Item 72
git_comments:

1. overflow
2. * * Create a new ExtendedIntervalIterator * @param in the iterator to wrap * @param before the number of positions to extend before the delegated interval * @param after the number of positions to extend beyond the delegated interval
3. * Licensed to the Apache Software Foundation (ASF) under one or more * contributor license agreements. See the NOTICE file distributed with * this work for additional information regarding copyright ownership. * The ASF licenses this file to You under the Apache License, Version 2.0 * (the "License"); you may not use this file except in compliance with * the License. You may obtain a copy of the License at * * http://www.apache.org/licenses/LICENSE-2.0 * * Unless required by applicable law or agreed to in writing, software * distributed under the License is distributed on an "AS IS" BASIS, * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. * See the License for the specific language governing permissions and * limitations under the License.
4. * * Wraps an IntervalIterator and extends the bounds of its intervals * * Useful for specifying gaps in an ordered iterator; if you want to match * `a b [2 spaces] c`, you can search for phrase(a, extended(b, 0, 2), c) * * An interval with prefix bounds extended by n will skip over matches that * appear in positions lower than n
5. * 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.
6. * * Create an {@link IntervalsSource} that wraps another source, extending its * intervals by a number of positions before and after. * * This can be useful for adding defined gaps in a block query; for example, * to find 'a b [2 arbitrary terms] c', you can call: * <pre> * Intervals.phrase(Intervals.term("a"), Intervals.extend(Intervals.term("b"), 0, 2), Intervals.term("c")); * </pre> * * Note that calling {@link IntervalIterator#gaps()} on iterators returned by this source * delegates directly to the wrapped iterator, and does not include the extensions. * * @param source the source to extend * @param before how many positions to extend before the delegated interval * @param after how many positions to extend after the delegated interval

git_commits:

1. **summary:** LUCENE-8612: Add Intervals.extend()
   message: LUCENE-8612: Add Intervals.extend()

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Add the ability to enforce gaps between intervals
   description: At the moment you can search for intervals with a maximum number of positions between them, but you cannot enforce gaps. It would be useful to be able to search for `a b [2 spaces] c`.
2. **summary:** Add the ability to enforce gaps between intervals
   description: At the moment you can search for intervals with a maximum number of positions between them, but you cannot enforce gaps. It would be useful to be able to search for `a b [2 spaces] c`.

3. **summary:** Add the ability to enforce gaps between intervals
   description: At the moment you can search for intervals with a maximum number of positions between them, but you cannot enforce gaps. It would be useful to be able to search for `a b [2 spaces] c`.
   label: code-design
4. **summary:** Add the ability to enforce gaps between intervals
   description: At the moment you can search for intervals with a maximum number of positions between them, but you cannot enforce gaps. It would be useful to be able to search for `a b [2 spaces] c`.
5. **summary:** Add the ability to enforce gaps between intervals
   description: At the moment you can search for intervals with a maximum number of positions between them, but you cannot enforce gaps. It would be useful to be able to search for `a b [2 spaces] c`.
6. **summary:** Add the ability to enforce gaps between intervals
   description: At the moment you can search for intervals with a maximum number of positions between them, but you cannot enforce gaps. It would be useful to be able to search for `a b [2 spaces] c`.
7. **summary:** Add the ability to enforce gaps between intervals
   description: At the moment you can search for intervals with a maximum number of positions between them, but you cannot enforce gaps. It would be useful to be able to search for `a b [2 spaces] c`.
8. **summary:** Add the ability to enforce gaps between intervals
   description: At the moment you can search for intervals with a maximum number of positions between them, but you cannot enforce gaps. It would be useful to be able to search for `a b [2 spaces] c`.
9. **summary:** Add the ability to enforce gaps between intervals
   description: At the moment you can search for intervals with a maximum number of positions between them, but you cannot enforce gaps. It would be useful to be able to search for `a b [2 spaces] c`.

jira_issues_comments:

1. Here is a patch adding {{Intervals.extend(source, before, after)}}, which allows you to make an interval cover extra positions before and after the actual terms. To search for {{a [space] b [space] c}}, for example, you could call {{Intervals.phrase(Intervals.term("a"), Intervals.extend(Intervals.term("b", 1, 1)), Intervals.term("c")}} The patch also clarifies the javadocs for how IntervalIterator should report start() and end() after the interval has been exhausted for the current document (important for the unordered algorithm), and fixes IntervalMatches to report start and end positions from its underlying IntervalIterator, rather than any wrapped Matches - for most iterators they are the same, but for ExtendedIntervalIterator we need to report the extended bounds so that phrase queries will still correctly report matches.
2. One thing I haven't added here but that we may want to think about is combining the 'before' count with a positional filter, so that a {{Intervals.extend(Intervals.term("b", 10, 0))}} won't match any instances of {{b}} that appear before position 10.
3. **body:** Updated patch, removing the specialized NotWithinFunction (no longer necessary), and adding the positional logic I described above, as I think it makes more sense to treat an extended interval as a 'block', and therefore requiring that prefix positions exist in the document.
   label: code-design
4. I removed the position filter part, as it ends up breaking notWithin in certain circumstances. Think this is ready to go.
5. Looks good. I'm wondering whether we should return NO_MORE_INTERVALS-1 if end() + after is equal to NO_MORE_INTERVALS?
6. > I'm wondering whether we should return NO_MORE_INTERVALS-1 if end() + after is equal to NO_MORE_INTERVALS? +1, I added a test
7. Commit 2532a5d31c8e4086120d74c8fb83102b8533777c in lucene-solr's branch refs/heads/master from Alan Woodward [https://git-wip-us.apache.org/repos/asf?p=lucene-solr.git;h=2532a5d] LUCENE-8612: Add Intervals.extend()