`ranx.fuse`: A Python Library for Metasearch

Elias Bassani[1,2] and Luca Romelli[2]

[1]Consorzio per il Trasferimento Tecnologico - C2T [2]University of Milano-Bicocca, Milan, Italy

What is Metasearch?

- **Metasearch**, sometimes called data-fusion, is the problem of **combining** the **results** returned by **multiple search engines** in response to a given query in a way that optimizes the performance of their combination.
- Previous works have shown this **combination** to consistently **improve** the retrieval **effectiveness** of the combined systems.
- Metasearch algorithms can be applied **externally**, when the combined search engines are completely independent from each other, or **internally**, when a single search engine comprises multiple retrieval models.

Metasearch Algorithm Classification

- **Score-based methods** combine the relevance scores given to the documents retrieved by multiple search engines to derive the final document scores.
- **Rank-based methods** rely only on the positioning of the documents retrieved by the considered search engines to derive the final ranking.
- **Probabilistic methods** derive a probability distribution of the relevance over the ranking positions. For every search engine, they assign to each ranking position the probability of finding a relevant document in that specific position.
- **Voting-based methods** adapt voting procedures, such as Borda Count and the Condorcet election method, to Metasearch, combining the preferences of multiple "experts", *i.e.*, the search engines.

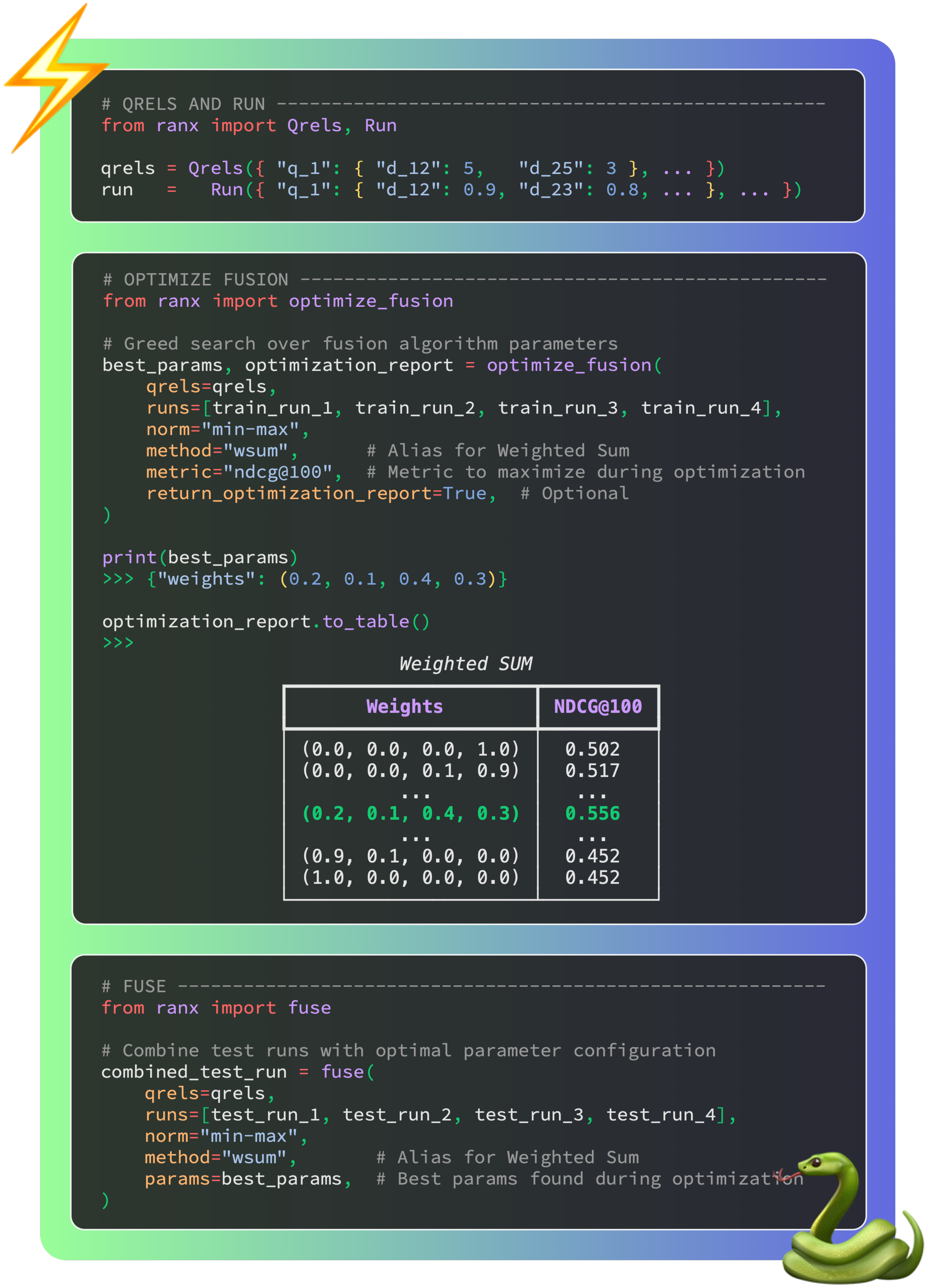
What is `ranx.fuse`?

- `ranx.fuse` is a collection of **Metasearch algorithms**, built on top of `ranx`'s `Numbda`-based data structures for high-speed vector operations and automatic parallelization.
- `ranx.fuse` embraces a *Plug & Play* philosophy, implementing a **user-friendly** interface to the provided Metasearch algorithms.

Main Features

- `ranx.fuse` provides **25 Metasearch algorithms** accessible through a common standardized interface, the `fuse` method.
- `ranx.fuse` implements **six normalization strategies** to transform the results of different search engines to make them comparable, which is mandatory for the correct application of many Metasearch algorithms.
- As many fusion algorithms require a training or optimization step, `ranx.fuse` implements the functionalities needed to optimize those algorithms, which are accessible through a single easy-to-use interface, the `optimize_fusion` method. In the case of the algorithms requiring hyper-parameters optimization, `ranx.fuse` comes with pre-defined hyper-parameters search spaces.

Overview



```
# QRELS AND RUN ---------------------------------------------------
from ranx import Qrels, Run

qrels = Qrels({ "q_1": { "d_12": 5,   "d_25": 3 }, ... })
run   =   Run({ "q_1": { "d_12": 0.9, "d_23": 0.8, ... }, ... })
```

```
# OPTIMIZE FUSION -------------------------------------------------
from ranx import optimize_fusion

# Greed search over fusion algorithm parameters
best_params, optimization_report = optimize_fusion(
    qrels=qrels,
    runs=[train_run_1, train_run_2, train_run_3, train_run_4],
    norm="min-max",
    method="wsum",       # Alias for Weighted Sum
    metric="ndcg@100",   # Metric to maximize during optimization
    return_optimization_report=True,  # Optional
)

print(best_params)
>>> {"weights": (0.2, 0.1, 0.4, 0.3)}

optimization_report.to_table()
>>>
                        Weighted SUM
          ┌─────────────────────────┬──────────┐
          │         Weights         │ NDCG@100 │
          ├─────────────────────────┼──────────┤
          │ (0.0, 0.0, 0.0, 1.0)    │  0.502   │
          │ (0.0, 0.0, 0.1, 0.9)    │  0.517   │
          │           ...           │   ...    │
          │ (0.2, 0.1, 0.4, 0.3)    │  0.556   │
          │           ...           │   ...    │
          │ (0.9, 0.1, 0.0, 0.0)    │  0.452   │
          │ (1.0, 0.0, 0.0, 0.0)    │  0.452   │
          └─────────────────────────┴──────────┘
```

```
# FUSE ------------------------------------------------------------
from ranx import fuse

# Combine test runs with optimal parameter configuration
combined_test_run = fuse(
    qrels=qrels,
    runs=[test_run_1, test_run_2, test_run_3, test_run_4],
    norm="min-max",
    method="wsum",       # Alias for Weighted Sum
    params=best_params,  # Best params found during optimization
)
```

Provided Metasearch Algorithms

Table 1. Supervised means the algorithm requires a training phase. Params column indicates whether the algorithm has parameters that need to be optimized. TT and PF columns indicate whether the algorithm is provided by `TrecTools` or `Polyfuse`, respectively.

Score-based Methods

Name	Supervised	Params	TT	PF
CombANZ	✗	✗	✓	✓
CombMAX	✗	✗	✓	✓
CombMED	✗	✗	✓	✓
CombMIN	✗	✗	✓	✓
CombMNZ	✗	✗	✓	✓
CombSUM	✗	✗	✓	✓
CombGMNZ	✗	✓	✗	✗
Mixed	✗	✓	✗	✗
WMNZ	✗	✓	✗	✗
Weighted Sum	✗	✓	✗	✗

Rank-based Methods

Name	Supervised	Params	TT	PF
ISR	✗	✗	✗	✓
Log_ISR	✗	✗	✗	✓
LogN_ISR	✗	✓	✗	✗
RBC	✗	✓	✓	✓
RRF	✗	✓	✓	✓

Probabilistic Methods

Name	Supervised	Params	TT	PF
BayesFuse	✓	✗	✗	✗
MAPFuse	✓	✗	✗	✗
PosFuse	✓	✗	✗	✗
ProbFuse	✓	✓	✗	✗
SegFuse	✓	✗	✗	✗
SlideFuse	✓	✓	✗	✗

Voting-based Methods

Name	Supervised	Params	TT	PF
BordaFuse	✗	✗	✓	✓
Weighted BordaFuse	✗	✓	✗	✗
Condorcet	✗	✗	✗	✗
Weighted Condorcet	✗	✓	✗	✗

Online Resources

- Lean more about `ranx.fuse` at `https://amenra.github.io/ranx/` (or scan the QR Code below).
- Would you like to see other features implemented? Feel free to open a feature request on our repository: `https://github.com/AmenRa/ranx`.

elias.bssn@gmail.com