Item 59
git_comments:

1. * * Returns <code>true</code> iff the ref contains the given slice. Otherwise * <code>false</code>. * * @param ref * the {@link BytesRef} to test * @param slice * the slice to look for * @param ignoreCase * whether the comparison should be case-insensitive * @return Returns <code>true</code> iff the ref contains the given slice. * Otherwise <code>false</code>.
2. if mincount<=0 and we're not examining the values for contains, then we won't discard any terms and we know exactly where to start.
3. if facet.contains specified, only actually collect the count if substring contained
4. **comment:** :TODO:can we do contains earlier than this to make it more efficient?
   label: code-design
5. * * If using facet contains, ignore case when comparing values.
6. * * Only return constraints of a facet field containing the given string.

git_commits:

1. **summary:** SOLR-1387: Add facet.contains and facet.contains.ignoreCase
   message: SOLR-1387: Add facet.contains and facet.contains.ignoreCase git-svn-id: https://svn.apache.org/repos/asf/lucene/dev/trunk@1669335 13f79535-47bb-0310-9956-ffa450edef68

github_issues:

github_issues_comments:

github_pulls:

github_pulls_comments:

github_pulls_reviews:

jira_issues:

1. **summary:** Add more search options for filtering field facets.
   description: Currently for filtering the facets, we have to use prefix (which use String.startsWith() in java). We can add some parameters like * facet.iPrefix : this would act like case-insensitive search. (or ---> facet.prefix=a&facet.caseinsense=on) * facet.regex : this is pure regular expression search (which obviously would be expensive if issued). Moreover, allowing multiple filtering for same field would be great like facet.prefix=a OR facet.prefix=A ... sth like this. All above concepts could be equally applicable to TermsComponent.

jira_issues_comments:

1. {quote} facet.iPrefix : this would act like case-insensitive search. (or ---> facet.prefix=a&facet.caseinsense=on) {quote} I don't see a reason as to why the case filter be there. you can always apply a lower case filter to you field while indexing and searching. {quote} facet.regex : this is pure regular expression search (which obviously would be expensive if issued). {quote} You mean wildcards. Right? {quote} Moreover, allowing multiple filtering for same field would be great like facet.prefix=a OR facet.prefix=A ... sth like this. {quote} This has been recently discussed on the dev mailing list here - http://www.lucidimagination.com/search/document/f954dbb323746ed1/multiple_facet_prefix The syntax that was agreed upon was local params in this manner - facet.field={!prefix=foo prefix=bar}myfield
2. > I don't see a reason as to why the case filter be there. you can always apply a lower case filter to you field while indexing and searching. suppose i indexed a field called "placename" having name like California, Nevada, San Jose... If I use LowerCaseFilterFactory it will be stored in lowered case and when retrieving as FACET (or TermsComponent) it is also in lowered case. --> (california, nevada, san jose) And this will mess thing up (at least for me). I know there are others who want this too. > You mean wildcards. Right? Yes, it would be the first step towards it... [again i don't mean A* or abc*.., i would rather want *a or a*bc] > This has been recently discussed on the dev mailing list here - http://www.lucidimagination.com/search/document/f954dbb323746ed1/multiple_facet_prefix The syntax

that was agreed upon was local params in this manner - facet.field={!prefix=foo prefix=bar}myfield Yes this is what i'm talking about, having an option to get both the individual list and merge list for each query (here 'foo' and 'bar') would be better.

3. **body:** I've come up with following code. Any suggestions?? [This is just a code snippet] {code:title=Extension of SimpleFacet.java|borderStyle=solid} /*** SEARCHING ***/ // HashSet is choosen to avoid duplicate entry HashSet<String> termsDump = new HashSet<String>(); for (String term: terms) { //<------ terms[] from FieldCache.DEFAULT ... StringIndex.loopup if (term == null) continue; for (String p : iprefixList) { //<--- list of prefix to be search case insensitively. // doing iprefix if (term.toUpperCase().startsWith(p.toUpperCase())) { //<---- Is this the best way to do?? termsDump.add(term); } } for (String re: regexList) { // <--- list of regular expression if (term.matches(re)) { //equivalent to Pattern.compile(re).matcher(term).matches() termsDump.add(term); } } } // Just add the list of input terms without searching :) termsDump.addAll(inputTermsList); /*** COUNTING ***/ // <-- this counting method is different from regular prefix (finding spectrum in an array) FieldType ft = searcher.getSchema().getFieldType(field); NamedList<Integer> res = new NamedList(); Term t = new Term(field); for (String term : termList) { // <---- termList = termsDump from above String internal = ft.toInternal(term); int count = searcher.numDocs(new TermQuery(t.createTerm(internal)), base); // <--- Do we loose performance on this?? res.add(term, count); } /*** SORTING ***/ // <-- regular CountPair<String,Integer> thing. for (int i = 0, n= nList.size(); i <n; i++){ queue.add(new CountPair<String,Integer>(res.getName(i), res.getVal(i))); } {code} The syntax would look like (localParams style) this: {code} &facet.field={!XFilter=on prefix=A,B,C iPrefix=a,b,c,d termsList=e,f,g,h regex=^a[a-z0-9]+g$,z*}field_name {code} XFilter: i called this eXtended Filter for facet!!
   label: code-design
4. linking issues so we ensure they are considered in conjunction
5. Bulk updating 240 Solr issues to set the Fix Version to "next" per the process outlined in this email... http://mail-archives.apache.org/mod_mbox/lucene-dev/201005.mbox/%3Calpine.DEB.1.10.1005251052040.24672@radix.cryptio.net%3E Selection criteria was "Unresolved" with a Fix Version of 1.5, 1.6, 3.1, or 4.0. email notifications were suppressed. A unique token for finding these 240 issues in the future: hossversioncleanup20100527
6. Bulk move 3.2 -> 3.3
7. 3.4 -> 3.5
8. I like the idea, as I haven't found any solutions to this problem that are compatible with Sunspot (ruby solr interface). Just looking at your code, you may want to move some of the loop invariant stuff out of the loops. e.g. the downcasing of prefixes is the same every iteration, but you downcase it each time through. Same goes for term.uppercase, you could move it out one loop as it doesn't change within the prefix loop.
9. Bulk of fixVersion=3.6 -> fixVersion=4.0 for issues that have no assignee and have not been updated recently. email notification suppressed to prevent mass-spam psuedo-unique token identifying these issues: hoss20120321nofix36
10. Is there any way to advocate for this feature more?
11. [~josowski73] - vote for it if you haven't. bq. If I use LowerCaseFilterFactory it will be stored in lowered case and when retrieving as FACET (or TermsComponent) it is also in lowered case. --> (california, nevada, san jose) Is this really true? Won't the original string be preserved if stored="true"? Or is the indexed/lowercased value used?
12. Bulk move 4.4 issues to 4.5 and 5.0
13. **body:** {quote}I don't see a reason as to why the case filter be there. you can always apply a lower case filter to you field while indexing and searching.{quote} The reason is very well explained in [SolrFacetingOverview|http://wiki.apache.org/solr/SolrFacetingOverview#Facet_Indexing] - faceting is performed on the indexed values and they are returned. I can't show lowercased values to my users. A use case - we facet a multivalued field after an "fq" and get thousands of values. The user gets an infinite scrollable list through the values but we also want to let him search. Ideally a search for "states" should match "United States" which is not supported for two reasons: - term is not at the beginning of the indexed string, - term and indexed string cases do not match thus prefix filter does not help. A wildcard search (*states*) would help a lot. Regexp may be better but less performant. Any other ideas?
    label: code-design
14. [~kaukas], we had the same problems as you describe, can't show users downcased results, "prefix" must match any word, not just anchored at the beginning of the token. What we ended up doing is encoding more information in the tokenized values than just the result. We included the downcased search term and the term to display, but delimited them with a tab, e.g. "star wars Star Wars". Then in our app we grabbed

the last half and showed it to the user. As for getting a matching prefix on different words, e.g. "wars", we created multiple tokens where we chomped a word off each time. e.g. "star wars Star Wars", "wars Star Wars". Each has the same "display portion", but we now have full control over the "matching portion".
15. Move issue to Solr 4.9.
16. **body:** I've been looking at this issue from the use-case of autocompletion, and in this case it's very desirable to include completions from the middle of a word. I've developed a patch which adds the following faceting parameters: facet.contains - similar to facet.prefix, but the string supplied may appear anywhere in the term facet.contains.ignoreCase - a Boolean value; if true, the comparison is case insensitive The implementation for facet.contains has been done for the enum, fc, fcs and grouped faceting methods. The memory usage and performance is likely to be as 'bad' as for the same query without the facet.contains restriction (you lose the advantage of sorted values that can be leveraged in facet.prefix). The ignore-case is implemented in terms of UTF-8 case insensitivity so is also potentially computationally expensive.
    label: code-design
17. **body:** This looks great. Rather than using BytesRef.utf8ToString() in StringUtils.contains() (which can be expensive), can we use CharacterUtils.toLowerCase() instead? Have a look at LowercaseFilterFactory to see how that works. It would be nice to make ignoreCase more general, rather than only applying to facet.contains, but I guess it won't really apply cleanly to things like facet.prefix.
    label: code-design
18. As the name suggests, CharacterUtils works on a char[] whereas we have a BytesRef (essentially a byte[]). But I think CharacterUtils.toLowerCase() is doing essentially the same as I'm doing in StringHelper.contains() in that it converts using Unicode case mapping information (via Character.toLowerCase(int)). Yes, sadly making ignoreCase more general would spoil the efficiency of facet.prefix so I thought safest to leave as a sub-parameter of facet.contains, which spoils that efficiency already.
19. **body:** facet.contains would be great to have. Any general comments on the worst case performance? Does it approach the cost of reading all possible facet values for a field?
    label: code-design
20. Indeed, using facet.contains without facet.prefix means examining every value for a facet, and using ignoreCase in addition makes it even worse.
21. Initial testing using this patch against the 5.x branch looks pretty promising. Using facet.contains and facet.contains.ignoreCase on a multi-value field with tens of millions of unique values in an index of roughly 100 million documents isn't super fast (~3s), but is usable. Our other attempted solution was to pull back all facet values for filtering in the client, but that caused the cluster to hang. Other than vote this issue up, is there anything else we can do to help move this issue along?
22. I'll try and get it in for 5.0 (which is approaching rapidly).
23. Commit 1669335 from [~romseygeek] in branch 'dev/trunk' [https://svn.apache.org/r1669335] SOLR-1387: Add facet.contains and facet.contains.ignoreCase
24. Excellent, thanks for your work Will
25. Commit 1669336 from [~romseygeek] in branch 'dev/branches/branch_5x' [https://svn.apache.org/r1669336] SOLR-1387: Add facet.contains and facet.contains.ignoreCase
26. Closing this for now - I have some ideas for extending faceting using Automata which would mean we could add support for filtering facets by arbitrary regexes, but that can go in a separate issue, I think.
27. Thanks Alan!
28. Thanks for finally pushing it into the code. The 'contain' and 'contains.IgnoreCase' will cover most of the use-cases. I remember during that time, the code I wrote performed just fine (not terrible) for regular expression case. But mostly it was used for auto-completion that didn't use regex and worked pretty good. Directly using FSA (or FST), like in Lucene would be great for regex (and interesting project!) Thanks again guys.
29. **body:** I'm concerned about the StringHelper.contains that was added for this issue: * Its signature implies it operates on BytesRef, but under the hood it secretly assumes the bytes are valid UTF-8 (only for the ignoreCase=true case) * It also secretly assumes Locale.ENGLISH for downcasing but the incoming UTF-8 bytes may not be English * It has potentially poor performance compared to known algos e.g. http://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_string_search_algorithm Can we move this method out for now, e.g. not put it in the shared StringHelper utility class?
    label: code-design
30. bq. Can we move this method out for now, e.g. not put it in the shared StringHelper utility class? Sure, we could move it into the Solr faceting code. I'm away from a svn-accessible machine for 10 days or so now, I can do it when I get back or feel free to move it yourself.

31. Patch moving the 'contains' method to SimpleFacets, and refactoring it a bit to just use Strings. I'll commit this later today.
32. Thank you [~romseygeek] But the patch doesn't seem to actually remove the method from StringHelper? Or maybe you ran "svn diff" from inside solr subdir, so the patch is missing the lucene/ changes?
33. Oops, yes, that's exactly what I did. Here's the correct version...
34. Commit 1672106 from [~romseygeek] in branch 'dev/trunk' [https://svn.apache.org/r1672106] SOLR-1387: Move contains() method to SimpleFacets
35. Commit 1672112 from [~romseygeek] in branch 'dev/branches/branch_5x' [https://svn.apache.org/r1672112] SOLR-1387: Move contains() method to SimpleFacets
36. Commit 1672113 from [~romseygeek] in branch 'dev/branches/lucene_solr_5_1' [https://svn.apache.org/r1672113] SOLR-1387: Move contains() method to SimpleFacets
37. Bulk close after 5.1 release