Item 337
git_comments:

1. * Advances to the next position and returns true, or returns false if it can't.
2. * * This class is used to ensure we don't over iterate the underlying * postings enum by keeping track of the position relative to the * frequency. * Ideally this would've been an implementation of a PostingsEnum * but it would have to delegate most methods and it seemed easier * to just wrap the tweaked method.
3. ignore the actual position; we don't care.
4. advance head the new position may be behind another postingsEnum in the queue
5. * Advances to the next position. Always returns -1; the caller is assumed not to care for the highlighter.
6. * * Provides a view over several underlying PostingsEnums for the iteration of offsets on the current document only. * It's not general purpose; the position returned is always -1 and it doesn't iterate the documents.
7. at most 1 doc is returned
8.
9. this postingsEnum is consumed; get rid of it. Another will take it's place.
10. * The provided {@link PostingsEnum}s must all be positioned to the same document, and must have offsets.
11. * * Build one {@link CharacterRunAutomaton} matching any term the query might match.
12. might be needed true==store offsets appears to be re-usable preFilter for MemoryIndex
13.
14. note: calls tokenStream.reset() & close()
15. * * Uses an {@link Analyzer} on content to get offsets and then populates a {@link MemoryIndex}. * * @lucene.internal
16. but this would have a performance cost for likely little gain in the user experience, it would only serve to make this method less bogus. instead, we always return freq() = Integer.MAX_VALUE and let the highlighter terminate based on offset... TODO: DWS perhaps instead OffsetsEnum could become abstract and this would be an impl?
17.
18. * * Analyzes the text, producing a single {@link OffsetsEnum} wrapping the {@link TokenStream} filtered to terms * in the query, including wildcards. It can't handle position-sensitive queries (phrases). Passage accuracy suffers * because the freq() is unknown -- it's always {@link Integer#MAX_VALUE} instead.
19. EOF
20. Offset tracking:
21. Get new tokenStream based on next segment divided by the splitChar
22. subTokenStream is already initialized to operate on the first value
23. Position tracking: usually true first token of additional values (not first val)
24. while loop to increment token of this new value
25. no more
26. reset Offset tracking:
27. (input is defined in TokenFilter set in the constructor) This is a grand trick we do -- knowing that the analyzer's re-use strategy is going to produce the very same tokenStream instance and thus have the same AttributeSource as this wrapping TokenStream since we used it as our input in the constructor. Were this not the case, we'd have

to copy every attribute of interest since we can't alter the AttributeSource of this wrapping TokenStream post-construction (it's all private/final). If this is a problem, we could do that instead; maybe with a custom CharTermAttribute that allows us to easily set the char[] reference without literally copying char by char.
28. note: 1 is the default. It is RARELY changed.
29. might adjust position increment
30. ... although we could if a need for it arises.
31. If there is no splitChar in content then we needn't wrap:
32. don't wrap in a composite if there's only one OffsetsEnum
33. Query: position-sensitive information TODO: rename Query: free-standing terms Query: free-standing wildcards (multi-term query)
34. Handle sourceTerms:
35. will be at most this long
36. only add if we have offsets
37. Handle automata
38. For analysis, prefer MemoryIndexOffsetStrategy
39. skip using a memory index since it's pure term filtering
40. a little past first sentence
41. tests interleaving of multiple wildcard matches with the CompositePostingsEnum In this case the CompositePostingsEnum will have an underlying PostingsEnum that jumps form pos 1 to 9 for bravo and a second with position 2 for Bravado
42. ask for 2 but we'll only get 1
43. note: all offset sources, by default, use term freq, so it shouldn't matter which we choose.

git_commits:

1. **summary:** LUCENE-7526: UnifiedHighlighter: enhance MTQ passage relevancy. TokenStreamFromTermVector isn't used by the UH anymore. Refactor AnalysisOffsetStrategy into TokenStream and MemoryIndex strategies, and related refactorings from that.
   message: LUCENE-7526: UnifiedHighlighter: enhance MTQ passage relevancy. TokenStreamFromTermVector isn't used by the UH anymore. Refactor AnalysisOffsetStrategy into TokenStream and MemoryIndex strategies, and related refactorings from that.

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Improvements to UnifiedHighlighter OffsetStrategies
   description: This ticket improves several of the UnifiedHighlighter FieldOffsetStrategies by reducing reliance on creating or re-creating TokenStreams. The primary changes are as follows: * AnalysisOffsetStrategy - split into two offset strategies ** MemoryIndexOffsetStrategy - the primary analysis mode that utilizes a MemoryIndex for producing Offsets ** TokenStreamOffsetStrategy - an offset strategy that avoids creating a MemoryIndex. Can only be used if the query distills down to terms and automata. * TokenStream removal ** MemoryIndexOffsetStrategy - previously a TokenStream was created to fill the memory index and then once consumed a new one was generated by uninverting the MemoryIndex back into a TokenStream if there were automata (wildcard/mtq queries) involved. Now this is avoided, which should save memory and avoid a second pass over the data. ** TermVectorOffsetStrategy - this was refactored in a similar way to avoid generating a TokenStream if automata are involved. ** PostingsWithTermVectorsOffsetStrategy - similar refactoring * CompositePostingsEnum - aggregates several underlying PostingsEnums for wildcard/mtq queries. This should improve relevancy by providing unified metrics for a wildcard across all it's term matches * Added a HighlightFlag for enabling the newly separated TokenStreamOffsetStrategy since it can adversely affect passage relevancy

jira_issues_comments:

1. Pull request forthcoming - I had some more merging work to do with master than I anticipated!

2. I'm looking forward to seeing this. :-) The summary heading "TokenStream removal" may be confusing to folks... TokenStreams are certainly going to be involved for the Analysis based offset source. I think you mean that the changes here relating to MultiTermQuery processing will mean that MTQs aren't coerced into a TokenStream over the index any more, except for the refactored-out TokenStreamOffsetStrategy. So this removes many but not all occurrences of TokenStream within the internal API.
3. GitHub user Timothy055 opened a pull request: https://github.com/apache/lucene-solr/pull/105 LUCENE-7526 Improvements to UnifiedHighlighter OffsetStrategies Pull request for LUCENE-7526 You can merge this pull request into a Git repository by running: $ git pull https://github.com/Timothy055/lucene-solr master Alternatively you can review and apply these changes as the patch at: https://github.com/apache/lucene-solr/pull/105.patch To close this pull request, make a commit to your master/trunk branch with (at least) the following in the commit message: This closes #105 ---- commit 02e932c4a6146363680b88f4947a693c6697c955 Author: Timothy Rodriguez <trodriguez25@bloomberg.net> Date: 2016-09-01T19:23:50Z Initial fork of PostingsHighlighter for UnifiedHighlighter commit 9d88411b3985a98851384d78d681431dba710e89 Author: Timothy Rodriguez <trodriguez25@bloomberg.net> Date: 2016-09-01T23:17:06Z Initial commit of the UnifiedHighlighter for OSS contribution commit e45e39bc4b07ea33e4423b264c2fefb9aa08777a Author: David Smiley <david.w.smiley@gmail.com> Date: 2016-09-02T12:45:49Z Fix misc issues; "ant test" now works. (#1) commit 046a28ef31acf4cea7d255bbbb4b827e6a714e3d Author: Timothy Rodriguez <trodriguez25@bloomberg.net> Date: 2016-09-02T20:58:31Z Minor refactoring of the AnalysisFieldHighlighter commit ccd1a2280abd4b48cfef8122696e5d9cfd12920f Author: David Smiley <dsmiley@apache.org> Date: 2016-09-03T12:55:20Z AbstractFieldHighlighter: order methods more sensibly; renamed a couple. commit d4714a04a3e41d5e95bbe942b275c32ed69b9c2e Author: David Smiley <dsmiley@apache.org> Date: 2016-09-04T01:03:29Z Improve javadocs and @lucene.external/internal labeling & scope. "ant precommit" now passes. commit e0659f18a59bf2893076da6d7643ff30f2fa5a52 Author: David Smiley <dsmiley@apache.org> Date: 2016-09-04T01:25:55Z Analysis: remove dubious filter() method commit ccd7ce707bff2c06da89b31853cca9aecea72008 Author: David Smiley <dsmiley@apache.org> Date: 2016-09-04T01:44:01Z getStrictPhraseHelper -> rm "Strict", getHighlightAccuracy -> getFlags, and only call filterExtractedTerms once. commit ffc2a22c700b8abcbf87673d5d05bb3659d177c9 Author: David Smiley <david.w.smiley@gmail.com> Date: 2016-09-04T15:21:08Z UnifiedHighlighter round 2 (#2) * AbstractFieldHighlighter: order methods more sensibly; renamed a couple. * Improve javadocs and @lucene.external/internal labeling & scope. "ant precommit" now passes. * Analysis: remove dubious filter() method * getStrictPhraseHelper -> rm "Strict", getHighlightAccuracy -> getFlags, and only call filterExtractedTerms once. commit 5f95e05595db462d3ab5bffc68c2c92f70875072 Author: David Smiley <dsmiley@apache.org> Date: 2016-09-04T16:12:33Z Refactor: FieldOffsetStrategy commit 86fb6265fbbdb955ead6d4baf944bf708175715e Author: David Smiley <dsmiley@apache.org> Date: 2016-09-04T16:21:32Z stop passing maxPassages into highlightFieldForDoc() commit f6fd80544eae9fab953b94b1e9346c0883f956eb Author: David Smiley <dsmiley@apache.org> Date: 2016-09-04T16:12:33Z Refactor: FieldOffsetStrategy commit b335a673c2ce45904890c1e9af7cbfda2bd27b0f Author: David Smiley <dsmiley@apache.org> Date: 2016-09-04T16:21:32Z stop passing maxPassages into highlightFieldForDoc() commit 478db9437b92214cbf459f82ba2e3a67c966a150 Author: David Smiley <dsmiley@apache.org> Date: 2016-09-04T18:29:44Z Rename subclasses of FieldOffsetStrategy. commit dbf4280755c11420a5032445cd618fadb7444b61 Author: David Smiley <dsmiley@apache.org> Date: 2016-09-04T18:31:34Z Re-order and harmonize params on methods called by UH.getFieldHighlighter() commit f0340e27e61dcda2e11992f08ec07a72fad6c24c Author: David Smiley <dsmiley@apache.org> Date: 2016-09-04T18:53:51Z FieldHighlighter: harmonize field/param order. And don't apply maxNoHighlightPasses twice. commit 817f63c1d48fd523c13b9c40a2ae9b8a4047209a Author: Timothy Rodriguez <trodriguez25@bloomberg.net> Date: 2016-09-06T20:43:20Z Merge of renaming changes commit 0f644a4f53c1ed4d41d562848f6fe51a87442a75 Author: Timothy Rodriguez <trodriguez25@bloomberg.net> Date: 2016-09-06T20:54:13Z add visibility tests commit 9171f49e117085e7d086267bb73836831ff07f8e Author: Timothy Rodriguez <trodriguez25@bloomberg.net> Date: 2016-09-07T14:26:59Z ADd additional extensibility test commit 7ce488147cb811e15cb6e9125a835171157746f2 Author: Timothy Rodriguez <trodriguez25@bloomberg.net> Date: 2016-09-28T22:04:15Z Reduce visibility of MultiTermHighlighting to package protected commit 2f08465020448592b0e8750db568ade5a9218267 Author: Timothy M. Rodriguez <timothy.rodriguez@gmail.com> Date: 2016-10-11T16:44:29Z Initial commit that will use memory index to generate offsets enum if the tokenstream is null commit 357f3dfb9ace4deef20787af19bc2e5a6b4ff61e Author: Timothy M. Rodriguez <timothy.rodriguez@gmail.com> Date: 2016-10-11T17:34:51Z Switched analysis offset strategy to not re-build a tokenstream commit 64153d288db5714cdaf3726328557f65c1635610 Author: Timothy M. Rodriguez <timothy.rodriguez@gmail.com> Date: 2016-10-11T17:42:12Z Switched to using chars ref builder commit f137779b1e1b7e57c4b78652614a04507b9e09e1 Author: Timothy Rodriguez <trodriguez25@bloomberg.net> Date: 2016-10-21T19:07:48Z minor cleanup commit ec814f974db6459eba7aa45bf7a4cdae04e6ad6f Author: Timothy M. Rodriguez <timothy.rodriguez@gmail.com> Date: 2016-10-21T21:25:57Z switch to use of a CompositePostingsEnum that

wraps the postings of wildcard matches commit 955a1e79b5189492fae2c95da39343c29e1cdb25 Author: Timothy M. Rodriguez <timothy.rodriguez@gmail.com> Date: 2016-10-21T21:32:48Z merge conflicts on PhraseHelper rename commit d35bd1cffd3c6e2aed67aeccfd03959bf855670a Author: Timothy Rodriguez <trodriguez25@bloomberg.net> Date: 2016-10-21T22:13:51Z minor cleanup of how automata are handled in the FieldOffsetStrategy commit aa8c92667272e5f397b9566cde05eae7e31bcce5 Author: Timothy Rodriguez <trodriguez25@bloomberg.net> Date: 2016-10-24T20:03:08Z Removed most use of TokenStreams except in pure Analysis commit db42d6a959ca19a77dee3cc7b09496a21c631bd6 Author: Timothy Rodriguez <trodriguez25@bloomberg.net> Date: 2016-10-24T20:08:25Z simplified some logic to not use continue statements commit 657c2a70c4d8ec8ef850e9f66b93cf85ec16f636 Author: Timothy Rodriguez <trodriguez25@bloomberg.net> Date: 2016-10-24T22:56:17Z split analysis mode into two, moved all offset sources from readers into the FieldOffsetStrategy ----

4. Thanks [~dsmiley] :). I've just submitted the pull request. You're right this only removes an additional use of token streams. In the case of the Analysis strategies a TokenStream is still necessary at least initially to analyze the field. I'm glad I got to work on this during the wonderful Boston Hackday event (https://github.com/flaxsearch/london-hackday-2016). Thanks [~dsmiley] for some tips while there and [~mbraun688] for some initial feedback on the pr.
5. Github user dsmiley commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85602404 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/CompositePostingsEnum.java --- @@ -0,0 +1,165 @@ +/* + * 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. + */ +package org.apache.lucene.search.uhighlight; + +import java.io.IOException; +import java.util.List; + +import org.apache.lucene.index.PostingsEnum; +import org.apache.lucene.util.BytesRef; +import org.apache.lucene.util.PriorityQueue; + + +final class CompositePostingsEnum extends PostingsEnum { --- End diff -- A comment would be helpful to explain the scope/purpose.
6. Github user dsmiley commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85602210 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/AnalysisOffsetStrategy.java --- @@ -17,174 +17,28 @@ package org.apache.lucene.search.uhighlight; import java.io.IOException; -import java.util.ArrayList; -import java.util.Arrays; -import java.util.Collections; -import java.util.List; import org.apache.lucene.analysis.Analyzer; -import org.apache.lucene.analysis.FilteringTokenFilter; import org.apache.lucene.analysis.TokenStream; -import org.apache.lucene.analysis.tokenattributes.CharTermAttribute; -import org.apache.lucene.index.IndexReader; -import org.apache.lucene.index.LeafReader; -import org.apache.lucene.index.Terms; -import org.apache.lucene.index.memory.MemoryIndex; -import org.apache.lucene.search.spans.SpanQuery; import org.apache.lucene.util.BytesRef; -import org.apache.lucene.util.automaton.Automata; import org.apache.lucene.util.automaton.CharacterRunAutomaton; +public abstract class AnalysisOffsetStrategy extends FieldOffsetStrategy { --- End diff -- All public classes need a javadoc comment. Remember lucene.internal for this one.
7. Github user dsmiley commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85603812 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/CompositePostingsEnum.java --- @@ -0,0 +1,165 @@ +/* + * 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. + */ +package org.apache.lucene.search.uhighlight; + +import java.io.IOException; +import java.util.List; + +import org.apache.lucene.index.PostingsEnum; +import org.apache.lucene.util.BytesRef; +import org.apache.lucene.util.PriorityQueue; + + +final class CompositePostingsEnum extends PostingsEnum { + + private static final int NO_MORE_POSITIONS = -2; + private final BytesRef term; + private final int freq; + private final PriorityQueue<BoundsCheckingPostingsEnum> queue; + + + /** + * This class is used to ensure we don't over iterate the underlying + * postings enum by keeping track of the position relative to the + *

frequency. + * Ideally this would've been an implementation of a PostingsEnum + * but it would have to delegate most methods and it seemed easier + * to just wrap the tweaked method. + */ + private static final class BoundsCheckingPostingsEnum { + + + private final PostingsEnum postingsEnum; + private final int freq; + private int position; + private int nextPosition; + private int positionInc = 1; + + private int startOffset; + private int endOffset; + + BoundsCheckingPostingsEnum(PostingsEnum postingsEnum) throws IOException { + this.postingsEnum = postingsEnum; + this.freq = postingsEnum.freq(); + nextPosition = postingsEnum.nextPosition(); + position = nextPosition; + startOffset = postingsEnum.startOffset(); + endOffset = postingsEnum.endOffset(); + } + + private boolean hasMorePositions() throws IOException { + return positionInc < freq; + } + + /** + * Returns the next position of the underlying postings enum unless + * it cannot iterate further and returns NO_MORE_POSITIONS; + * @return + * @throws IOException + */ + private int nextPosition() throws IOException { + position = nextPosition; + startOffset = postingsEnum.startOffset(); + endOffset = postingsEnum.endOffset(); + if (hasMorePositions()) { + positionInc++; + nextPosition = postingsEnum.nextPosition(); + } else { + nextPosition = NO_MORE_POSITIONS; + } + return position; + } + + } + + CompositePostingsEnum(BytesRef term, List<PostingsEnum> postingsEnums) throws IOException { + this.term = term; + queue = new PriorityQueue<BoundsCheckingPostingsEnum>(postingsEnums.size()) { + @Override + protected boolean lessThan(BoundsCheckingPostingsEnum a, BoundsCheckingPostingsEnum b) { + return a.position < b.position; --- End diff -- In the event the positions are equal (e.g. two terms a the same position in which the wildcard matches both), we might want to fall-back on startOffset then endOffset? Or maybe simply ignore position altogether and just do offsets, so then you needn't even track the position?

8. Github user dsmiley commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85607262 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/TokenStreamOffsetStrategy.java --- @@ -0,0 +1,60 @@ +/* + * 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. + */ +package org.apache.lucene.search.uhighlight; + +import java.io.Closeable; +import java.io.IOException; +import java.util.Collections; +import java.util.List; + +import org.apache.lucene.analysis.Analyzer; +import org.apache.lucene.analysis.TokenStream; +import org.apache.lucene.index.IndexReader; +import org.apache.lucene.index.PostingsEnum; +import org.apache.lucene.util.BytesRef; +import org.apache.lucene.util.automaton.CharacterRunAutomaton; + +public class TokenStreamOffsetStrategy extends AnalysisOffsetStrategy { + + private static final BytesRef[] ZERO_LEN_BYTES_REF_ARRAY = new BytesRef[0]; + + public TokenStreamOffsetStrategy(String field, BytesRef[] terms, PhraseHelper phraseHelper, CharacterRunAutomaton[] automata, Analyzer indexAnalyzer) { + super(field, terms, phraseHelper, automata, indexAnalyzer); + this.automata = convertTermsToAutomata(terms, automata); + this.terms = ZERO_LEN_BYTES_REF_ARRAY; + } + + @Override + public List<OffsetsEnum> getOffsetsEnums(IndexReader reader, int docId, String content) throws IOException { + TokenStream tokenStream = tokenStream(content); + PostingsEnum mtqPostingsEnum = MultiTermHighlighting.getDocsEnum(tokenStream, automata); --- End diff -- I think there's a case to be made in moving ` MultiTermHighlighting.getDocsEnum` into this class, to thus keep the TokenStream aspect more isolated?
9. Github user dsmiley commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85602867 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/CompositePostingsEnum.java --- @@ -0,0 +1,165 @@ +/* + * 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. + */ +package org.apache.lucene.search.uhighlight; + +import java.io.IOException; +import java.util.List; + +import org.apache.lucene.index.PostingsEnum; +import org.apache.lucene.util.BytesRef; +import org.apache.lucene.util.PriorityQueue; + + +final class CompositePostingsEnum extends PostingsEnum { + + private static final int NO_MORE_POSITIONS = -2; + private final BytesRef term; + private final int freq; + private final PriorityQueue<BoundsCheckingPostingsEnum> queue; + + + /** + * This class is used to ensure

we don't over iterate the underlying + * postings enum by keeping track of the position relative to the + * frequency. + * Ideally this would've been an implementation of a PostingsEnum + * but it would have to delegate most methods and it seemed easier + * to just wrap the tweaked method. + */ + private static final class BoundsCheckingPostingsEnum { + + + private final PostingsEnum postingsEnum; + private final int freq; --- End diff -- Instead of holding `freq` and `nextPosition`, why not just `remainingPositions`?

10. Github user dsmiley commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85608645 --- Diff: lucene/highlighter/src/test/org/apache/lucene/search/uhighlight/visibility/TestUnifiedHighlighterExtensibility.java --- @@ -79,7 +90,7 @@ public void testFieldOffsetStrategyExtensibility() { @Test public void testUnifiedHighlighterExtensibility() { final int maxLength = 1000; - UnifiedHighlighter uh = new UnifiedHighlighter(null, new MockAnalyzer(random())){ + UnifiedHighlighter uh = new UnifiedHighlighter(null, new MockAnalyzer(new Random())){ --- End diff -- ? why not `random()` ? This will likely fail precommit.
11. Github user dsmiley commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85606885 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/Passage.java --- @@ -40,7 +40,7 @@ BytesRef matchTerms[] = new BytesRef[8]; int numMatches = 0; - void addMatch(int startOffset, int endOffset, BytesRef term) { + public void addMatch(int startOffset, int endOffset, BytesRef term) { --- End diff -- Excellent; now it's possible for someone to override `FieldHighlighter.highlightOffsetsEnums()` to make Passages. But do add @lucene.experimental.
12. Github user dsmiley commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85607859 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/UnifiedHighlighter.java --- @@ -116,6 +116,8 @@ private boolean defaultHighlightPhrasesStrictly = true; // AKA "accuracy" or "query debugging" + private boolean defaultPassageRelevancyOverSpeed = true; //Prefer using a memory index --- End diff -- Suggest the comment be: For analysis, prefer MemoryIndex approach
13. Github user dsmiley commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85603003 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/CompositePostingsEnum.java --- @@ -0,0 +1,165 @@ +/* + * 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. + */ +package org.apache.lucene.search.uhighlight; + +import java.io.IOException; +import java.util.List; + +import org.apache.lucene.index.PostingsEnum; +import org.apache.lucene.util.BytesRef; +import org.apache.lucene.util.PriorityQueue; + + +final class CompositePostingsEnum extends PostingsEnum { + + private static final int NO_MORE_POSITIONS = -2; + private final BytesRef term; + private final int freq; + private final PriorityQueue<BoundsCheckingPostingsEnum> queue; + + + /** + * This class is used to ensure we don't over iterate the underlying + * postings enum by keeping track of the position relative to the + * frequency. + * Ideally this would've been an implementation of a PostingsEnum + * but it would have to delegate most methods and it seemed easier + * to just wrap the tweaked method. + */ + private static final class BoundsCheckingPostingsEnum { + + + private final PostingsEnum postingsEnum; + private final int freq; + private int position; + private int nextPosition; + private int positionInc = 1; + + private int startOffset; --- End diff -- Don't need these. They can be fetched on-demand from the head of the queue, easily & cheaply enough.
14. Github user dsmiley commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85606333 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/FieldOffsetStrategy.java --- @@ -65,58 +65,88 @@ public String getField() { */ public abstract List<OffsetsEnum> getOffsetsEnums(IndexReader reader, int docId, String content) throws IOException; - protected List<OffsetsEnum> createOffsetsEnums(LeafReader leafReader, int doc, TokenStream tokenStream) throws IOException { - List<OffsetsEnum> offsetsEnums = createOffsetsEnumsFromReader(leafReader, doc); - if (automata.length > 0) { - offsetsEnums.add(createOffsetsEnumFromTokenStream(doc, tokenStream)); + protected List<OffsetsEnum> createOffsetsEnumsFromReader(LeafReader leafReader, int doc) throws IOException { + final Terms termsIndex = leafReader.terms(field); + if (termsIndex == null) { + return Collections.emptyList(); } - return offsetsEnums; - } - protected List<OffsetsEnum> createOffsetsEnumsFromReader(LeafReader atomicReader, int doc) throws IOException { // For strict positions, get a Map of term to Spans: // note: ScriptPhraseHelper.NONE does the right thing for these method calls final Map<BytesRef, Spans> strictPhrasesTermToSpans = - strictPhrases.getTermToSpans(atomicReader, doc); +

phraseHelper.getTermToSpans(leafReader, doc); // Usually simply wraps terms in a List; but if willRewrite() then can be expanded final List<BytesRef> sourceTerms = - strictPhrases.expandTermsIfRewrite(terms, strictPhrasesTermToSpans); + phraseHelper.expandTermsIfRewrite(terms, strictPhrasesTermToSpans); - final List<OffsetsEnum> offsetsEnums = new ArrayList<>(sourceTerms.size() + 1); + final List<OffsetsEnum> offsetsEnums = new ArrayList<>(sourceTerms.size() + automata.length); - Terms termsIndex = atomicReader == null || sourceTerms.isEmpty() ? null : atomicReader.terms(field); - if (termsIndex != null) { + // Handle sourceTerms: + if (!sourceTerms.isEmpty()) { TermsEnum termsEnum = termsIndex.iterator();//does not return null for (BytesRef term : sourceTerms) { - if (!termsEnum.seekExact(term)) { - continue; // term not found - } - PostingsEnum postingsEnum = termsEnum.postings(null, PostingsEnum.OFFSETS); - if (postingsEnum == null) { - // no offsets or positions available - throw new IllegalArgumentException("field '" + field + "' was indexed without offsets, cannot highlight"); - } - if (doc != postingsEnum.advance(doc)) { // now it's positioned, although may be exhausted - continue; + if (termsEnum.seekExact(term)) { + PostingsEnum postingsEnum = termsEnum.postings(null, PostingsEnum.OFFSETS); + + if (postingsEnum == null) { + // no offsets or positions available + throw new IllegalArgumentException("field '" + field + "' was indexed without offsets, cannot highlight"); + } + + if (doc == postingsEnum.advance(doc)) { // now it's positioned, although may be exhausted + postingsEnum = phraseHelper.filterPostings(term, postingsEnum, strictPhrasesTermToSpans.get(term)); + if (postingsEnum != null) { + offsetsEnums.add(new OffsetsEnum(term, postingsEnum)); + } + } } - postingsEnum = strictPhrases.filterPostings(term, postingsEnum, strictPhrasesTermToSpans.get(term)); - if (postingsEnum == null) { - continue;// completely filtered out + } + } + + // Handle automata + if (automata.length > 0) { + offsetsEnums.addAll(createAutomataOffsetsFromTerms(termsIndex, doc)); + } + + return offsetsEnums; + } + + protected List<OffsetsEnum> createAutomataOffsetsFromTerms(Terms termsIndex, int doc) throws IOException { + Map<CharacterRunAutomaton, List<PostingsEnum>> automataPostings = new IdentityHashMap<>(automata.length); --- End diff -- I suggest a parallel array to automata, so that later you can avoid a map lookup on each matching term. Also, I suggest lazy-initializing the array later... perhaps some wildcards in a disjunction might never match.

15. Github user dsmiley commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85604667 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/CompositePostingsEnum.java --- @@ -0,0 +1,165 @@ +/* + * 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. + */ +package org.apache.lucene.search.uhighlight; + +import java.io.IOException; +import java.util.List; + +import org.apache.lucene.index.PostingsEnum; +import org.apache.lucene.util.BytesRef; +import org.apache.lucene.util.PriorityQueue; + + +final class CompositePostingsEnum extends PostingsEnum { + + private static final int NO_MORE_POSITIONS = -2; + private final BytesRef term; + private final int freq; + private final PriorityQueue<BoundsCheckingPostingsEnum> queue; + + + /** + * This class is used to ensure we don't over iterate the underlying + * postings enum by keeping track of the position relative to the + * frequency. + * Ideally this would've been an implementation of a PostingsEnum + * but it would have to delegate most methods and it seemed easier + * to just wrap the tweaked method. + */ + private static final class BoundsCheckingPostingsEnum { + + + private final PostingsEnum postingsEnum; + private final int freq; + private int position; + private int nextPosition; + private int positionInc = 1; + + private int startOffset; + private int endOffset; + + BoundsCheckingPostingsEnum(PostingsEnum postingsEnum) throws IOException { + this.postingsEnum = postingsEnum; + this.freq = postingsEnum.freq(); + nextPosition = postingsEnum.nextPosition(); + position = nextPosition; + startOffset = postingsEnum.startOffset(); + endOffset = postingsEnum.endOffset(); + } + + private boolean hasMorePositions() throws IOException { + return positionInc < freq; + } + + /** + * Returns the next position of the underlying postings enum unless + * it cannot iterate further and returns NO_MORE_POSITIONS; + * @return + * @throws IOException + */ + private int nextPosition() throws IOException { + position = nextPosition; + startOffset = postingsEnum.startOffset(); + endOffset = postingsEnum.endOffset(); + if (hasMorePositions()) { + positionInc++; + nextPosition = postingsEnum.nextPosition(); + } else { + nextPosition = NO_MORE_POSITIONS; + } + return position; + } + + } + + CompositePostingsEnum(BytesRef term, List<PostingsEnum> postingsEnums) throws IOException { + this.term = term; + queue = new PriorityQueue<BoundsCheckingPostingsEnum>(postingsEnums.size()) { + @Override + protected boolean lessThan(BoundsCheckingPostingsEnum a, BoundsCheckingPostingsEnum b) { + return a.position < b.position; + } + }; + + int freqAdd = 0; + for (PostingsEnum postingsEnum : postingsEnums) { + queue.add(new BoundsCheckingPostingsEnum(postingsEnum)); + freqAdd += postingsEnum.freq(); + } + freq

= freqAdd; + } + + @Override + public int freq() throws IOException { + return freq; + } + + @Override + public int nextPosition() throws IOException { + int position = NO_MORE_POSITIONS; + while (queue.size() >= 1) { + queue.top().nextPosition(); + queue.updateTop(); //the new position may be behind another postingsEnum in the queue + position = queue.top().position; + + if (position == NO_MORE_POSITIONS) { + queue.pop(); //this postingsEnum is consumed, let's get rid of it + } else { + break; //we got a new position + } + + } + return position; + } + + @Override + public int startOffset() throws IOException { + return queue.top().startOffset; + } + + @Override + public int endOffset() throws IOException { + return queue.top().endOffset; + } + + @Override + public BytesRef getPayload() throws IOException { + //The UnifiedHighlighter depends on the payload for a wildcard + //being the term representing it + return term; + } + + @Override + public int docID() { + return queue.top().postingsEnum.docID(); --- End diff -- I think this one should be held in a field of CompositePostingsEnum initialized in constructor? Not a big deal though, since I don't think the highlighter is going to call this, let alone sometime after positions are exhausted (which would trigger an NPE in this case).

16. It seems the ASF's GitHub -> JIRA comment integration doesn't incorporate overall review comments (i.e. comments not attached to a line number). Please look in GH for that. After a round of edits or two, I'll run the benchmarks to see how this fairs. I expect it'll be a little slower for the analysis case when no phrases are present due to the change of default to prefer MemoryIndex for the benefit of better passage relevancy (now a toggle'able option). Hopefully very little. If the difference seems imperceptible then there's a case for dropping the TokenStreamOffsetStrategy along with some quirks related to it, which would be nice in terms of code maintenance.
17. Github user Timothy055 commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85611673 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/CompositePostingsEnum.java --- @@ -0,0 +1,165 @@ +/* + * 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. + */ +package org.apache.lucene.search.uhighlight; + +import java.io.IOException; +import java.util.List; + +import org.apache.lucene.index.PostingsEnum; +import org.apache.lucene.util.BytesRef; +import org.apache.lucene.util.PriorityQueue; + + +final class CompositePostingsEnum extends PostingsEnum { + + private static final int NO_MORE_POSITIONS = -2; + private final BytesRef term; + private final int freq; + private final PriorityQueue<BoundsCheckingPostingsEnum> queue; + + + /** + * This class is used to ensure we don't over iterate the underlying + * postings enum by keeping track of the position relative to the + * frequency. + * Ideally this would've been an implementation of a PostingsEnum + * but it would have to delegate most methods and it seemed easier + * to just wrap the tweaked method. + */ + private static final class BoundsCheckingPostingsEnum { + + + private final PostingsEnum postingsEnum; + private final int freq; + private int position; + private int nextPosition; + private int positionInc = 1; + + private int startOffset; --- End diff -- I thought the same, but unfortunately there's no way to check the current position of the underlying PostingsEnum before calling nextPosition. Since we call nextPosition the PostingsEnum has moved on before the highlighter calls startOffset or endOffset.
18. Github user Timothy055 commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85611978 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/CompositePostingsEnum.java --- @@ -0,0 +1,165 @@ +/* + * 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. + */ +package org.apache.lucene.search.uhighlight; + +import java.io.IOException; +import java.util.List; + +import org.apache.lucene.index.PostingsEnum; +import org.apache.lucene.util.BytesRef; +import org.apache.lucene.util.PriorityQueue; + + +final class CompositePostingsEnum extends PostingsEnum { + + private static final int NO_MORE_POSITIONS = -2; + private final BytesRef term; + private final int freq; + private final PriorityQueue<BoundsCheckingPostingsEnum> queue; + + + /** + * This class is used to ensure we don't over iterate the underlying + * postings enum by keeping track of the position relative to the + * frequency. + * Ideally this would've been an implementation of a PostingsEnum + * but it would have to

delegate most methods and it seemed easier + * to just wrap the tweaked method. + */ + private static final class BoundsCheckingPostingsEnum { + + + private final PostingsEnum postingsEnum; + private final int freq; + private int position; + private int nextPosition; + private int positionInc = 1; + + private int startOffset; + private int endOffset; + + BoundsCheckingPostingsEnum(PostingsEnum postingsEnum) throws IOException { + this.postingsEnum = postingsEnum; + this.freq = postingsEnum.freq(); + nextPosition = postingsEnum.nextPosition(); + position = nextPosition; + startOffset = postingsEnum.startOffset(); + endOffset = postingsEnum.endOffset(); + } + + private boolean hasMorePositions() throws IOException { + return positionInc < freq; + } + + /** + * Returns the next position of the underlying postings enum unless + * it cannot iterate further and returns NO_MORE_POSITIONS; + * @return + * @throws IOException + */ + private int nextPosition() throws IOException { + position = nextPosition; + startOffset = postingsEnum.startOffset(); + endOffset = postingsEnum.endOffset(); + if (hasMorePositions()) { + positionInc++; + nextPosition = postingsEnum.nextPosition(); + } else { + nextPosition = NO_MORE_POSITIONS; + } + return position; + } + + } + + CompositePostingsEnum(BytesRef term, List<PostingsEnum> postingsEnums) throws IOException { + this.term = term; + queue = new PriorityQueue<BoundsCheckingPostingsEnum>(postingsEnums.size()) { + @Override + protected boolean lessThan(BoundsCheckingPostingsEnum a, BoundsCheckingPostingsEnum b) { + return a.position < b.position; + } + }; + + int freqAdd = 0; + for (PostingsEnum postingsEnum : postingsEnums) { + queue.add(new BoundsCheckingPostingsEnum(postingsEnum)); + freqAdd += postingsEnum.freq(); + } + freq = freqAdd; + } + + @Override + public int freq() throws IOException { + return freq; + } + + @Override + public int nextPosition() throws IOException { + int position = NO_MORE_POSITIONS; + while (queue.size() >= 1) { + queue.top().nextPosition(); + queue.updateTop(); //the new position may be behind another postingsEnum in the queue + position = queue.top().position; + + if (position == NO_MORE_POSITIONS) { + queue.pop(); //this postingsEnum is consumed, let's get rid of it + } else { + break; //we got a new position + } + + } + return position; + } + + @Override + public int startOffset() throws IOException { + return queue.top().startOffset; + } + + @Override + public int endOffset() throws IOException { + return queue.top().endOffset; + } + + @Override + public BytesRef getPayload() throws IOException { + //The UnifiedHighlighter depends on the payload for a wildcard + //being the term representing it + return term; + } + + @Override + public int docID() { + return queue.top().postingsEnum.docID(); --- End diff -- K

19. Github user Timothy055 commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85612011 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/AnalysisOffsetStrategy.java --- @@ -17,174 +17,28 @@ package org.apache.lucene.search.uhighlight; import java.io.IOException; -import java.util.ArrayList; -import java.util.Arrays; -import java.util.Collections; -import java.util.List; import org.apache.lucene.analysis.Analyzer; -import org.apache.lucene.analysis.FilteringTokenFilter; import org.apache.lucene.analysis.TokenStream; -import org.apache.lucene.analysis.tokenattributes.CharTermAttribute; -import org.apache.lucene.index.IndexReader; -import org.apache.lucene.index.LeafReader; -import org.apache.lucene.index.Terms; -import org.apache.lucene.index.memory.MemoryIndex; -import org.apache.lucene.search.spans.SpanQuery; import org.apache.lucene.util.BytesRef; -import org.apache.lucene.util.automaton.Automata; import org.apache.lucene.util.automaton.CharacterRunAutomaton; +public abstract class AnalysisOffsetStrategy extends FieldOffsetStrategy { --- End diff -- Thanks
20. Github user Timothy055 commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85613130 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/FieldOffsetStrategy.java --- @@ -65,58 +65,88 @@ public String getField() { */ public abstract List<OffsetsEnum> getOffsetsEnums(IndexReader reader, int docId, String content) throws IOException; - protected List<OffsetsEnum> createOffsetsEnums(LeafReader leafReader, int doc, TokenStream tokenStream) throws IOException { - List<OffsetsEnum> offsetsEnums = createOffsetsEnumsFromReader(leafReader, doc); - if (automata.length > 0) { - offsetsEnums.add(createOffsetsEnumFromTokenStream(doc, tokenStream)); + protected List<OffsetsEnum> createOffsetsEnumsFromReader(LeafReader leafReader, int doc) throws IOException { + final Terms termsIndex = leafReader.terms(field); + if (termsIndex == null) { + return Collections.emptyList(); } - return offsetsEnums; - } - protected List<OffsetsEnum> createOffsetsEnumsFromReader(LeafReader atomicReader, int doc) throws IOException { // For strict positions, get a Map of term to Spans: // note: ScriptPhraseHelper.NONE does the right thing for these method calls final Map<BytesRef, Spans> strictPhrasesTermToSpans = - strictPhrases.getTermToSpans(atomicReader, doc); + phraseHelper.getTermToSpans(leafReader, doc); // Usually simply wraps terms in a List; but if willRewrite() then can be expanded final List<BytesRef> sourceTerms = - strictPhrases.expandTermsIfRewrite(terms, strictPhrasesTermToSpans); + phraseHelper.expandTermsIfRewrite(terms, strictPhrasesTermToSpans); - final List<OffsetsEnum> offsetsEnums = new ArrayList<>(sourceTerms.size() + 1); + final List<OffsetsEnum> offsetsEnums = new ArrayList<>(sourceTerms.size() + automata.length); - Terms termsIndex = atomicReader == null || sourceTerms.isEmpty() ? null : atomicReader.terms(field); - if (termsIndex != null) { + // Handle sourceTerms: + if (!sourceTerms.isEmpty()) { TermsEnum termsEnum = termsIndex.iterator();//does not return

null for (BytesRef term : sourceTerms) { - if (!termsEnum.seekExact(term)) { - continue; // term not found - } - PostingsEnum postingsEnum = termsEnum.postings(null, PostingsEnum.OFFSETS); - if (postingsEnum == null) { - // no offsets or positions available - throw new IllegalArgumentException("field '" + field + "' was indexed without offsets, cannot highlight"); - } - if (doc != postingsEnum.advance(doc)) { // now it's positioned, although may be exhausted - continue; + if (termsEnum.seekExact(term)) { + PostingsEnum postingsEnum = termsEnum.postings(null, PostingsEnum.OFFSETS); + + if (postingsEnum == null) { + // no offsets or positions available + throw new IllegalArgumentException("field '" + field + "' was indexed without offsets, cannot highlight"); + } + + if (doc == postingsEnum.advance(doc)) { // now it's positioned, although may be exhausted + postingsEnum = phraseHelper.filterPostings(term, postingsEnum, strictPhrasesTermToSpans.get(term)); + if (postingsEnum != null) { + offsetsEnums.add(new OffsetsEnum(term, postingsEnum)); + } + } } - postingsEnum = strictPhrases.filterPostings(term, postingsEnum, strictPhrasesTermToSpans.get(term)); - if (postingsEnum == null) { - continue;// completely filtered out + } + } + + // Handle automata + if (automata.length > 0) { + offsetsEnums.addAll(createAutomataOffsetsFromTerms(termsIndex, doc)); + } + + return offsetsEnums; + } + + protected List<OffsetsEnum> createAutomataOffsetsFromTerms(Terms termsIndex, int doc) throws IOException { + Map<CharacterRunAutomaton, List<PostingsEnum>> automataPostings = new IdentityHashMap<>(automata.length); --- End diff -- One minor problem with that was that the code would not longer by type-safe because of the lack of generic arrays in java. I wouldn't be able to do `List<PostingsEnum>[]` = new ArrayList<PostingsEnum>[automata.length];` but could do `List<PostingsEnum>[]` = new ArrayList[automata.length];` with unchecked casts. Seem worth it?

21. Github user Timothy055 commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85613814 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/FieldOffsetStrategy.java --- @@ -65,58 +65,88 @@ public String getField() { */ public abstract List<OffsetsEnum> getOffsetsEnums(IndexReader reader, int docId, String content) throws IOException; - protected List<OffsetsEnum> createOffsetsEnums(LeafReader leafReader, int doc, TokenStream tokenStream) throws IOException { - List<OffsetsEnum> offsetsEnums = createOffsetsEnumsFromReader(leafReader, doc); - if (automata.length > 0) { - offsetsEnums.add(createOffsetsEnumFromTokenStream(doc, tokenStream)); + protected List<OffsetsEnum> createOffsetsEnumsFromReader(LeafReader leafReader, int doc) throws IOException { + final Terms termsIndex = leafReader.terms(field); + if (termsIndex == null) { + return Collections.emptyList(); } - return offsetsEnums; - } - protected List<OffsetsEnum> createOffsetsEnumsFromReader(LeafReader atomicReader, int doc) throws IOException { // For strict positions, get a Map of term to Spans: // note: ScriptPhraseHelper.NONE does the right thing for these method calls final Map<BytesRef, Spans> strictPhrasesTermToSpans = - strictPhrases.getTermToSpans(atomicReader, doc); + phraseHelper.getTermToSpans(leafReader, doc); // Usually simply wraps terms in a List; but if willRewrite() then can be expanded final List<BytesRef> sourceTerms = - strictPhrases.expandTermsIfRewrite(terms, strictPhrasesTermToSpans); + phraseHelper.expandTermsIfRewrite(terms, strictPhrasesTermToSpans); - final List<OffsetsEnum> offsetsEnums = new ArrayList<>(sourceTerms.size() + 1); + final List<OffsetsEnum> offsetsEnums = new ArrayList<>(sourceTerms.size() + automata.length); - Terms termsIndex = atomicReader == null || sourceTerms.isEmpty() ? null : atomicReader.terms(field); - if (termsIndex != null) { + // Handle sourceTerms: + if (!sourceTerms.isEmpty()) { TermsEnum termsEnum = termsIndex.iterator();//does not return null for (BytesRef term : sourceTerms) { - if (!termsEnum.seekExact(term)) { - continue; // term not found - } - PostingsEnum postingsEnum = termsEnum.postings(null, PostingsEnum.OFFSETS); - if (postingsEnum == null) { - // no offsets or positions available - throw new IllegalArgumentException("field '" + field + "' was indexed without offsets, cannot highlight"); - } - if (doc != postingsEnum.advance(doc)) { // now it's positioned, although may be exhausted - continue; + if (termsEnum.seekExact(term)) { + PostingsEnum postingsEnum = termsEnum.postings(null, PostingsEnum.OFFSETS); + + if (postingsEnum == null) { + // no offsets or positions available + throw new IllegalArgumentException("field '" + field + "' was indexed without offsets, cannot highlight"); + } + + if (doc == postingsEnum.advance(doc)) { // now it's positioned, although may be exhausted + postingsEnum = phraseHelper.filterPostings(term, postingsEnum, strictPhrasesTermToSpans.get(term)); + if (postingsEnum != null) { + offsetsEnums.add(new OffsetsEnum(term, postingsEnum)); + } + } } - postingsEnum = strictPhrases.filterPostings(term, postingsEnum, strictPhrasesTermToSpans.get(term)); - if (postingsEnum == null) { - continue;// completely filtered out + } + } + + // Handle automata + if (automata.length > 0) { + offsetsEnums.addAll(createAutomataOffsetsFromTerms(termsIndex, doc)); + } + + return offsetsEnums; + } + + protected List<OffsetsEnum> createAutomataOffsetsFromTerms(Terms termsIndex, int doc) throws IOException { + Map<CharacterRunAutomaton, List<PostingsEnum>> automataPostings = new IdentityHashMap<>(automata.length); --- End diff -- How about List<List<PostingsEnum>> that should give better locality without lose of type safety?
22. Github user Timothy055 commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85614277 --- Diff: lucene/highlighter/src/test/org/apache/lucene/search/uhighlight/visibility/TestUnifiedHighlighterExtensibility.java

--- @@ -79,7 +90,7 @@ public void testFieldOffsetStrategyExtensibility() { @Test public void testUnifiedHighlighterExtensibility() { final int maxLength = 1000; - UnifiedHighlighter uh = new UnifiedHighlighter(null, new MockAnalyzer(random())){ + UnifiedHighlighter uh = new UnifiedHighlighter(null, new MockAnalyzer(new Random())){ --- End diff -- fixed

23. Github user Timothy055 commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85616651 --- Diff: lucene/highlighter/src/test/org/apache/lucene/search/uhighlight/visibility/TestUnifiedHighlighterExtensibility.java --- @@ -79,7 +90,7 @@ public void testFieldOffsetStrategyExtensibility() { @Test public void testUnifiedHighlighterExtensibility() { final int maxLength = 1000; - UnifiedHighlighter uh = new UnifiedHighlighter(null, new MockAnalyzer(random())){ + UnifiedHighlighter uh = new UnifiedHighlighter(null, new MockAnalyzer(new Random())){ --- End diff -- I recall now why I changed it. I started getting this error and figured it was a change elsewhere: java.lang.IllegalStateException: No context information for thread: Thread[id=1, name=main, state=RUNNABLE, group=main]. Is this thread running under a class com.carrotsearch.randomizedtesting.RandomizedRunner runner context? Add @RunWith(class com.carrotsearch.randomizedtesting.RandomizedRunner.class) to your test class. Make sure your code accesses random contexts within @BeforeClass and @AfterClass boundary (for example, static test class initializers are not permitted to access random contexts).
24. Github user Timothy055 commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85616894 --- Diff: lucene/highlighter/src/test/org/apache/lucene/search/uhighlight/visibility/TestUnifiedHighlighterExtensibility.java --- @@ -79,7 +90,7 @@ public void testFieldOffsetStrategyExtensibility() { @Test public void testUnifiedHighlighterExtensibility() { final int maxLength = 1000; - UnifiedHighlighter uh = new UnifiedHighlighter(null, new MockAnalyzer(random())){ + UnifiedHighlighter uh = new UnifiedHighlighter(null, new MockAnalyzer(new Random())){ --- End diff -- I'm a bit confused. What I should I change?
25. Github user Timothy055 commented on the issue: https://github.com/apache/lucene-solr/pull/105 I've pushed some more changes now. Still taking a look at what we might be able to do further with CompositePostingsEnum
26. Github user Timothy055 commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85618489 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/CompositePostingsEnum.java --- @@ -0,0 +1,165 @@ +/* + * 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. + */ +package org.apache.lucene.search.uhighlight; + +import java.io.IOException; +import java.util.List; + +import org.apache.lucene.index.PostingsEnum; +import org.apache.lucene.util.BytesRef; +import org.apache.lucene.util.PriorityQueue; + + +final class CompositePostingsEnum extends PostingsEnum { + + private static final int NO_MORE_POSITIONS = -2; + private final BytesRef term; + private final int freq; + private final PriorityQueue<BoundsCheckingPostingsEnum> queue; + + + /** + * This class is used to ensure we don't over iterate the underlying + * postings enum by keeping track of the position relative to the + * frequency. + * Ideally this would've been an implementation of a PostingsEnum + * but it would have to delegate most methods and it seemed easier + * to just wrap the tweaked method. + */ + private static final class BoundsCheckingPostingsEnum { + + + private final PostingsEnum postingsEnum; + private final int freq; --- End diff -- Hmm, did you mean to calculate freq dynamically in the method and just use a remainingPositions count as walking over the postings? Instead of re-computing it's cached in the constructor, but if we modified it, the freq count would change as the postings were iterated.
27. Github user Timothy055 commented on a diff in the pull request: https://github.com/apache/lucene-solr/pull/105#discussion_r85619786 --- Diff: lucene/highlighter/src/java/org/apache/lucene/search/uhighlight/FieldOffsetStrategy.java --- @@ -65,58 +65,88 @@ public String getField() { */ public abstract List<OffsetsEnum> getOffsetsEnums(IndexReader reader, int docId, String content) throws IOException; - protected List<OffsetsEnum> createOffsetsEnums(LeafReader leafReader, int doc, TokenStream tokenStream) throws IOException { - List<OffsetsEnum> offsetsEnums = createOffsetsEnumsFromReader(leafReader, doc); - if (automata.length > 0) { - offsetsEnums.add(createOffsetsEnumFromTokenStream(doc, tokenStream)); + protected List<OffsetsEnum> createOffsetsEnumsFromReader(LeafReader leafReader, int doc) throws IOException { + final Terms termsIndex = leafReader.terms(field); + if (termsIndex == null) { + return Collections.emptyList(); } - return offsetsEnums; - } - protected List<OffsetsEnum> createOffsetsEnumsFromReader(LeafReader atomicReader,

int doc) throws IOException { // For strict positions, get a Map of term to Spans: // note: ScriptPhraseHelper.NONE does the right thing for these method calls final Map<BytesRef, Spans> strictPhrasesTermToSpans = - strictPhrases.getTermToSpans(atomicReader, doc); + phraseHelper.getTermToSpans(leafReader, doc); // Usually simply wraps terms in a List; but if willRewrite() then can be expanded final List<BytesRef> sourceTerms = - strictPhrases.expandTermsIfRewrite(terms, strictPhrasesTermToSpans); + phraseHelper.expandTermsIfRewrite(terms, strictPhrasesTermToSpans); - final List<OffsetsEnum> offsetsEnums = new ArrayList<>(sourceTerms.size() + 1); + final List<OffsetsEnum> offsetsEnums = new ArrayList<>(sourceTerms.size() + automata.length); - Terms termsIndex = atomicReader == null || sourceTerms.isEmpty() ? null : atomicReader.terms(field); - if (termsIndex != null) { + // Handle sourceTerms: + if (!sourceTerms.isEmpty()) { TermsEnum termsEnum = termsIndex.iterator();//does not return null for (BytesRef term : sourceTerms) { - if (!termsEnum.seekExact(term)) { - continue; // term not found - } - PostingsEnum postingsEnum = termsEnum.postings(null, PostingsEnum.OFFSETS); - if (postingsEnum == null) { - // no offsets or positions available - throw new IllegalArgumentException("field '" + field + "' was indexed without offsets, cannot highlight"); - } - if (doc != postingsEnum.advance(doc)) { // now it's positioned, although may be exhausted - continue; + if (termsEnum.seekExact(term)) { + PostingsEnum postingsEnum = termsEnum.postings(null, PostingsEnum.OFFSETS); + + if (postingsEnum == null) { + // no offsets or positions available + throw new IllegalArgumentException("field '" + field + "' was indexed without offsets, cannot highlight"); + } + + if (doc == postingsEnum.advance(doc)) { // now it's positioned, although may be exhausted + postingsEnum = phraseHelper.filterPostings(term, postingsEnum, strictPhrasesTermToSpans.get(term)); + if (postingsEnum != null) { + offsetsEnums.add(new OffsetsEnum(term, postingsEnum)); + } + } } } - postingsEnum = strictPhrases.filterPostings(term, postingsEnum, strictPhrasesTermToSpans.get(term)); - if (postingsEnum == null) { - continue;// completely filtered out + } + } + + // Handle automata + if (automata.length > 0) { + offsetsEnums.addAll(createAutomataOffsetsFromTerms(termsIndex, doc)); + } + + return offsetsEnums; + } + + protected List<OffsetsEnum> createAutomataOffsetsFromTerms(Terms termsIndex, int doc) throws IOException { + Map<CharacterRunAutomaton, List<PostingsEnum>> automataPostings = new IdentityHashMap<>(automata.length); --- End diff -- Pushed that for now to see what you think.

28. Github user Timothy055 commented on the issue: https://github.com/apache/lucene-solr/pull/105 I don't think there's a way to avoid keeping the position state, unfortunately. The reason is that we can move one of the postings enums to the next position, but then realize the next position for that term is behind the position for a different term (and postings enum) that also matches the wildcard. Then we'll update the top and switch to the next postings enum (by offset now), but once it's exhausted or we switch back to the previous one from interleaving the position is lost. :/ An alternative to avoid this would be to change PostingsEnum to allow fetching of the currentPosition, then nearly all the house keeping would go away.
29. Github user dsmiley commented on the issue: https://github.com/apache/lucene-solr/pull/105 I started playing with a bit and realized the same thing. It looks straight-forward but it's deceptively more complicated. Then it hit me -- lets not try to return the correct position at all! A "normal" PostingsEnum should but this one is _only_ used for offsets. So always return -1 -- we can get away with it for this internal use.
30. Github user Timothy055 commented on the issue: https://github.com/apache/lucene-solr/pull/105 Hmm, clever! But not sure I find it very clean though. I feel like that can lead to trouble down the road if code ever expects the offsets to be ordered. If we went that route we wouldn't even need the priority queue though. Btw, I MultiTermHighlighting is nearly gone except for one method that is used in the UnifiedHighlighter and MemoryIndexOffsetStrategy for extracting automata from a query. Any ideas on good ways to move it? Perhaps the UnifiedHighlighter should do all automata extraction and pass that in?
31. Github user dsmiley commented on the issue: https://github.com/apache/lucene-solr/pull/105 Check it out: https://github.com/dsmiley/lucene-solr/commit/50e2ea89c7eb15c863aa6e04e14fd32085ee85bd Remember this is a package-private class internal to the UnifiedHighlighter. Not completely implementing an interface if the caller won't care is okay. RE MultiTermHighlighting: I think it's fine as-is. The UH class is very long to adopt this -- too long as it is IMO.
32. Github user Timothy055 commented on the issue: https://github.com/apache/lucene-solr/pull/105 Hmm, I know we're already knowingly breaking the PostingsEnum contract, but rather than throwing a IllegaStateException, maybe we could still return positions, they just happen to be unordered, that way when the final position is exhausted we return -1 as the sentinel value still? Otherwise, I'm good with this, I'll merge in. Btw, we've seen other needs for something like a CompositePostingsEnum that abstracts over a set of terms, but since this is still internal, dropping the house-keeping will also make this code do less. We can always try to cross that bridge later.
33. Github user Timothy055 commented on the issue: https://github.com/apache/lucene-solr/pull/105 Forget it, you can't do that because the next position on one enum might be -1, but there's more enums left in the queue so the user of this class could inadvertently terminate early if looking for -1.
34. Github user Timothy055 commented on the issue: https://github.com/apache/lucene-solr/pull/105 Merged your commit.

35. Github user dsmiley commented on the issue: https://github.com/apache/lucene-solr/pull/105 _(I wrote this as you made your last comments, but rather than delete it I'll comment any way)_ The documentation for `PostingsEnum.nextPosition()` states that calling it more than `freq()` times is undefined. Thus it's quite valid to throw an IllegalStateException. > Btw, we've seen other needs for something like a CompositePostingsEnum that abstracts over a set of terms, but since this is still internal, dropping the house-keeping will also make this code do less. I don't think I quite get what you're saying. By "other needs" do you mean Bloomberg internally? If so, how would that relate this this one inside the UH? Are you advocating a general purpose Multi-PosrtingsEnum? On the latter... a highlighter wouldn't be where to introduce such a thing. There is a `org.apache.lucene.index.MultiPostingsEnum` which I looked at while at the Lucene hackday code sprint as it got my curiosity. Unfortunately, it doesn't seem quite general purpose enough for us to use -- it demands a MultiTermsEnum parent. Perhaps that could be improved to be useful without demanding a MultiTermsEnum parent... but that seems like a separate issue.
36. Github user Timothy055 commented on the issue: https://github.com/apache/lucene-solr/pull/105 Fair enough on both ends. And yes, we have seen a potential use case for a unified view over terms. I'll take a look at MultiPostingsEnum and see if we can use that.
37. Github user dsmiley commented on the issue: https://github.com/apache/lucene-solr/pull/105 Nevermind on `MultiPostingsEnum`. MPE assumes the PostingsEnum are for a disjoint set of documents -- i.e. each PostingsEnum matches documents disjoint from the others. This makes sense given it's use for a single view over multiple segments. In our case we want what might be called a MultiPositionsPostingsEnum or MultiOffsetsPostingsEnum or "Composite" instead of "Multi" (synonymous, really). We merge the positions (yet we don't care about the actual position #s; it's the other stuff we expose per-position). BTW my code you merged had a "nocommit" to consider changing the class name.
38. Github user Timothy055 commented on the issue: https://github.com/apache/lucene-solr/pull/105 Ah, missed that. Fixed
39. I've merged with the changes from LUCENE-7544 and also ran some benchmarks. (Thanks [~dsmiley] for the fix on LUCENE-7546!) Original: ||Impl||Terms||Phrases||Wildcards|| |(search)|1.14|1.43|2.44| |SH_A|7.36|7.49|16.37| |UH_A|5.32|4.55|9.24| |SH_V|4.12|4.42|8.47| |FVH_V|3.46|2.98|7.13| |UH_V|3.7|3.45|6.61| |PH_P|3.76|3.45|9.6| |UH_P|3.34|2.91|9.33| |UH_PV|3.26|2.8|6.72| With improvements from LUCENE-7526: ||Impl||Terms||Phrases||Wildcards|| |(search)|1.18|1.38|2.52| |SH_A|7.98|7.53|16.62| |UH_A|5.46|4.6|9.43| |SH_V|4.13|4.42|8.26| |FVH_V|3.45|3.05|6.93| |UH_V|3.79|3.43|6.62| |PH_P|3.82|3.47|9.4| |UH_P|3.33|3.03|9.46| |UH_PV|3.24|2.81|6.92| If you disable the new option to prefer passage relevancy over speed you'll get the following for analysis: ||Impl||Terms||Phrases||Wildcards|| |(search)|1.1|1.43|2.44| |UH_A|5.31|4.66|9.14| I wasn't able to get very consistent times with the benchmarks, but it looks like the changes keep close performance while simplifying the code and improving relevancy in the Analysis case (unless preferPassageRelevancyOverSpeed is disabled). If that option is disabled the timings line up pretty closely with the originals, providing a minor speed boost. There should also be a memory savings by avoiding re-creation of TokenStreams, but that was difficult to measure, but could prove beneficial if there is memory pressure. I performed these benchmark on a machine with the following configuration: Processor: AMD Phenom II X4 960T 3.0GHz Memory: 24GB DDR3 Disk: Crucial CT256MX SSD OS: Windows 10 Java: Java HotSpot(TM) 64-Bit Server VM (build 25.25-b02, mixed mode) All versions of the benchmarks incorporated above included the changes from LUCENE-7544. [~dsmiley] It looks like my older processor took significantly longer to highlight across the board than in your initial run for LUCENE-7438. I'd be curious how this set of changes performs on your machine now.
40. Other than that, I think this code is in good shape for committing.
41. RE Benchmarks: Yeah it seems hard to get consistency. I even upped the iterations counts to 2000 (from 500) and saw more variability than I'd like. Nonetheless your analysis looks good to me; it's how I would sum up my observations today. I ran my benchmarks with "-Xms4G -Xmx4G -XX:NewRatio=1" (to reduce the effects of GC). My machine is a MacBook Pro Retina "Late 2013", 2 GHz i7, 8GB RAM. This index is on an external spinning disk but there is no disk activity after warm-up because it's all in the O.S. Cache. As I looked over things carefully, I made a few more changes; all pretty minor. A few were in reaction to "ant precommit". https://github.com/dsmiley/lucene-solr/commits/uh_Tim -- the test & some javadoc issues. I also did a little bit of refactoring I hope you'll be good with. In a couple cases I merely moved a method up or down so that the code flows top to bottom better. Regarding Passage: I'll create a separate issue expressly for making Passage usable by anyone customizing the highlighter. It's not as simple as addMatch being public; there are other methods. _What's there now is in good shape for committing._ A couple ideas occurred to me; _feel free to punt to another issue or never_: * MultiValueTokenStream isn't needed for MemoryIndexOffsetStrategy, albeit with a change to loop over content.split(separatorChar). MemoryIndex.addField is overloaded to take the position increment gap. Then, MultiValueTokenStream could move to an inner class of TokenStreamOffsetStrategy, and it wouldn't generally be used (as it's no longer by default). That'd be nice -- keeping the complexity over there, and it's a bit of a hack too. * We had made OffsetsEnum & TokenStreamPostingsEnum implement Closeable to ameliorate the ramifications of the text analysis code throwing an exception, i.e. due to a bug. The only beneficiary of this now is TokenStreamOffsetStrategy, which isn't the default anymore. It could be removed to

simplify things. But then again, perhaps it could be useful for those implementing custom OffsetStrategies. I guess it should stay; there's very little to this after all. Proposed CHANGES.txt in "Improvements": * Enhanced UnifiedHighlighter's passage relevancy for queries with wildcards and sometimes just terms. Added shouldPreferPassageRelevancyOverSpeed() which can be overridden to return false to eek out more speed in some cases. (Tim Rodriguez, David Smiley)

42. Added the proposed changes. I'm on the fence around the refactor for MultiValueTokenStream, I'd much prefer to get rid of it completely if we could. But for now having some symmetry between the two impls seems worthwhile to me? I'd like to punt on that one.
43. I understand RE MultiValueTokenStream. I was motivated by considering highlighting a TokenStream without a MemoryIndex resulted in some complexity in various places (not introduced here, I mean pre-existing); some of it is addressed with this patch, and others still linger. Another hack that could move is the getPayoad() returning the Term. If this offset strategy returned a subclassed OffsetEnum, that quirk could be better isolated. Any way; let that be for another day. Attached is the patch delta from master. I'll commit tomorrow night.
44. Commit 7af454ad767c3a0364757d6fcf55bff9f063febe in lucene-solr's branch refs/heads/master from [~dsmiley] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=7af454a] LUCENE-7526: UnifiedHighlighter: enhance MTQ passage relevancy. TokenStreamFromTermVector isn't used by the UH anymore. Refactor AnalysisOffsetStrategy into TokenStream and MemoryIndex strategies, and related refactorings from that.
45. Commit 0790c34cc555ec9aaaab09cae04da6e61f465cfc in lucene-solr's branch refs/heads/branch_6x from [~dsmiley] [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=0790c34] LUCENE-7526: UnifiedHighlighter: enhance MTQ passage relevancy. TokenStreamFromTermVector isn't used by the UH anymore. Refactor AnalysisOffsetStrategy into TokenStream and MemoryIndex strategies, and related refactorings from that. (cherry picked from commit 7af454a)
46. Committed. Thanks Tim and thanks [~mbraun688] for some internal code reviewing.
47. Github user Timothy055 closed the pull request at: https://github.com/apache/lucene-solr/pull/105