

Implementing State of the Art Ranking for Lucene

Implementation Plan

David Nemeskey
Eötvös Loránd University
`nemeskey.david@sztaki.hu`

1 Basic Decisions

Fully pluggable ranking requires loose coupling between the ranking component and other parts of the query class hierarchy. All document score-related computations must happen in the ranking component. Implementations of ranking algorithms should adhere to a well-defined interface. In the current API, this is the role of the `Similarity` class; therefore, to minimize unnecessary changes, it shall serve as the base for the new ranking framework as well.

2 Architecture changes

In line with the decision above, `Similarity` will not only expose term weight factors in its API, but it will also be responsible to combine them in the new `score()` method. Methods in other classes that are strongly tied to the ranking algorithm, such as `Weight.sumOfSquaredWeights()` will be factored out and moved into `Similarity` as well.

For performance reasons, Lucene stores the VSM document length norm in the index. Since every ranking algorithm uses a different norm, it is impossible to follow this solution in the new framework. Hence, the new ranking methods will compute the norms on the fly.

3 Ranking implementation

While the proposal established the need for a class hierarchy for ranking functions, a flexible implementation requires more. Most scoring methods consist of two parts: one describes the term distribution in the document (*tf*) and another in the corpus (*idf*). Variations on the two parts are not uncommon; DFR and language models, for instance, feature a wide range of smoothing functions. Therefore, two interfaces, `Frequency` and `Smoothing`¹, will be introduced for classes that realize different *tf* and *idf* weighting schemes. `Similarity` will delegate *tf* and *idf* computation to implementations of these interfaces.

4 Configuration

The ranking function can be selected by instantiating the appropriate `Similarity` object and passing it to the `Scorers` and `Searchers`, or by setting it as the system default. The `Similarity` will accept a dictionary of properties, which can be used to specify the parameters of the ranking algorithm. The parameter names and their roles will match those found in the literature.

¹Working names.