LucidWorks™

Relevancy Workbench Module

1.0 Documentation

Created: 10-Sep-2013

Table of Contents

Relevancy Workbench Module

The Relevancy Workbench module allows modeling changes to search defaults and how the changes will effect how results are ordered for users.

Using a convenient side-by-side comparison model, the module allows tweaking several parameters and judging results "before" and "after" the changes are implemented.

The module does not save changes to Solr, but allows you to run experiments with different approaches and easily view how results change. The module gives you a way to run experiments with parameters.

This documentation covers:

- How to install the Relevancy Workbench module and use it with your existing Solr implementation
- How to use the Relevancy Workbench module UI to tweak results and interpret changes
- General information on Running Relevancy Test Sessions to help you plan your efforts

When just starting out, keep a few tips in mind:

- Understand the relevancy problem you are trying to solve. If a user's favorite query isn't ranking results in the expected way, is that indicative of a broader problem? If so, how?
- Start with modifying one parameter at a time so you can understand the impact of each possible change.
- Don't pay too close attention to how documents score; that's a relative measure, and Lucene/Solr scoring includes factors that normalize scoring for a particular result set. The scoring information provides a window into why a document scored the way it did, and why it scored higher or lower than another document, but does not mean much more than that.
- Plan to test the potential changes with multiple sample queries that reflect how your users search your application.

The free download includes a 60-day license for use of the module. After the 60-day trial period, the software will stop working. If you would like to purchase this module, please contact sales@lucidworks.com for pricing and further details.

Installing the Relevancy Workbench Module

The Relevancy Workbench module can be run as a standalone web application or deployed to a web container. If it is run as a standalone application, it is started and stopped separate from Solr, and runs on its own Jetty port.

System Requirements

Solr version 4.0 or higher is required.

JavaScript must be enabled in the browser for the module to function properly. We have tested the module on modern versions of Internet Explorer, Firefox, Safari, and Chrome.

Additionally, some understanding of Solr's search syntax is helpful, particularly the parameters that impact boosting. If you are new to Solr, the Searching section of the Solr Reference Guide is a good place to start.

Standalone Relevancy Workbench

If you are running as a standalone application, you can locate the `relevancy.jar` file anywhere on your filesystem.

To start, you only need to run the `relevancy.jar` file:

```
java -jar relevancy.jar
```

Then you can connect to the application by accessing it through a browser on the default port of 8080. For example, once started, you can access the application at `http://localhost:8080/`. To change the default port, start the application with the `-Djetty.port` system property:

```
java --Djetty.port=9090 jar relevancy.jar
```

By default, the application will try to connect to a Solr server at `localhost:8983/solr`. To change this, start the application with the `-Dsolr.url` property and define the proper URL for Solr:

```
java -Dsolr.url=http://10.0.0.1:8888/solr -jar relevancy.jar
```

The Relevancy Workbench module uses real data in your Solr indexes to show you the impact of changes, so if Solr is not running or is not accessible, you will not be able to test results.

Deploy to a Web Container

If you would prefer to run the Relevancy Workbench module in your existing web container, you can move the `relevancy.war` file to your container's web applications directory and declare a web context for it. Consult your web container documentation if you are not sure how to set up a context.

The Relevancy Workbench application can run under a named context path (`/relevancy` for instance) or the `/` root context path.

By default, the application will try to find Solr at `http://localhost:8983/solr`. This can be overridden by setting the `solr.url` system property.

Relevancy Workbench Parameters

The Relevancy Workbench allows administrators to model adjustments to overall search behavior by showing the effect of changes to search defaults and how results are ordered for users. You can use the screen to run two queries with different parameters and evaluate the results side-by-side. Parameters for queries can be independently adjusted and will change the results shown in the result comparison window at the bottom of the page.

This tool is used for modeling changes. It does not currently support modifying the real search behavior of your application. Once desired adjustments are determined, your search application will need to be modified to make the changes live for users.

Relevancy ranking of results is a complex topic, and whether or not a document is relevant to a query is nearly always subjective. Changes that may make the results for one query seem "better" may have a negative impact on other queries. This tool is designed to help you make informed decisions before making site-wide changes.

If you are looking to force specific records to the top of the results, you may want to consider using the Query Elevation Component.

Topics discussed in this section:

Using the Relevancy Workbench Screen

The page is arranged into several sections. The first section allows you to define two general parameters that will apply to both of the queries that will be run. These include the query parser and the query to use.

Once the general parameters have been defined, the screen is split to define further parameters that will differ between the two queries. The left side defines the "base" parameters, which are compared with the right panel. Changes can be made in the left panel, but they will be used for comparison with the selections in the right. If no parameters are entered in either side, system defaults are used, depending on the query parser selected, and both left and right result displays will be equivalent.

The overall workflow for using the Relevancy Workbench is:

- Enter a query term that will retrieve documents.
- Change settings on left, right, or both, and click "Compare".
- Review the differences between the left and right results.
- Enter other search terms so the system does not become optimized for a single query.
- Adjust settings as needed.
- Update system configuration or search application code as needed so future queries are run with the new parameters.

Relevancy Workbench Parameters

As mentioned, the screen is divided into sections: the first for General Parameters, then left and right panels to enter parameters for comparing results. The left and right sections are further divided into Basic and Advanced options. These are parameters that are commonly used to control search results, but have been grouped in this way to provide usability. There is an option in the Advanced parameters to add additional parameters not directly exposed with this UI. You can toggle the Advanced parameters on and off by clicking the "Advanced" button at the top of the page, near the "Relevancy Workbench" title.

At any time, you can change a parameter so the same entries apply to both searches separately by clicking the arrows on the right side of the window. Clicking again will cause the entry for the parameter to apply to both queries. For example, by default, the Query Parser parameter is shown as applying to both searches. By clicking the arrows on the right side, you will see two Query Parser pull-down menus, which would allow you to compare the same query across two parsers.

When looking at the various parameters, note the abbreviations in parentheses next to each label. These abbreviations show you the parameter syntax to use when modifying your search application once you have decided on the best changes for your users.

When all the parameters have been entered, click **Compare** to see results. The section Viewing Results has details on how to read the results panels. The Troubleshooting section contains further information for using the tool.

General Parameters

The general parameters apply to all searches.

Query Text (q)

Sets the query to use to find documents. Any Solr query that can be specified with the 'q' parameter in a Solr request can be used here.

Query Parser (defType)

Sets the query parser to use. The query parser selected may impact how the other parameters are interpreted.

The pull-down menu lists the current query parsers for Solr (dismax and edismax at least). Any additional parsers listed in `solrconfig.xml` with the `queryParser` attribute will also be listed in the pull-down menu. If your query parser is not listed and you would like to use it, you can define it in the "Other Parameters" field (one of the advanced options) by entering `defType=myparser` (replacing "myparser" with the name of the query parser to use).

For more information about query parsers in Solr, see the Solr Reference Guide section Query Syntax and Parsing.

Basic Parameters

The basic parameters include the most commonly changed search parameters. Even though they are common, they may still have a big impact on results. Access to these parameters is always available, even when the Advanced parameters toggle is on.

Default Field (df)

This parameter defines the default field to search. If none is specified, the default for the query handler will be used.

Query Fields and Boosts (qf)

Defines boost factors to fields when the query matches terms in those fields. For example, if `title^5 author` is entered, documents with the query term in those fields will be boosted. Because we've defined a boost factor (in the form of `^5`) to the title, a document with the query term in the title will be boosted by a factor of 5 over a document where the same term appears in the author field of a document. Multiple boosts can be defined; for example, `title^5 content^2 comments^0.5`.

When using this parameter, note that it will override parameters entered in the Default Field parameter.

More information is in the Solr Reference Guide section on qf.

Filter Query (fq)

Filters the results of the query according to the rules defined here. This essentially adds to the user's query without impacting the score of the documents. Common uses for a filter query are to restrict documents to a date range, or by user popularity, or another factor that improves the user's search experience.

More information is available in the Solr Reference Guide section on fq.

Advanced Parameters

The advanced panel contains more settings that may be useful, but these are considered more complex settings require some familiarity with Lucene and Solr query syntax. You can toggle the Advanced parameters on and off by clicking the "Advanced" button at the top of the page, near the "Relevancy Workbench" title.

Default operator (q.op)

The default operator defines how multiple term queries (that are not phrases) are interpreted. This would apply if no operator was specified. The options are to take the default for the selected query parser, to specify it explicitly as AND, or specify it explicitly as OR. The AND operator inserts an AND between each term, while OR would insert an OR.

Phrase Fields (pf)

When multiple terms are defined by the user as part of a query, this parameter provides a boost when the user's query terms are found in the specified field. For example, if the query is 'mortgage rate', and Phrase Fields is 'title', documents with the terms 'mortgage' or 'rate' in the title will be given an additional boost. The format is the same as the Query Fields parameter, so can be a simple field name, a single field with a boost value, or multiple fields with different boost values (for example, `title^5 content^2 comments^0.5`).

This parameter can be used even when the user does not explicitly define a phrase, but when any multiple term query is entered.

More information is available in the Solr Reference Guide section on pf.

Phrase Slop (ps)

Used with the Phrase Fields parameter, this parameter defines a maximum distance between terms in the defined query. This allows documents to be found and boosted even when the exact terms do not appear in the field defined with the Phrase Fields parameter. Using a query of "mortgage rate" again as an example, a Phrase Slop of 4 would mean that documents that include the terms "mortgage" and "rate" within 4 words of each other in the title (assuming "title" was the phrase defined in the Phrase Fields parameter) would get a boost.

More information is available in the Solr Reference Guide section on ps.

Query Slop (qs)

This allows loosening a user's phrase query (defined as query terms surrounded by quote marks) so the terms do not need to be next to each other in order to match the user's query. Phrases entered by the user will be queried as terms which must be near each other by the specified number of tokens. For example, using "mortgage rate" again, this time entered as a phrase with quotes, a Query Slop of 3 gives a boost to documents where 'mortgage' and 'rate' appear within 3 terms of each other.

More information is available in the Solr Reference Guide section on qs.

Tie Breaker (tie)

This allows defining how to handle clauses that match in more than one field. The value must always be less than 1 (one), and recommended to be much less than 1.

When a term from the user's input is tested against multiple fields, more than one field may match. If so, each field will generate a different score based on how common that word is in that field (for each document relative to all other documents). The tie parameter lets you control how much the final score of the query will be influenced by the scores of the lower scoring fields compared to the highest scoring field.

When the Tie Breaker value is increased to 1.0, the final score becomes a sum of all of the sub-scores instead of the maximum score plus a fraction (i.e., the Tie Breaker value when less than 1.0) of the sum of the other scores.

More information is available in the Solr Reference Guide section on tie.

Boost (boost)

This allows specifying a function that will be multiplied into the score. It is multiplicative, meaning the resulting score of the boost calculation is multiplied into the score. A typical use of this may be to boost more recent documents, or documents representing products that have sold more units in the past 30 days. This parameter can only be used with the 'edismax' query parser and will not appear if another query parser is chosen.

While the default behavior is to parse the input as a function, it is also possible to use local parameters to use any sub-query parser that you want (such as a field query, or a spatial query). This means the scores of the sub-query parser would be multiplied into the overall score. For example, an e-commerce site may want to boost products which have a user rating of 5. One approach could be to enter `{!field f=userRating}5`, which would boost documents with a "userRating" of 5 higher.

This parameter does not guarantee the results will appear at the top, but adds to the score of the document, increasing the document's chance to appear at the top of the results.

It differs from a boost query ('bq', below) because the resulting boost is multiplied into the score instead of added. This allows much higher boosting capability, if required.

More information is available in the Solr Reference Guide section on boost.

Boost Query (bq)

This defines a query string that will be used to boost documents. For example, a boost query could be used to boost more recent documents, or documents that are sponsored (if that information is in a field of the document). Queries must be expressed in Solr query syntax, so this would be a valid entry: `bq=sponsored:true`. Just as with the Boost parameter, you can also use any sub-query parser. If a sub-query parser is not defined, the default is the Lucene parser.

This does not guarantee the results will appear at the top, but adds to the score of the document, increasing the document's chance to appear at the top of the results.

More information is available in the Solr Reference Guide section on bq.

Minimum Match (mm)

This allows defining a minimum number of clauses entered by the user that must match documents. The match number can be entered as a positive number (1, 2, etc.), a negative number, a percentage, or as a conditional statement of a number and/or percentage. Minimum match allows some very flexible combinations, such as:

- Positive number: at least this number of optional clauses must match, regardless of how many clauses there are (such as "3").
- Negative number: the total number of optional clauses, minus this number, must match (such as "-1").
- Percentage: this percent of optional clauses must match (such as "75%). Percentages are calculated by rounding down.
- Negative percentage: this percent of clauses can be missing (such as "-25%"). Percentages are calculated by rounding down.
- Conditional relationships:
    - Positive integer less than percent: if the number of clauses is equal to or less than (or greater than, depending on the symbol) the integer, they are all required. If the number of clauses is greater than the integer, the percentage applies (such as "3<80%").
    - Positive integer less than negative percent: if the number of clauses is equal to or less than (or greater than, depending on the symbol) the integer, they are all required. If the number of clauses is greater than the integer, the defined percentage of clauses can be missing (such as "2<-25%").
    - Multiple conditions: if the number of clauses is equal to or less than (or greater than, depending on the symbol) the first integer, they are all required. If the number of clauses is greater than the second integer, the second condition is applied. Values between the first and second integers use the first percentage (such as "2<-25% 9<-3").

The default minimum match is determined by the "q.op" parameter. If "q.op" is AND, then minimum match defaults to 100%. If "q.op" is OR, then minimum match defaults to 0%.

More information is available in the Solr Reference Guide section on mm.

Other Parameters

Any other valid parameters are acceptable here, as long as they are in the proper syntax and formatted as a URL-escaped sequence of key=value pairs. For example, you may wish to show the results starting at a item 50 in the full result list and display the next 30 results. In that case, you could add `start=50&rows=30` in the Other Parameters field, and your results shown will be items 50-79 in the list.

Viewing Results

The results are displayed under the parameters section, on the left and the right, corresponding to the parameters entered. You can change the number of results shown by choosing 25, 50, 100 from the pulldown on the right.

Clicking on the title of a document opens a window that displays each field of the document. Click "Scoring" in this window, and you will see how documents were scored.

The differences between the results are highlighted in this manner:

- If a document is shown on the left, but parameters on the right prevent it from being shown, the document will be shown with a red background in the left panel. This is limited by the number of documents shown, so if a document disappears, it may appear with a red background on the left, but may really have moved so far down the list it is no longer shown. If you change the number of results (rows) to show, you may find the record in the right.

- If a document is not shown on the left, but parameters on the right allow it to be shown, the document will be shown with a green background in the right panel. Again, this is limited to the number of documents shown.

- If a document has moved places in the order of results in the right panel, an up or down arrow will be shown with a number to indicate the number of places it has moved either up or down.

- If a document has not moved places at all, it will be shown with no decoration. However, these documents may have different scores (if scores are shown), since scoring is always relative to the other documents in the results list.

Troubleshooting

Same Results on Both Sides

This is most likely to occur when the default ranking already puts results in a reasonable order, and the changes you want to make don't really make much of an impact. An example of this would be trying to boost title when there are only a few documents with the search term in the title, and they are already at or near the top of the list.

In order to verify that the changes are being calculated, however, enable the option to show the "Explain scores". The scores between the two lists will likely be different.

No Documents are Shown

If no documents are shown, or all documents are red or all documents are green, there can be a few possible causes:

- There is something in the Default Field parameter that doesn't match any documents in either the left or the right panels.
- An entry in the Query Fields and Boosts parameter has overridden an entry in the Default Fields parameter.
- The fields in the documents may not match the fields being defined in the query.

Many Documents are Red or Green

If there are a lot of red or green documents, the cause may be similar to those described in the section No Documents are Shown. In addition, you may need to increase the Number of Results shown in the General Parameters section at the top of the page. It's also possible that your changes may not have as great an impact as you thought. In that case, you'll want to test several queries to be sure you understand how your changes will effect most users.

Not Many Documents are Red or Green

This may indicate that your changes do not have a big impact on the results. If so, you'll want to test several queries to be sure you understand how your changes will effect most users.

A Specific Document is Not Shown First

If all you want to do is make sure a specific document always returns to the top for a certain query, you may want to consider using the QueryElevationComponent instead. This feature allows elevating or blacklisting specific documents in response to a user's query. Details on how to use it are available in the Solr Reference Guide section on the QueryElevationComponent.

Another option is to take a look at the Business Rules module for Solr. More information on that module can be found in the Business Rules Documentation.

Running Relevancy Test Sessions

Relevancy is a complex topic, and frequently two users performing the same query have different opinions about which documents best match the query. In the end, judging relevance has an inherent subjectivity to it. However, there are some ways to assess relevance and adjust how documents are scored to improve ranking.

For this reason, relevancy should also be judged in the context of the queries users usually submit and the types of documents in the index. In addition, taking a systematic approach to testing for possible problems prevents solving relevancy for one query but breaking it for another.

Here are some approaches to testing you might want to consider:

- **Empirical Tests**: Extract the 50 or so queries from logs, plus an additional 10 or more random queries, and ask 1-3 users to run the queries and rank the results. Repeating this test as changes are made will show the effect changes are having on the overall system.
- **A/B Tests**: Show two groups of users different sets of results and evaluate user clicks and choices to determine who had the "better" set of results.
- **User Ranking**: Allow users to rate documents using a star or number system.
- **Focus Groups**: Assemble a focus group representing your users and allow them to interact with the system for a period of time, recording their behaviors and feedback.
- **TREC Evaluations**: Run a relevance study using queries, documents, and judgements created by a third-party group, such as the Text Retrieval Evaluation Conference (TREC) or similar.

There are more options, too. Whichever approach is taken, however, the goal is to be systematic in evaluating relevancy to eliminate individual subjectivity as much as possible. For more reading on relevance test sessions and improving relevance in a search application, you may be interested in these articles:

- Relevance chapter from the Apache Solr Reference Guide
- Debugging Search Application Relevance Issues, by Grant Ingersoll, hosted on SearchHub.org
- Options to Tune Documents' Relevance, by Tomàs Fernàndez Löbbe, hosted on SearchHub.org