Item 66
**git_comments:**

1. * * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
2. * * Expert: Factory to provide a {@link SolrSimilarityProvider}. * <p> * Usually you would implement this if you want to customize the * scoring routines that are not field-specific, such as coord() and queryNorm(). * Most scoring customization happens in the fieldtype's Similarity
3. * * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
4. * * SimilarityProvider that uses the default Lucene implementation, unless * otherwise specified by the fieldtype. * <p> * You can extend this class to customize the behavior of the parts * of lucene's ranking system that are not field-specific, such as * {@link #coord(int, int)} and {@link #queryNorm(float)}.
5. * * Solr implementation delegates to the fieldtype's similarity. * If this does not exist, uses the schema's default similarity.
6. note: this is intentionally final, to maintain consistency with whatever is specified in the the schema!
7. * * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
8. * * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
9. * test a field where a configurable sim factory is defined
10. * * Tests per-field similarity support in the schema
11. * test a field where no similarity is specified
12. * ... and for a dynamic field
13. * * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
14. * test a field where the sim is specified directly
15. * test a field that does not exist

16. * * NOTE: this runs in linear time (it scans starting at the * beginning of the list until it finds the first pair with * the specified name).
17. * * Gets the Similarity used when scoring fields of this type * *  * The default implementation returns null, which means this type * has no custom similarity associated with it. *  * * This method exists to internally support SolrSimilarityProvider. * Custom application code interested in a field's Similarity should * instead query via the searcher's SimilarityProvider. * @lucene.internal
18. * @lucene.internal
19. * * Sets the Similarity used when scoring fields of this type * @lucene.internal
20. just like always, assume it's a SimilarityProviderFactory and get a ClassCastException - reasonable error handling configure a factory, get a similarity back
21. * fallback similarity, in the case a field doesnt specify
22. **comment:** just like always, assume it's a Similarity and get a ClassCastException - reasonable error handling
    **label:** code-design
23. a custom similarity[Factory]
24. some per-field similarity examples
25. don't specify any sim at all: get the default
26. specify a Similarity factory
27. make sure custom sims work with dynamic fields
28. specify a Similarity classname directly
29. expert: SimilarityProvider contains scoring routines that are not field-specific, such as coord() and queryNorm(). most scoring customization happens in the fieldtype. A custom similarity provider may be specified here, but the default is fine <similarityProvider class="org.apache.solr.schema.CustomSimilarityProviderFactory"> </similarityProvider> <!-- default similarity, unless otherwise specified by the fieldType

**git_commits:**

1. **summary:** SOLR-2338: improved per-field similarity integration into schema.xml
   **message:** SOLR-2338: improved per-field similarity integration into schema.xml git-svn-id: https://svn.apache.org/repos/asf/lucene/dev/trunk@1087430 13f79535-47bb-0310-9956-ffa450edef68

**github_issues:**

**github_issues_comments:**

**github_pulls:**

**github_pulls_comments:**

**github_pulls_reviews:**

**jira_issues:**

1. **summary:** improved per-field similarity integration into schema.xml
   **description:** Currently since LUCENE-2236, we can enable Similarity per-field, but in schema.xml there is only a 'global' factory for the SimilarityProvider. In my opinion this is too low-level because to customize Similarity on a per-field basis, you have to set your own CustomSimilarityProvider with <similarity class=.../> and manage the per-field mapping yourself in java code. Instead I think it would be better if you just specify the Similarity in the FieldType, like after <analyzer>. As far as the example, one idea from LUCENE-1360 was to make a "short_text" or "metadata_text" used by the various metadata fields in the example that has better norm quantization for its shortness...

**jira_issues_comments:**

1. Yep, sounds like a great idea! Should we specify the similarity class in each fieldType that want's to use a non-default similarity: {code} <fieldType> <analyzer>...</analyzer> <similarity class=...></similarity> </fieldType> {code} Or use named similarities and refer to them: {code} <fieldType> <analyzer>... </analyzer> <similarity name="short_text"/> </fieldType> <similarity name="short_text" class=...> </similarity> {code}

2. **body:** doesn't matter to me really, but what is the advantage of the named similarities? this would be a bit inconsistent from how you configure analyzers (and an additional level of indirection that might be confusing)... or am I missing something?
   **label:** code-design
3. Other components in solrconfig use that indirection, but I'm fine w/ the approach taken by tokenizer / token filter config.
4. **body:** Most existing situations where plugins are dereferenced by name are so we can reuse the exact same object instance (ie: for recording stats, or because they are heavyweight to construct on the fly) in the case of similarity, the main advantage i can think of would be if we wanted true per-field similiarity declaration, not just per field type ie... {code} <similarity name="S_XX" class=...></similarity> <similarity name="S_YY" class=...></similarity> ... <fieldType name="FT_AA"> <analyzer>... </analyzer> <similarity name="S_XX"/> </fieldType> ... <field name="F_111" type="FT_AA" /><!-- implied S_XX --> <field name="F_222" type="FT_AA" similarity="S_YY" /> {code} ...but even if we don't do that, i suppose it's also conceivable that someone might have their own Similarity implementation that is expensive to instantiate (ie: maintains some big in memory data structures?) and might want to be able to declare one instance and then refer to it by name in many different fieldType declarations. I think for now just supporting the first example yonik cited... {code} <fieldType> <analyzer>...</analyzer> <similarity class=...></similarity> </fieldType> {code} would be a huge win, and we can always enhance to add name derefrencing later.
   **label:** code-design
5. **body:** {quote} ...but even if we don't do that, i suppose it's also conceivable that someone might have their own Similarity implementation that is expensive to instantiate (ie: maintains some big in memory data structures?) and might want to be able to declare one instance and then refer to it by name in many different fieldType declarations. {quote} I don't think this is really a use case we need to support: the purpose of Similarity today is to do term weighting, not to be a huge data-structure holder. While I know Mike's original patch went this way with LUCENE-2392 (e.g. norms), I'm not sure i like it being in Similarity in the future either. Otherwise concepts like lazy-loading norms and all this other stuff get pushed onto the sim, which is an awkward place (imagine if you have many fields). So, I think we shouldn't really design for abuses of the API. If there are other use cases for "named similarity" that have to do with term weighting, I'm interested.
   **label:** code-design
6. **body:** Here's a first stab: I included LUCENE-2986's cleanup work for easy testing (this issue depends upon it). Here is the syntax: {noformat} <!-- specify a Similarity classname directly --> <fieldType name="sim1" class="solr.TextField"> <analyzer> <tokenizer class="solr.WhitespaceTokenizerFactory"/> </analyzer> <similarity class="org.apache.lucene.misc.SweetSpotSimilarity"/> </fieldType> <!-- specify a Similarity factory --> <fieldType name="sim2" class="solr.TextField"> <analyzer> <tokenizer class="solr.WhitespaceTokenizerFactory"/> </analyzer> <similarity class="org.apache.solr.schema.CustomSimilarityFactory"> <str name="echo">is there an echo?</str> </similarity> </fieldType> {noformat} Additionally, its necessary to allow customization of the SimilarityProvider too, in order to customize the non-field specific stuff like coord()... this is done via: {noformat} <!-- expert: SimilarityProvider contains scoring routines that are not field-specific, such as coord() and queryNorm(). most scoring customization happens in the fieldtype. A custom similarity provider may be specified here, but the default is fine for most applications. --> <similarityProvider class="org.apache.solr.schema.CustomSimilarityProviderFactory"> <str name="echo">is there an echo? </str> </similarityProvider> {noformat}
   **label:** code-design
7. We resolved LUCENE-2986. Here's the same patch, without those changes for convenience. Would appreciate anyones opinions on this.. its easier to review now I think.
8. I'll commit this in a few days unless anyone objects.
9. **body:** skimming the patch, the one thing i'm not clear on is what happens when someone who has been using a custom similarity (or similarity factory) in Solr 1.4 or 3.1 will be affected on upgrading. the patch seems to remove the code that allows for a (global) {{<similarity/>}} element in schema.xml (replacing it with a check for {{<similarityProvider/>}} i'm not clear on whether there is really a compelling reason for this (if there is we should have a nice fat warning in the upgrading section of CHANGES.txt) or if we could still continue to respect the {{<similarity/>}} syntax ... it seems like that one tag could be used to refrence (by classname) a global SimilarityProviderFactory, or a default Similarity instance (or default SimilarityFactory instance). even if there's a really good reason not to keep using what we might find in {{<similarity/>}}, we should check for it and log a nice fat error message saying it's being ignored.
   **label:** code-design

10. **body:** Hoss, thanks for looking at the patch. As far as the current form, Similarity is purely per-field, dictated by the schema (like Analyzers). For simplicity i removed the global one completely. If we want, the following options are available: * (global) <similarity/> configures the "default", which is used unless overridden by the fieldtype. * (global) <similarity/> triggers an error. The only reason I yanked it in the patch, was that I felt it could be confusing to have this "inheritance" so to speak... but if you think its not confusing, we could support the global <similarity/>, which would always be used unless you supply a different <similarity/> for a specific field.
    **label:** code-design
11. **body:** bq. but if you think its not confusing, we could support the global <similarity/>, which would always be used unless you supply a different <similarity/> for a specific field. i don't personally think it would be confusing, but i also don't think we need to advertise it in the example. we should definitely encourage using similarity per field type, but for people who have used it in the past, having it continue to work as a "global default" when fieldTypes don't define a similarity gives us nice back-compatibility. More generally though, i'm thinking that the same {{<similarity/>}} tag can be used for both the old style (global default) Similarity/SimilarityFactory and the new SimilarityProviderFactory using instanceof checks... * instanceOf SimilarityProviderFactory ** instantiate it and use it. * instanceOf SimilarityFactory ** instantiate and wrap it in a SolrSimilarityProvider that delegates to it when the field type has no similarity set on it. return an anonymous SimilarityProviderFactory that uses this SolrSimilarityProvider * instanceOf Similarity ** instantiate and wrap it in a SolrSimilarityProvider that delegates to it when the field type has no similarity set on it. return an anonymous SimilarityProviderFactory that uses this SolrSimilarityProvider ...that way there is only one "global" option that can be specified, and we don't have to deal with weird edge cases of what the default should be for a fieldTYpe w/o a similarity if the schema.xml specifies both {{<similarity/>}} and {{<similarityProvider/>}} The one other thing i just noticed is that you have SimilarityProviderFactory.init(SolrParams) ... my vote would be to start using NamedList based initialization for all new types of solr plugins. it requires more verbosity in the config, but it supports a lot more types of information (multivalued keys, nested lists/maps, etc...) and could eventually lead us to actually being able to validate our config files using an XMLSchema and/or DTD (since the element/node names are finite)
    **label:** code-design
12. **body:** {quote} i don't personally think it would be confusing, but i also don't think we need to advertise it in the example. we should definitely encourage using similarity per field type, but for people who have used it in the past, having it continue to work as a "global default" when fieldTypes don't define a similarity gives us nice back-compatibility. {quote} I agree here, this is a good compromise and by not advertising it in the example, I won't have concerns about the example being confusing. {quote} More generally though, i'm thinking that the same <similarity/> tag can be used for both the old style (global default) Similarity/SimilarityFactory and the new SimilarityProviderFactory using instanceof checks... {quote} I have to disagree on this one. The new SimilarityProvider serves a totally different purpose, its not a global sim: it answers to requests for sims for specific fields. The only reason I provided a factory for it, is so that users can tune the parts of lucene's relevance ranking system that are not per-field: coord() and queryNorm(). But its not a way to configure tf() or idf() or anything like that. In the patch I added this with "expert" to the example, though we could remove it from the example entirely if its too expert (might be?) So I think we should do as you suggest and allow a global <similarity/> that is the default term weighting unless otherwise specified by a field, but we shouldn't confuse this with the parts that arent field-specific... {quote} The one other thing i just noticed is that you have SimilarityProviderFactory.init(SolrParams) {quote} I configured it this way, because this is how <similarity/> worked before (and it was just enough XML to not scare me away). Is it possible we can defer this improvement to a later issue? I think we should give this a little more thought, for example if we do this its a break in the API for <similarityFactory/>, which this patch does not actually do: it only MOVES it to the fieldType.
    **label:** code-design
13. **body:** By the way on the last part, its probably hard to see in the patch: this is because SimilarityFactory is currently (temporarily) backwards broken versus say, Solr 1.4, because I didn't want to take away any Solr capabilities until we resolved this issue... so in trunk right now it returns SimilarityProvider (which i know makes looking at the patch confusing).
    **label:** code-design
14. I see, as far as using NamedList for the new SimilarityProviderFactory, that would be easy. However, at the moment my vote is against this, at least until SOLR-2292 is completed. At the moment NamedList

contains methods such as get(String) which have a slow linear runtime, and just out of paranoia I think we should keep NamedList as far away from the scoring system as possible.

15. **body:** i was confused by some of roberts comments, and clarified them with him on IRC. summary (from my perspective) * "global default" similarity(factory) (using existing {{<similarity/>}} tag) is a good idea as a fall back for fieldTypes that don't define custom similarity * {{<similarity/>}} should probably not be advertised in the example configs .. but maybe, depends * SimilarityProvider should use a distinct config tag ({{<similarityProvider/>}} because it really is distinct, and people should (in theory) be able to use both) * SolrSimilarityProvider's get(field) method (which i didn't realize was final, hence part of my confusion) should be changed to use the {{<similarity/>}} as a default if it was specified. * SolrSimilarityProvider's get(field) method really needs to stay final, and should have docs explain why (consistency with schema) * SimilarityProviderFactory.init can be changed to using NamedList, but the docs should warn people about the possibility of performance penalties for using it directly in their SolrSimilarityProvider
    **label:** code-design
16. **body:** bq. <similarity/> should probably not be advertised in the example configs .. Do you mean the expert-level SimilarityProvider, which is typically only needed to change coord() or queryNorm()? If so, I'd agree - it's so expert level that most people shouldn't worry about it, hence it can simply be documented elsewhere.
    **label:** documentation
17. **body:** bq. Do you mean the expert-level SimilarityProvider, no i ment the existing global default {{<similarity/>}} mentioned in the previous bullet (but i was definitley vague -- sorry). we probably shouldn't go out of our way to advertise the global option anymore, we should encourage people to use the fieldType specific similarity instead. whether we should promote customizing the SimilarityProvider in the example .. i dunno, i can see it either way. but it seems like the order of importance for visibility is probably something like: * customize per fieldType similarity (new hotness) * customize similarityprovider (not something you are likely to need to do, but if you do there's really only one way so maybe we should advertise it's an option) * customize global default similarity (odds are you really don't need this, you can probably do it using per fieldType similarity)
    **label:** code-design
18. **body:** I agree with your order, i think to keep the example simple we should just have a commented out example for a fieldType that customizes its similarity? I think we can support customizing the similarityprovider (in case you want to change how coord is calculated), and support changing the default-unless-otherwise-specified-in-the-schema similarity (for easier backwards transition), but I'm thinking it would be actually more confusing than helpful to advertise these in the example.
    **label:** code-design
19. **body:** I dunno - it seems much more likely that someone would want to easily change the default sim for all fields rather than change coord.
    **label:** code-design
20. OK, well there's clearly no consensus on the example, but fortunately none of this need block this issue. I'll add support for default <similarity/>, and leave the example unchanged. we can open a followup issue and debate this example stuff separately.
21. **body:** attached is the modifications: * default <similarity/> is used as a fallback if not otherwise specified in the fieldtype * simprovider takes namedlist * the get(String) and remove(String) methods are documented to run in linear time. * i left the example unchanged, we can TBD this for later.
    **label:** code-design
22. Committed revision 1087430. Thanks hoss and yonik for feedback.
23. test-files/solr/conf/schema.xml contains sample of per-field definitions; example/solr/schema.xml doesn't have it yet
24. Fuad: note from the comments above that this was intentional -- for now. i've opened SOLR-2600 to ensure that this is dealt with before the feature is released.