization.

For example:

(imbalance_size - matched_size) / (matched_size + imbalance_size)

and

(bid_size- ask_size) / (ask_size + bid_size)

are both valuable features. However, randomly recombining their subtrees would result in a very messy structure, which not only fails to capture common patterns but also poses a high risk of overfitting.

In conclusion, by introducing syntax trees and syntax-guided optimization methods, feature generation and optimization can be more effective in higher-dimensional spaces. In contrast, black-box evolutionary algorithms may lead to unstable optimization results.