Analyzing 750 billion events and 46 TB of code
What you can learn from GitHub's shared data on BigQuery

Felipe Hoffa
Developer Advocate

@felipehoffa

Google Cloud Platform
# Copyright 2015 The TensorFlow Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
# http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
"""Updates generated docs from Python doc comments."""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse
import collections
import os.path
import svs
# Copyright 2015 The TensorFlow Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License"); you may
# not use this file except in compliance with the License. You may obtain
# a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

"""Updates generated docs from Python doc comments."""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
import argparse
import collections
import os.path
import sys
''''Updates generated docs from Python doc comments.''''

from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse
import collections
import os.path
import sys
# Copyright 2015 The TensorFlow Authors. All Rights Reserved.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
# 
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

"""Updates generated docs from Python doc comments."""

from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse
import collections
import os.path
import sys
Copyright 2015 The TensorFlow Authors. All Rights Reserved.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

"""Updates generated docs from Python doc comments."""

from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
import argparse
import collections
import os.path
import sys
DATA

DATA

DATA
Who wants to analyze GitHub?
Project maintainers

- Popularity
- Who and how?
- Change management:
  - New APIs?
  - Breaking changes?
- Is my project healthy?
  - Issues closed on time?
  - Community participation?
Project users

- What other projects to follow?
- Requesting features
  - Data based requests
- Effective phrasing
Project choosers

- Is this project popular?
- Is this project healthy?
- Is this project well adopted?
- Related projects?
Data lovers

- Data integrators
- You
- Me :}
3 main datasets:

- GitHub Archive
  - 8.7 billion events
  - Hourly updates
- GHTorrent
  - These events annotated
  - Real-time updates
- GitHub repos on BigQuery
  - 46 TB of code
Google BigQuery

- Fast: terabytes in seconds
- Simple: SQL
- Scaleable: From bytes to petabytes
- No CAPEX: Always on
- Interoperable: Tableau, R, Python...
- Instant sharing
- Free monthly quota
Top projects by stars 2016?
```sql
#standardSQL
SELECT repo.name, COUNT(*) c
FROM `githubarchive.month.2016`
WHERE type='WatchEvent'
GROUP BY 1
ORDER BY c DESC
LIMIT 20
```

<table>
<thead>
<tr>
<th>Row</th>
<th>name</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FreeCodeCamp/FreeCodeCamp</td>
<td>185399</td>
</tr>
<tr>
<td>2</td>
<td>jwasham/google-interview-university</td>
<td>31496</td>
</tr>
<tr>
<td>3</td>
<td>vuejs/vue</td>
<td>29184</td>
</tr>
<tr>
<td>4</td>
<td>vhf/free-programming-books</td>
<td>29060</td>
</tr>
<tr>
<td>5</td>
<td>tensorflow/tensorflow</td>
<td>28634</td>
</tr>
<tr>
<td>6</td>
<td>facebook/react</td>
<td>26422</td>
</tr>
<tr>
<td>7</td>
<td>getify/You-Dont-Know-JS</td>
<td>25349</td>
</tr>
<tr>
<td>8</td>
<td>sindresorhus/awesome</td>
<td>25236</td>
</tr>
<tr>
<td>9</td>
<td>chrislgarry/Apollo-11</td>
<td>23633</td>
</tr>
<tr>
<td>10</td>
<td>yarnpkg/yarn</td>
<td>21487</td>
</tr>
</tbody>
</table>
Really?
```sql
#standardSQL
SELECT repo.name, COUNT(*) c, COUNT(DISTINCT actor.id) real_c
FROM `githubarchive.month.2016`
WHERE type='WatchEvent'
GROUP BY 1
ORDER BY real_c DESC
LIMIT 20
```

<table>
<thead>
<tr>
<th>Row</th>
<th>name</th>
<th>c</th>
<th>real_c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>FreeCodeCamp/FreeCodeCamp</td>
<td>185399</td>
<td>174984</td>
</tr>
<tr>
<td>2</td>
<td>jwasham/google-interview-university</td>
<td>31496</td>
<td>30973</td>
</tr>
<tr>
<td>3</td>
<td>vhf/free-programming-books</td>
<td>29060</td>
<td>27937</td>
</tr>
<tr>
<td>4</td>
<td>vuejs/vue</td>
<td>29184</td>
<td>27533</td>
</tr>
<tr>
<td>5</td>
<td>tensorflow/tensorflow</td>
<td>28634</td>
<td>27373</td>
</tr>
<tr>
<td>6</td>
<td>facebook/react</td>
<td>26422</td>
<td>24976</td>
</tr>
<tr>
<td>7</td>
<td>getify/You-Dont-Know-JS</td>
<td>25349</td>
<td>24466</td>
</tr>
<tr>
<td>8</td>
<td>sindresorhus/awesome</td>
<td>25236</td>
<td>24345</td>
</tr>
<tr>
<td>9</td>
<td>chrislgray/Apollo-11</td>
<td>23633</td>
<td>23282</td>
</tr>
<tr>
<td>10</td>
<td>yarnpkg/yarn</td>
<td>21487</td>
<td>21234</td>
</tr>
</tbody>
</table>
I got stars!
What else did they star?
```sql
SELECT repo, COUNT(*) c FROM (
    SELECT actor.login, ARRAY_AGG(DISTINCT repo.name) repos
    FROM `githubarchive.month.2016`
    WHERE type="WatchEvent" GROUP BY 1
    HAVING ARRAY_LENGTH(repos) BETWEEN 3 AND 30000  # if you star
    AND 'tensorflow/tensorflow' IN UNNEST(repos)
    AND 'FreeCodeCamp/FreeCodeCamp' NOT IN UNNEST(repos)
), UNNEST(repos) repo
GROUP BY 1 ORDER BY 2 DESC
LIMIT 30
```
How did they find me?

Hacker News?
The Hacker News frontpage effect on a project's GitHub stars

To produce this dataset I used the GitHub Archive dataset and the Hacker News dataset - both loaded into Google BigQuery. The question is: How many times does a project get starred after being featured on the frontpage of Hacker News? --@felipehoffa

GitHub daily stars for [gloomyson/StarCraft]

- Show HN. HTML5 version of StarCraft
  HN score: 320

GitHub daily stars for [zenorocha/clipboard.js]

- Clipboard.js: Modern Copy to Clipboard
  HN score: 304

- Clipboard.js: Modern Copy to Clipboard
  HN score: 4

GitHub daily stars for [0xAX/linux-insides]

- Linux-insides: System calls in the Linux kernel, Part 3
  HN score: 73

- Linux-insides: How does the Linux kernel run a program?
  HN score: 132

- Introduction to getting started with Linux kernel development
  HN score: 115
GitHub daily stars for [elixir-lang/elixir]

With BigQuery:

```sql
SELECT repo_name, created_at date, COUNT(*) c, GROUP_CONCAT_UNQUOTED(UNIQUE(hn_date+':'+STRING(hn_score))) hn_dates, SUM(UNIQUE(hn_score)) hn_score, SUM(c) OVER(PARTITION BY repo_name) month_stars
FROM {
    SELECT repo_name, actor_login, DATE(MAX(created_at)) created_at, date hn_date, score hn_score
    FROM [githubarchive:month.201509] a
    JOIN {
        SELECT REGEXP_EXTRACT(url, r'github.com/([a-zA-Z0-9-\.]\+[a-zA-Z0-9-\.]*)') mention, DATE(time_ts) date, score
        FROM [fh-bigquery:hackernews.stories]
        WHERE REGEXP_MATCH(url, r'github.com/[a-zA-Z0-9-\.]\+')
        AND score>10
        AND YEAR(time_ts)=2015 AND MONTH(time_ts)=9
        HAVING NOT (mention CONTAINS '.com/search?' OR mention CONTAINS '.com/blog/')
    } b
    ON a.repo_name=b.mention
    WHERE type="WatchEvent"
    GROUP BY 1,2, hn_date, hn_score
}
GROUP BY 1,2
HAVING hn_score>300
ORDER BY 1,2,4
LIMIT 1000
```
Project health

- Projects with most issues
- Projects with most people filing issues
- Projects with most engagement
- Best projects at closing issues
- Best phrasing for issue closing
SELECT repo.name, COUNT(*) c
FROM [githubarchive:month.201606]
WHERE type IN ( 'IssueCommentEvent')
GROUP BY repo.name
ORDER BY c DESC
LIMIT 10

<table>
<thead>
<tr>
<th>Row</th>
<th>repo_name</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>kubernetes/kubernetes</td>
<td>17883</td>
</tr>
<tr>
<td>2</td>
<td>apache/spark</td>
<td>8620</td>
</tr>
<tr>
<td>3</td>
<td>openshift/origin</td>
<td>5753</td>
</tr>
<tr>
<td>4</td>
<td>sauron-demo/sauron-demo</td>
<td>4922</td>
</tr>
<tr>
<td>5</td>
<td>docker/docker</td>
<td>4661</td>
</tr>
<tr>
<td>6</td>
<td>tgstation/tgstation</td>
<td>4443</td>
</tr>
<tr>
<td>7</td>
<td>cms-sw/cmssw</td>
<td>4352</td>
</tr>
<tr>
<td>8</td>
<td>d3athrow/vgstation13</td>
<td>4134</td>
</tr>
<tr>
<td>9</td>
<td>servo/servo</td>
<td>3668</td>
</tr>
<tr>
<td>10</td>
<td>rust-lang/rust</td>
<td>3450</td>
</tr>
</tbody>
</table>

GitHub repos with the most comments on issues June 2016
```sql
SELECT repo.name,
       ROUND(COUNT(*)/EXACT_COUNT_DISTINCT(actor.login),2) AS comments_per_author,
       EXACT_COUNT_DISTINCT(actor.login) AS authors,
       COUNT(*) AS comments
FROM [githubarchive:month.201606]
WHERE type IN ('IssueCommentEvent')
AND actor.login NOT IN ( SELECT actor.login FROM ( SELECT actor.login, COUNT(*) AS c FROM [githubarchive:month.201606] WHERE type IN ('IssueCommentEvent') GROUP BY 1 HAVING c>1000 ORDER BY 2 DESC ))

<table>
<thead>
<tr>
<th>Row</th>
<th>repo_name</th>
<th>comments_per_author</th>
<th>authors</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>kubernetes/kubernetes</td>
<td>18.03</td>
<td>499</td>
<td>8997</td>
</tr>
<tr>
<td>2</td>
<td>tensorflow/tensorflow</td>
<td>6.11</td>
<td>405</td>
<td>2475</td>
</tr>
<tr>
<td>3</td>
<td>docker/docker</td>
<td>5.94</td>
<td>785</td>
<td>4660</td>
</tr>
<tr>
<td>4</td>
<td>Microsoft/vscode</td>
<td>5.71</td>
<td>601</td>
<td>3433</td>
</tr>
<tr>
<td>5</td>
<td>facebook/react-native</td>
<td>5.15</td>
<td>608</td>
<td>3133</td>
</tr>
<tr>
<td>6</td>
<td>angular/angular</td>
<td>4.89</td>
<td>532</td>
<td>2604</td>
</tr>
<tr>
<td>7</td>
<td>magento/magento2</td>
<td>4.17</td>
<td>468</td>
<td>1952</td>
</tr>
<tr>
<td>8</td>
<td>npm/npm</td>
<td>2.24</td>
<td>439</td>
<td>984</td>
</tr>
<tr>
<td>9</td>
<td>FortAwesome/Font-Awesome</td>
<td>1.6</td>
<td>643</td>
<td>1026</td>
</tr>
</tbody>
</table>
```

GitHub projects with more comments per author within top commented projects (removing bots)
Even text analysis?
Most common ways to open an issue on GitHub, and % that get closed

<table>
<thead>
<tr>
<th>Row</th>
<th>start</th>
<th>authors</th>
<th>percentage_closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>It would be nice</td>
<td>1292</td>
<td>56.5</td>
</tr>
<tr>
<td>2</td>
<td>Is it possible to</td>
<td>1122</td>
<td>73.8</td>
</tr>
<tr>
<td>3</td>
<td>I am trying to</td>
<td>1060</td>
<td>79.8</td>
</tr>
<tr>
<td>4</td>
<td>I would like to</td>
<td>823</td>
<td>66.4</td>
</tr>
<tr>
<td>5</td>
<td>Is there a way</td>
<td>749</td>
<td>74.2</td>
</tr>
<tr>
<td>6</td>
<td>It would be great</td>
<td>663</td>
<td>57.6</td>
</tr>
<tr>
<td>7</td>
<td>So that the humans</td>
<td>632</td>
<td>59.6</td>
</tr>
<tr>
<td>8</td>
<td>When I try to</td>
<td>579</td>
<td>77.4</td>
</tr>
<tr>
<td>9</td>
<td>Would it be possible</td>
<td>377</td>
<td>67.5</td>
</tr>
<tr>
<td>10</td>
<td>Is there any way</td>
<td>282</td>
<td>65.3</td>
</tr>
<tr>
<td>11</td>
<td>Not sure if this</td>
<td>282</td>
<td>81.8</td>
</tr>
<tr>
<td>12</td>
<td>I get the following</td>
<td>242</td>
<td>86.6</td>
</tr>
<tr>
<td>13</td>
<td>I was trying to</td>
<td>229</td>
<td>69.5</td>
</tr>
<tr>
<td>14</td>
<td>I want to use</td>
<td>228</td>
<td>64.2</td>
</tr>
<tr>
<td>15</td>
<td>I have the following</td>
<td>225</td>
<td>78.7</td>
</tr>
</tbody>
</table>

SELECT REGEXP_EXTRACT(JSON_EXTRACT(`issues`, '.*\"start":\"(.*\")'), r'\^([A-Za-z]+ [A-Za-z]+)+$') AS start FROM githubarchive:mont WHERE type='IssuesEvent' GROUP BY 1 HAVING start IS NOT null ORDER BY 2 DESC LIMIT 20
So where's the code?
All the open source code in GitHub now shared within BigQuery: Analyze all the code!
Table Details: contents

Description

Unique file contents of text files under 1 MiB on the HEAD branch.

Can be joined to [bigquery-public-data:github_repos.files] table using the id columns to identify the repository and file path.

Table Info

<table>
<thead>
<tr>
<th>Table ID</th>
<th>bigquery-public-data:github_repos.contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Size</td>
<td>1.79 TB</td>
</tr>
<tr>
<td>Number of Rows</td>
<td>213,849,274</td>
</tr>
<tr>
<td>Creation Time</td>
<td>Mar 11, 2016, 8:18:08 PM</td>
</tr>
<tr>
<td>Last Modified</td>
<td>Jan 26, 2017, 1:33:15 PM</td>
</tr>
<tr>
<td>Data Location</td>
<td>US</td>
</tr>
<tr>
<td>Labels</td>
<td>None</td>
</tr>
</tbody>
</table>
## Table Details: contents

<table>
<thead>
<tr>
<th>Schema</th>
<th>Details</th>
<th>Preview</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>STRING</td>
<td>NULLABLE</td>
</tr>
<tr>
<td>size</td>
<td>INTEGER</td>
<td>NULLABLE</td>
</tr>
<tr>
<td>content</td>
<td>STRING</td>
<td>NULLABLE</td>
</tr>
<tr>
<td>binary</td>
<td>BOOLEAN</td>
<td>NULLABLE</td>
</tr>
<tr>
<td>copies</td>
<td>INTEGER</td>
<td>NULLABLE</td>
</tr>
</tbody>
</table>
```sql
SELECT SUM(size*copies) total_bytes
FROM [bigquery-public-data:github_repos.contents]
```
Rules to analyze
[bigquery-public-data:github_repos.contents]

• Text files <1MB
• One copy of each unique file
• JOIN with [github_repos.files] for paths
• Don't JOIN with [github_repos.files] to get contents*path.
• Extract first, analyze later
• [github_repos.sample_contents]
  -> 10% of contents, top projects, 1 sample path.

• Only open source projects - https://developer.github.com/v3/licenses/
• Some projects missing - why?
Top java imports growth 2013-16
<table>
<thead>
<tr>
<th>Row</th>
<th>Line</th>
<th>imports_2013</th>
<th>imports_2016</th>
<th>ratio_2013</th>
<th>ratio_2016</th>
<th>win</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>import javax.inject.Inject;</td>
<td>5784</td>
<td>25060</td>
<td>0.17</td>
<td>0.42</td>
<td>41.54</td>
</tr>
<tr>
<td>2</td>
<td>import com.google.common.collect.ImmutableList;</td>
<td>4619</td>
<td>19245</td>
<td>0.14</td>
<td>0.32</td>
<td>39.91</td>
</tr>
<tr>
<td>3</td>
<td>import android.os.Build;</td>
<td>3625</td>
<td>14133</td>
<td>0.11</td>
<td>0.23</td>
<td>37.08</td>
</tr>
<tr>
<td>4</td>
<td>import android.text.TextUtils;</td>
<td>3646</td>
<td>12899</td>
<td>0.11</td>
<td>0.21</td>
<td>32.82</td>
</tr>
<tr>
<td>5</td>
<td>import javax.annotation.Nullable;</td>
<td>5083</td>
<td>16487</td>
<td>0.15</td>
<td>0.27</td>
<td>28.89</td>
</tr>
<tr>
<td>6</td>
<td>import java.util.UUID;</td>
<td>7110</td>
<td>22440</td>
<td>0.21</td>
<td>0.37</td>
<td>27.63</td>
</tr>
<tr>
<td>7</td>
<td>import android.view.LayoutInflater;</td>
<td>9090</td>
<td>28453</td>
<td>0.27</td>
<td>0.47</td>
<td>27.25</td>
</tr>
<tr>
<td>8</td>
<td>import org.springframework.beans.factory.annotation.Autowired;</td>
<td>12280</td>
<td>38003</td>
<td>0.36</td>
<td>0.63</td>
<td>26.72</td>
</tr>
<tr>
<td>9</td>
<td>import android.view.MenuItem;</td>
<td>4929</td>
<td>15187</td>
<td>0.15</td>
<td>0.25</td>
<td>26.52</td>
</tr>
<tr>
<td>10</td>
<td>import android.view.ViewGroup;</td>
<td>10722</td>
<td>32863</td>
<td>0.32</td>
<td>0.55</td>
<td>26.28</td>
</tr>
<tr>
<td>11</td>
<td>import static org.mockito.Mockito.verify;</td>
<td>3243</td>
<td>9910</td>
<td>0.1</td>
<td>0.16</td>
<td>26.14</td>
</tr>
<tr>
<td>12</td>
<td>import static org.mockito.Mockito.mock;</td>
<td>3736</td>
<td>11195</td>
<td>0.11</td>
<td>0.19</td>
<td>25.22</td>
</tr>
<tr>
<td>13</td>
<td>import static org.mockito.Mockito.when;</td>
<td>3960</td>
<td>11736</td>
<td>0.12</td>
<td>0.19</td>
<td>24.7</td>
</tr>
<tr>
<td>14</td>
<td>import android.widget.ImageView;</td>
<td>5625</td>
<td>16630</td>
<td>0.17</td>
<td>0.28</td>
<td>24.59</td>
</tr>
</tbody>
</table>

Biggest raise in imports from 2013 to 2016
Requesting a feature for Go
Proposal: Add `time.Until()` to the time package

SamWhited opened this issue on Mar 1, 2016 · 13 comments

SamWhited commented on Mar 1, 2016

I'd like to propose that a `time.Until(t time.Time) time.Duration` function be added to the time package to compliment the existing `Since()` shortcut. This would make writing expressions with an expiration time a bit more readable:

```go
<-After(time.Until(expirationTime))
```

vs.

```go
<-After(expirationTime.Sub(time.Now()))
```

While it's still fairly obvious what the second one does, it takes a little longer to recognize "sub" as subtraction than just seeing the symbol. Also keeping time expressions more or less readable as english is a nice benefit of having the until shortcut (as you can do with the existing since function).

If this accepted, I've got a CL [here](https://example.com) for review.
There's in total around 2000 repos (counting forks only once) that could benefit from this feature.

```
SELECT REGEXP_EXTRACT(repo_name, r'\.*/(.*)') as project, FLOOR(COUNT(*) / COUNT(DISTINCT repo)) as n FROM (SELECT id, split(content, '\n') as line FROM [campoy-github:go_files.contents] HAVING line CONTAINS '.Sub(time.Now())') as contents JOIN [campoy-github:go_files.files] as files ON contents.id = files.id GROUP BY project ORDER BY n DESC
```

The 10 projects that would benefit the most are

<table>
<thead>
<tr>
<th>Row</th>
<th>project</th>
<th>n</th>
<th>sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>robin</td>
<td>62.0</td>
<td>time.Sleep(expectedExp.Sub(time.Now()) - 500*ti</td>
</tr>
<tr>
<td>3</td>
<td>j2</td>
<td>26.0</td>
<td>time.Sleep(expectedExp.Sub(time.Now()) - 500*ti</td>
</tr>
<tr>
<td>4</td>
<td>gitarchive</td>
<td>20.0</td>
<td>c.traceInfo.firstLine.deadline = deadline.Sub(t</td>
</tr>
<tr>
<td>5</td>
<td>megacfs</td>
<td>20.0</td>
<td>sleep := nextRun.Sub(time.Now())</td>
</tr>
<tr>
<td>6</td>
<td>microcosm</td>
<td>20.0</td>
<td>d := deadline.Sub(time.Now())</td>
</tr>
<tr>
<td>7</td>
<td>concourse-pipeline-resource</td>
<td>20.0</td>
<td>req.TimeoutSeconds = proto.Float64(cn.readDeadl</td>
</tr>
<tr>
<td>8</td>
<td>etcd2-bootstrapper</td>
<td>20.0</td>
<td>Timeout: d.Sub(time.Now())</td>
</tr>
<tr>
<td>9</td>
<td>zypper-docker</td>
<td>18.0</td>
<td>timeout = deadline.Sub(time.Now())</td>
</tr>
<tr>
<td>10</td>
<td>doit</td>
<td>16.0</td>
<td>Timeout: d.Sub(time.Now())</td>
</tr>
</tbody>
</table>

In my opinion there's some projects that could benefit of this, but it's clearly not a high priority addition to the stdlib.
Beyond regex

Static code analysis with UDFs
Static JavaScript code analysis inside a SQL query: JSHint+GitHub+BigQuery

Can we run a static code analysis tool for JavaScript inside BigQuery? Yes we can.
SELECT x.error, COUNT(*) files_affected
FROM js(
  SELECT content, sample_path, sample_repo_name
  FROM [fh-bigquery:github_extracts.contents_js]
  WHERE LENGTH(content) BETWEEN 1000 AND 1800
  AND ABS(HASH(id))%1000=0  # sampling
),
content, sample_path, sample_repo_name,
"[
  {name: 'x', type:'string'},
  {name: 'sample_path', type:'string'},
  {name: 'sample_repo_name', type:'string'},
  {name: 'content', type:'string'}]",
"function(r, emit) {
  JSHINT(r.content, {'maxdepth':2});
  // data = JSHINT.data();
  errors = JSHINT.errors;
  set_errors=new Set(errors.map(
    function(x) {
      if(x && 'raw' in x) {return x.raw}}));
  set_errors.forEach(function(x) {
    if(!x) {return;}
    emit({
      x: x,
      sample_repo_name: r.sample_repo_name,
      sample_path: r.sample_path,
    });
  });
}"
GROUP BY 1
ORDER BY 2 DESC
LIMIT 100

7.4s elapsed, 103 GB processed
SELECT error, COUNT(*) files_affected
FROM js
(
    SELECT content, sample_path, sample_repo_name
    FROM [fh-biquerry:github_extracts.content.js]
    WHERE LENGTH(content) BETWEEN 1000 AND 10000
    AND ASSIGNMENT(id)=1800
    # sampling
)
content, sample_path, sample_repo_name
{
    name: 'a', type:'str'
    name: 'sample_path', t
    name: 'sample_repo', name
    'content', type:
    "function(r, e) {
        JSHint(r.content, 'e' // data = JSHint.data: errors = JSHint.errors
        set_errors=new Set(error
        function(x) {
            if (x && 'r' in set_errors, for each (x)
            emit({
                x: x,
                sample_repo_name: sample_path.r.ss
            });
        });
    }"
GROUP BY 1
ORDER BY 2 DESC
LIMIT 100

7.4s elapsed, 183 GB proc

<table>
<thead>
<tr>
<th>Row</th>
<th>error</th>
<th>files_affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Missing semicolon.</td>
<td>377</td>
</tr>
<tr>
<td>2</td>
<td>Use the function form of &quot;use strict&quot;.</td>
<td>199</td>
</tr>
<tr>
<td>3</td>
<td>'{a}' is not defined.</td>
<td>192</td>
</tr>
<tr>
<td>4</td>
<td>'{a}' is only available in ES6 (use esnext option).</td>
<td>120</td>
</tr>
<tr>
<td>5</td>
<td>Expected an assignment or function call and instead saw an expression.</td>
<td>118</td>
</tr>
<tr>
<td>6</td>
<td>'{a}' is available in ES6 (use esnext option) or Mozilla JS extensions (use moz).</td>
<td>117</td>
</tr>
<tr>
<td>7</td>
<td>Use '{a}' to compare with '{b}'.</td>
<td>65</td>
</tr>
<tr>
<td>8</td>
<td>Expected '{a}' and instead saw '{b}'.</td>
<td>53</td>
</tr>
<tr>
<td>9</td>
<td>Blocks are nested too deeply. ({a})</td>
<td>42</td>
</tr>
<tr>
<td>10</td>
<td>Expected an identifier and instead saw '{a}'.</td>
<td>38</td>
</tr>
<tr>
<td>11</td>
<td>Bad line breaking before '{a}'.</td>
<td>33</td>
</tr>
<tr>
<td>12</td>
<td>'{a}' is already defined.</td>
<td>27</td>
</tr>
<tr>
<td>13</td>
<td>Unnecessary semicolon.</td>
<td>26</td>
</tr>
<tr>
<td>14</td>
<td>This character may get silently deleted by one or more browsers.</td>
<td>26</td>
</tr>
</tbody>
</table>
400,000 GitHub repositories, 1 billion files, 14 terabytes of code: Spaces or Tabs?

Tabs or spaces. We are going to parse a billion files among 14 programming languages to decide which one is on top.
Spaces vs Tabs -
GitHub on BigQuery edition

The rules:

• **Data source**: GitHub files stored in BigQuery.
• **Stars matter**: We’ll only consider the top 400,000 repositories – by number of stars they got on GitHub during the period Jan-May 2016.
• **No small files**: Files need to have at least 10 lines that start with a space or a tab.
• **No duplicates**: Duplicate files only have one vote, regardless of how many repos they live in.
• **One vote per file**: Some files use a mix of spaces or tabs. We’ll count on which side depending on which method they use more.
• **Top languages**: We’ll look into files with the extensions (.java, .h, .js, .c, .php, .html, .cs, .json, .py, .cpp, .xml, .rb, .cc, .go).
SELECT a.id id, size, content, binary, copies, sample_repo_name, sample_path
FROM (
    SELECT id, FIRST(path) sample_path, FIRST(repo_name) sample_repo_name
    FROM [bigquery-public-data:github_repos.sample_files]
    WHERE REGEXP_EXTRACT(path, r'\.(^[^/].*)$') IN ('java','h','js','c','php','html','cs','json','py','cpp','xml','rb','cc','go')
    GROUP BY id
) a
JOIN [bigquery-public-data:github_repos.contents] b
ON a.id=b.id

864.6s elapsed, 1.60 TB processed
Spaces vs Tabs - Apply the rules

```sql
SELECT ext, tabs, spaces, countext, LOG((spaces+1)/(tabs+1)) lratio
FROM (  
    SELECT REGEXP_EXTRACT(sample_path, r'\.(\^[\.]\*)$') ext,
            SUM(best='tab') tabs, SUM(best='space') spaces,
            COUNT(*) countext
    FROM (  
        SELECT sample_path, sample_repo_name, IF(SUM(line=''), SUM(line='\t'), 'space', 'tab') best,  
               COUNT(line) bestc  
        FROM (  
            SELECT LEFT(SPLIT(content, '\n'), 1) line, sample_path,  
                   sample_repo_name  
            FROM [fh-
                   bigquery:github_extracts.contents_top_repos_top_langs]  
            HAVING REGEXP_MATCH(line, r'\[ \t\]')  
        )  
        HAVING bestc > 10 # at least 10 lines that start with space or tab  
    )
    GROUP BY ext
)  
ORDER BY countext DESC
LIMIT 100

16.0s elapsed, 133 GB processed
```
Spaces vs Tabs - Results

Tabs vs Spaces (top 400,000 GitHub repos)

Java: tabs - 600,000, spaces - 1,800,000
C: tabs - 600,000, spaces - 1,200,000
JavaScript: tabs - 600,000, spaces - 700,000
PHP: tabs - 600,000, spaces - 800,000
HTML: tabs - 600,000, spaces - 1,000,000
CSS: tabs - 600,000, spaces - 1,200,000
JSON: tabs - 600,000, spaces - 1,800,000
C++: tabs - 600,000, spaces - 1,200,000
Python: tabs - 600,000, spaces - 1,000,000
XML: tabs - 600,000, spaces - 1,200,000
Ruby: tabs - 600,000, spaces - 800,000
CoffeeScript: tabs - 600,000, spaces - 600,000
Go: tabs - 600,000, spaces - 800,000

@felipehoffa
bit.ly/spacetabs
Who wants to analyze GitHub?

Project maintainers
Project users
Project choosers
Data lovers
YOU!
Exploring GitHub with BigQuery at GitHub
Way more:

- 1 hour after the dataset announcement @thomasdarimont was able to find all the java projects that declare certain dependency.
- Lakshmanan V “Popular Java projects on GitHub that could use some help” (analyzed using BigQuery and Dataflow).
- Guillaume Laforge “What can we learn from million lines of Groovy code on GitHub?”.
- Filippo Valsorda “Analyzing Go Vendoring with BigQuery”.
- Go project uses BigQuery stats to guide design decisions, more than once.
- David Gageot analyzes 281,212 Docker projects.
- Kan Nishida uses R to cluster R packages.
- Aja Hammerly compares most popular gems according to Rubygems.org download data vs GitHub gem calls.
- Sergey Abakumoff looks at the most popular npm packages and trending keywords. Justin Beckwith performs a similar analysis. Sergey follows up with a deeper assessment on why almost empty packages duplicate all over GitHub. Sergey Abakumoff also analyzes Angular vs React messages.
- Brent Shaffer analyzes PHP code and libraries—also test coverage for different languages.
- A full run down by Egor Zhuk, “Yet another analysis of Github data with Google BigQuery”.
- John-David Dalton informs the travis-ci team on the counts for Node versions tested.
- Alex Zhimtisky reviews 779,236 Java Logging Statements, 1,313 GitHub Repositories to determine “ERROR, WARN or FATAL”?
- Florin Badita “Naming conventions in Python import statements”. Then “Naming conventions in Python def function()”.

- @anvaka “analyzed ~2TB of code to build an index of the most common words in programming languages”. Cool visualizations, full code on GitHub, and a lot of comments on reddit.
- Sergey Abakumoff comes back, linking code to StackOverflow.
- Gareth Rushgrove finds all kind of metrics for Puppet.

A series of posts by Robert Kozikowski:

- Advanced GitHub search with BigQuery.
- Top emacs packages used in GitHub repos.
- Visualizing relationships between python packages.
Thanks to @Google #BigQuery, you can now enjoy 1 TB of lightening-fast SQL queries to @TravisTorrent for free: travisTorrent.testroots.org/page_access/

TravisTorrent now in BigQuery: 60 facts for each of 3.7 million Travis CI jobs (2.87GB) dlvr.it/ND1zJk

Become a @travisci log miner (with TravisTorrent) and participate in the MSR Mining Challenge 2017! blog.travis-ci.com/2017-01-16-tra...
Top countries by number of unique ids / population

@felipehofffa
Imports in Java from 2013 to 2016: Winners and losers

With all of GitHub open source contents inside BigQuery, we can now go into the actual contents on each file. For example, let’s find out the most popular Java imports in 2016:

```
SELECT line, COUNT(*) c
FROM (  
    SELECT SPLIT(content, '\n') line  
    FROM [fh-bigquery:github_extracts.contents_java_2016]  
    HAVING REGEXP_MATCH(line, '^import')  
)
GROUP BY 1
ORDER BY 2 DESC
LIMIT 300
```
2016 top imports vs 2010 top imports

<table>
<thead>
<tr>
<th>Row</th>
<th>line</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>import java.util.List;</td>
<td>428404</td>
</tr>
<tr>
<td>2</td>
<td>import java.util.ArrayList;</td>
<td>249648</td>
</tr>
<tr>
<td>3</td>
<td>import java.util.Map;</td>
<td>221261</td>
</tr>
<tr>
<td>4</td>
<td>import java.io.IOException;</td>
<td>218662</td>
</tr>
<tr>
<td>5</td>
<td>import org.junit.Test;</td>
<td>157233</td>
</tr>
<tr>
<td>6</td>
<td>import java.util.HashMap;</td>
<td>128605</td>
</tr>
<tr>
<td>7</td>
<td>import java.util.Set;</td>
<td>101870</td>
</tr>
<tr>
<td>8</td>
<td>import java.io.File;</td>
<td>100194</td>
</tr>
<tr>
<td>9</td>
<td>import org.slf4j.Logger;</td>
<td>86523</td>
</tr>
<tr>
<td>10</td>
<td>import org.slf4j.LoggerFactory;</td>
<td>84086</td>
</tr>
<tr>
<td>11</td>
<td>import java.util.Arrays;</td>
<td>82459</td>
</tr>
<tr>
<td>12</td>
<td>import android.content.Context;</td>
<td>80615</td>
</tr>
<tr>
<td>13</td>
<td>import java.util.Collections;</td>
<td>78121</td>
</tr>
<tr>
<td>14</td>
<td>import java.util.Collection;</td>
<td>73953</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Row</th>
<th>line</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>import java.io.IOException;</td>
<td>36503</td>
</tr>
<tr>
<td>2</td>
<td>import java.util.List;</td>
<td>35465</td>
</tr>
<tr>
<td>3</td>
<td>import java.util.ArrayList;</td>
<td>24212</td>
</tr>
<tr>
<td>4</td>
<td>import java.util.Map;</td>
<td>20491</td>
</tr>
<tr>
<td>5</td>
<td>import java.io.File;</td>
<td>14455</td>
</tr>
<tr>
<td>6</td>
<td>import java.util.HashMap;</td>
<td>13444</td>
</tr>
<tr>
<td>7</td>
<td>import java.util.Set;</td>
<td>11242</td>
</tr>
<tr>
<td>8</td>
<td>import java.util.Iterator;</td>
<td>10944</td>
</tr>
<tr>
<td>9</td>
<td>import org.junit.Test;</td>
<td>10797</td>
</tr>
<tr>
<td>10</td>
<td>import java.util.*;</td>
<td>10612</td>
</tr>
<tr>
<td>11</td>
<td>import java.util.Collection;</td>
<td>8705</td>
</tr>
<tr>
<td>12</td>
<td>import java.io.InputStream;</td>
<td>8650</td>
</tr>
<tr>
<td>13</td>
<td>import junit.framework.TestCase;</td>
<td>7721</td>
</tr>
<tr>
<td>14</td>
<td>import java.util.Arrays;</td>
<td>7697</td>
</tr>
</tbody>
</table>
GHTorrent on the Google cloud

GHTorrent can be accessed over Google Cloud services. To access the data requires you to have a Google Cloud account. Reasonable use is free of charge and, in the case of BigQuery, it should no longer require a credit card. (Pub/Sub still requires a credit card). You can check what Google considers reasonable at any given moment here.

- **Google BigQuery** contains an up to date import of the latest GHTorrent MySQL dump.
- **Google Pub/Sub** exposes real-time streams of GitHub activity.

Both services can be accessed through the Web, the command line (after installing the Google Cloud command line utils) or through various programming languages.

**BigQuery**

With BigQuery, you can query GHTorrent’s MySQL dataset using an SQL-like language (lately, BigQuery also supports vanilla SQL); more importantly, you can join the dataset with other open datasets (e.g. GitHub’s own project data, Reddit, TravisTorrent etc) hosted on BigQuery.

To get the most popular programming languages by number of bytes written, run the following:

```sql
select pl3.lang, sum(pl3.size) as total_bytes
from ( 
    select pl2.bytes as size, pl2.language as lang
    from ( 
    ```
Pub/Sub

Pub/Sub allows subscribers to get events of what is happening on GitHub (or at least GHTorrent's interpretation of what is happening on GitHub) in almost real time. To do so, one needs to subscribe to one of the available topics with a client in order to start receiving events.

The service is complimentary, even though less fine-grained, to GHTorrent's own streaming interface. As is also the case with GHTorrent streaming, the contents of the streams are generated by following the live MongoDB server replication stream. See the code here.

To subscribe to a topic, e.g. commits, run the following:

```
gcloud beta pubsub subscriptions create my_commits_subscription --topic projects/ghtorrent-bq/topics/commits
```

To start receiving events, you can try the command line

```
gcloud beta pubsub subscriptions pull --auto-ack --max-messages 5 -- my_commits_subscription
```

The available topics are the following:

```
projects/ghtorrent-bq/topics/commits
projects/ghtorrent-bq/topics/events
projects/ghtorrent-bq/topics/followers
projects/ghtorrent-bq/topics/forks
projects/ghtorrent-bq/topics/issue_comments
projects/ghtorrent-bq/topics/issue_events
projects/ghtorrent-bq/topics/issues
projects/ghtorrent-bq/topics/org_members
projects/ghtorrent-bq/topics/pull_request_comments
projects/ghtorrent-bq/topics/pull_requests
projects/ghtorrent-bq/topics/repo_collaborators
projects/ghtorrent-bq/topics/repo_labels
projects/ghtorrent-bq/topics/repos
projects/ghtorrent-bq/topics/users
projects/ghtorrent-bq/topics/watchers
```