Axiomatic Retrieval Experimentation with \texttt{ir\_axioms}

### Axiomatic Thinking

Successful retrieval scoring functions share similar properties:

\[
BM25(q, d) = \sum_{t} \frac{IDF(t) \cdot TF(t, d)}{TF(t, d) + k_1 \cdot (1 - b \cdot \frac{L}{avdl} + b \cdot \frac{TF(t, d)}{IDF(t)})}
\]

Axioms formally capture such properties.

- **TFCl**: prefer documents with more query terms \cite{fang2004}
- **LNC1**: penalize non-query terms in longer doc's \cite{fang2022}

### Axiom applications

- Improving an initial retrieval result via re-ranking \cite{hagen2016}
- Using axioms as regularization loss in neural models \cite{rossat2019}
- Learning how to combine retrieval models \cite{arora2019}
- Analyzing / explaining neural rankings \cite{volske2021}

### The \texttt{ir\_axioms} Framework

- Python framework for experiments with IR axioms
- Implements 25 axioms (parameterizable preconditions, multi-term queries, etc.)
- Access to retrieval models and test collections in PyTerrier and \texttt{ir\_datasets}
- Caching and parallelization

### Examples

#### Implemented axioms:

<table>
<thead>
<tr>
<th>Objective</th>
<th>Axioms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Term frequency</td>
<td>TFCl, TF3, TDC, M-TDC</td>
</tr>
<tr>
<td>Document length</td>
<td>LNC1, TF-LNC</td>
</tr>
<tr>
<td>Lower-bound TF</td>
<td>LB1</td>
</tr>
<tr>
<td>Query aspects</td>
<td>REG, AND, DIV</td>
</tr>
<tr>
<td>Semantic similarity</td>
<td>STMC1, STMC2</td>
</tr>
<tr>
<td>Term proximity</td>
<td>PROX1–PROX5</td>
</tr>
<tr>
<td>Argumentativeness</td>
<td>ArgUC, QTArg, TQ9pkg, sSSLDoc</td>
</tr>
<tr>
<td>Other</td>
<td>ORIG, ORACLE</td>
</tr>
</tbody>
</table>

#### Implementing the TFCl axiom:

```python
class TFCl(Axiom):
    name = "TFCl"
    def preference(self, c, q, d_i, d_j):
        # Length precondition.
        if not approx_same_length(c, d_i, d_j, 0.1):
            return 0

        # Count query terms.
        tf_i = sum(c.term_frequency(d_i, t) for t in c.terms(q))
        tf_j = sum(c.term_frequency(d_j, t) for t in c.terms(q))
        if approx_equal(tf_i, tf_j, 0.1):
            return 1 if tf_i > tf_j else -1
```

#### Post-hoc Analysis

\[
\text{bm25} = \text{BatchRetrieve}(\text{index, } \"BM25\")
\]

#### Axiomatic Re-ranking

```
# Re-ranking top-20 BM25 results.
kwiksort = bm25 % 20 >> KwikSortReranker( # KWIKsort with axiomatic features from top-10.
    [ArgUC(), QTArg(), QTPArg()],
    [mean, median])
```

### Resources and Installation

Contributions and feedback are welcome!

- \url{webis-de/ir\_axioms}
- \texttt{pip install ir\_axioms}
- \texttt{10.1145/3477495.3531743}

Thanks to the SIGIR for a student travel grant.

This work has been partially supported by the DFG through the project "ACQuA 2.0: Answering Comparative Questions with Arguments" (project number 376430233) as part of the priority program "RATIO: Robust Argumentation Machines" (SPP 1999).