fn function_report_top_k

Michael Shoemate

April 12, 2025

1 Hoare Triple

Precondition

Compiler-verified

Types consistent with pseudocode.

Caller-verified

None

Pseudocode

```
def function_report_top_k(k: int, optimize: str) -> Callable[[list[TIA]], list[int]]:
    def function(x: list[TIA]) -> int:
        if optimize == "max":  #
            cmp = lambda a, b: a > b
        else:
            cmp = lambda a, b: a < b

        def max_sample(a, b):  #
            return a if cmp(a[1], b[1]) else b

        return [i for i, _ in top(x, k, max_sample)]   #

    return function
```

Postcondition

Theorem 1.1. Returns a noninteractive function with no side-effects that, when given a vector of scores x_i, returns the indices of the top k y_i, where each $y_i = -x_i$ if `optimize` is `min`, else $y_i = x_i$.

The returned function will only error if there are no non-null scores.

Proof. The comparator on line 3 effectively flips the sign of the scores if `optimize` is `min`, therefore each $y_i = -x_i$ if `optimize` is `min`, else $y_i = x_i$. This is done to avoid the negation of a signed integer.

Assume the scores are non-null, as required by the returned function precondition. Then `max_sample` on line 8 defines a total ordering on the scores.

Since the score vector is finite, and `max_sample` defines a total ordering, then the preconditions for `top` are met. Therefore on line 11 `top` returns the pairs with the top k scores. Line 11 then discards the scores, returning only the indices, which is the desired output.

There is one source of error in the function, when there are no non-null scores in the input vector. □