Item 291
git_comments:

1. * * Get the first available {@link Summarizer} extension. * @return the first available {@link Summarizer} extension, or * <code>null</code> if none available.
2. * * Copyright 2005 The Apache Software Foundation * * Licensed 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.
3. Hadoop imports
4. * * A factory for retrieving {@link Summarizer} extensions. * * @author Jérôme Charron
5. Nutch imports
6. JDK imports
7. * The first available {@link Summarizer}
8. * My logger
9. * * Tests Summary-generation. User inputs the name of a * text file and a query string
10. * * Copyright 2005 The Apache Software Foundation * * Licensed 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.
11. ---------- * * </implementation:Configurable> * * ----------
12. * * How many fragments we have.
13. * * Return an Enum for all the fragments
14. Hadoop imports
15. * * Class Excerpt represents a single passage found in the document, with some * appropriate regions highlit.
16. ---------- * * </implementation:Summarizer> * * ----------
17. Lucene imports
18. ignore
19. Nutch imports
20. * * Return how many unique toks we have
21. ---------- * * <implementation:Configurable> * * ----------
22. *
23. ---------- * * <implementation:Summarizer> * * ----------
24. We found the series of search-term hits and added them (with intervening text) to the excerpt. Now we need to add the trailing edge of text. So if (j < tokens.length) then there is still trailing text to add. (We haven't hit the end of the source doc.) Add the words since the last hit-term insert.
25. JDK imports
26. * * Add a frag to the list.
27. * * Copyright 2005 The Apache Software Foundation * * Licensed 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.
28. ---------- * * </implementation:Configurable> * * ----------
29. Nothing to do...
30. ---------- * * <implementation:Summarizer> * * ----------
31. Hadoop imports
32. Nutch imports
33. **comment:** TODO : The max number of fragments (3) should be configurable
    label: code-design
34. ---------- * * </implementation:Summarizer> * * ----------
35. JDK imports

36. ---------- * * <implementation:Configurable> * * ----------
37. Lucene imports
38. * * Extension point for summarizer. * * @author Jérôme Charron
39. * The name of the extension point.
40. Hadoop imports
41. Nutch imports

git_commits:

1. **summary:** NUTCH-134 : Added a summarizer extension point and two enxtensions:
   message: NUTCH-134 : Added a summarizer extension point and two enxtensions: * summary-basic is the current nutch implementation moved into a plugin * summary-lucene a raw version of a summarizer plugin based on lucene highlighter git-svn-id: https://svn.apache.org/repos/asf/lucene/nutch/trunk@405165 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Summarizer doesn't select the best snippets
   description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
2. **summary:** Summarizer doesn't select the best snippets
   description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
3. **summary:** Summarizer doesn't select the best snippets
   description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
4. **summary:** Summarizer doesn't select the best snippets
   description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add()

operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.

5. **summary:** Summarizer doesn't select the best snippets
   description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
6. **summary:** Summarizer doesn't select the best snippets
   description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
7. **summary:** Summarizer doesn't select the best snippets
   description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
   label: test
8. **summary:** Summarizer doesn't select the best snippets
   description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
9. **summary:** Summarizer doesn't select the best snippets
   description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
   label: code-design

10. **summary:** Summarizer doesn't select the best snippets
    description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
11. **summary:** Summarizer doesn't select the best snippets
    description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
12. **summary:** Summarizer doesn't select the best snippets
    description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
13. **summary:** Summarizer doesn't select the best snippets
    description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
    label: code-design
14. **summary:** Summarizer doesn't select the best snippets
    description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
15. **summary:** Summarizer doesn't select the best snippets
    description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the

Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
label: code-design

16. **summary:** Summarizer doesn't select the best snippets
description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
17. **summary:** Summarizer doesn't select the best snippets
description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
18. **summary:** Summarizer doesn't select the best snippets
description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
19. **summary:** Summarizer doesn't select the best snippets
description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.
label: code-design
20. **summary:** Summarizer doesn't select the best snippets
description: Summarizer.java tries to select the best fragments from the input text, where the frequency of query terms is the highest. However, the logic in line 223 is flawed in that the excerptSet.add() operation will add new excerpts only if they are not already present - the test is performed using the Comparator that compares only the numUniqueTokens. This means that if there are two or more excerpts, which score equally high, only the first of them will be retained, and the rest of equally-scoring excerpts will be discarded, in favor of other excerpts (possibly lower scoring). To fix this the Set should be replaced with a List + a sort operation. To keep the relative position of excerpts in the original order the Excerpt class should be extended with an "int order" field, and the collected excerpts should be sorted in that order prior to adding them to the summary.

jira_issues_comments:

1. Can we yet replace Nutch's summarizer with the summarizer in Lucene's contrib directory? Are there features that Nutch requires that that does not yet implement? It's a shame to maintain two summarizers.

When I first wrote Nutch's summarizer there was no Lucene contrib summarizer...
2. I _think_ the Lucene summarizer requires more CPU than this one... but this has to be checked. I'll work on that.
3. I would take more cpu for better summaries any day :) cpu power is cheaper than manual intervention! If any testing is needed, don't hesitate to drop me a patch.. i've been working on a 500million page index using mapred branch on a 10 node cluster so i have plenty of numbers to test against.
4. Where is the lucene summarizer from the contrib? i'm not seeing anything obvious (unless it's under a different name)
5. Byron - It's under contrib/highlighter. For Nutch, which uses Lucene's trunk version, you'll want to build the Highlighter from scratch against that version of Lucene.
6. Thanks Erik, I was able to pull down the highlighter and i'll be loading it up on mozdex.com to test out over the weekend (1/21/2006). i'll let people know if my cpu skyrockets :)
7. **body:** byron, Did you ever get a chance to run a cpu perf test on using lucene/contrib/highlighter for extracting summaries? chris
   label: test
8. Since we can imagine a lot of Summarizer implementation, and that each kind of Nutch deployment can have some specific needs about Summarizer, I suggest to create : 1. a Summarizer extension point, so that we can easily switch from one implementation to another one. 2. a summarizer-basic plugin that is in fact the current summarizer implementation. 3. a summarizer-lucene plugin that wraps the lucene's summarizer. If everybody is ok, i will implement this.
9. **body:** Please consider changing the API to return an array of writables as a result, instead of the current single String/UTF8. There are many applications (non-html output, clustering) that could greatly benefit from this change.
   label: code-design
10. Jerome, Let me know if you could use a hand in implementation. I'd like to get to know nutch and lucene code base better for my project. This looks like a good area to start in, so any opportunity to jump in would be great. chris
11. Andrzej, my solution to this problem was to fix the comparator to actually compare the fragments if numFragments() was the same for both excerpts. Sounds like there are grander plans afoot, but this got me past my problem of only seeing one summary fragment when I actually had 3 (they were seen as equal so only the last was on the set). Steven
12. Here is a patch that add a summarizer extension point and two summarizer plugins : summarizer-basic (the current nutch implementation) and summarizer-lucene (the lucene highlighter implementation). Please notice that the lucene plugin is a very crude implementation : the highlighter directly constructs a text representation of the summary, so we need to parse the text to build a Summary object!!! (improvements are welcome). This is a first step to this issue resolution. If no objection, I will commit this patch in the next few days and then: 1. Fix in the summarizer-basic the original issue reported by Andrzej 2. Add a toString(Encoder, Formatter) method in Summarizer so that a Summary object could be encoded and formatted with many implementations (it is the same logic as the one in Lucene Highlight) - Andrzej, do you prefer this solution or a solution where Summary is Writable? PS: Chris, sorry but the major part of this patch was already done when you added your comment.
13. **body:** I still prefer Summary as Writable. The reason is that there are users of Summary that don't want a single String with HTML formatting - recovering from this format is tedious and error-prone. On the other hand, returning a Writable may have performance implications.
    label: code-design
14. (back from holidays, so a bit delayed, but) I confirm Andrzej's suggestion -- a plain-text only summarized is ideal for clustering for example. HTML is quite uncomfortable to work with.
15. **body:** +1 for Summary as Writable and change HitSummarizer.getSummary() to return a Summary directly rather than a String. I don't think this has bad performance implications.
    label: code-design
16. Linking to a related issue that would benefit if Summarizer returned Summary objects rather than Strings.
17. Solution proposed by Andrzej implemented.
18. closing issues for released versions