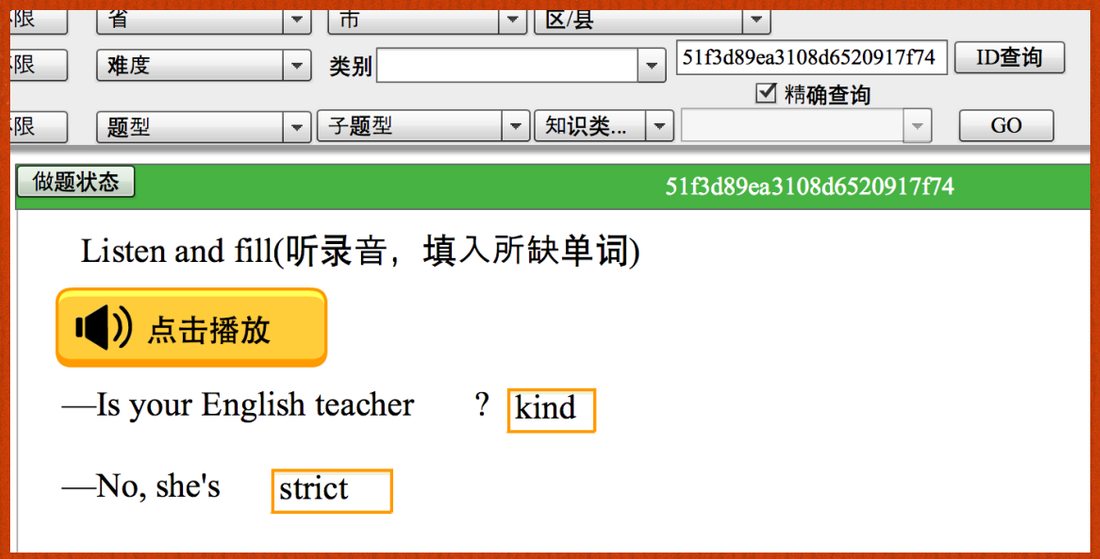
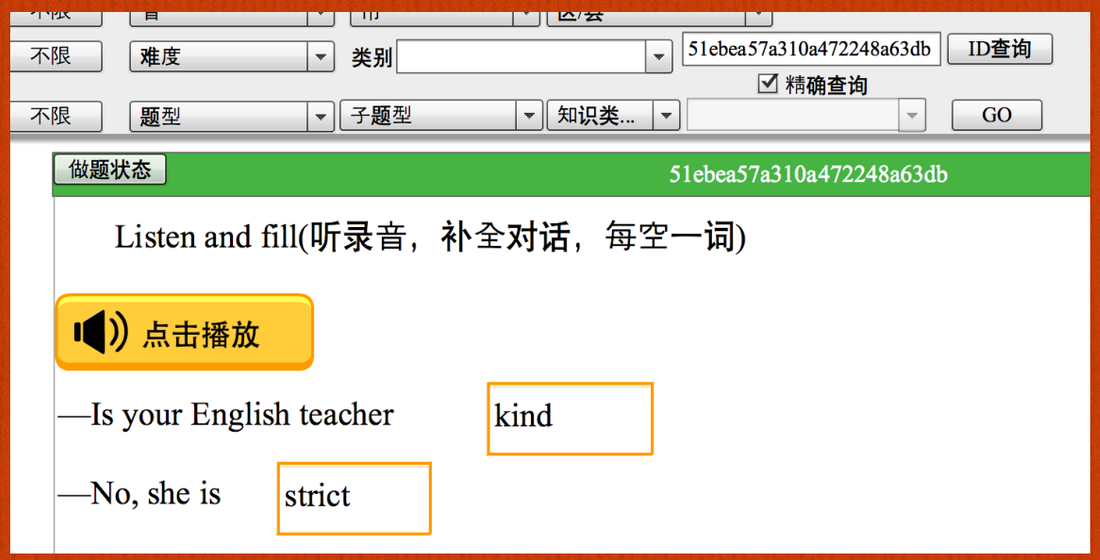
DetDup排重引擎

陈大伟 @ 17zuoye

http://hg.17zuoye.net/detdup

2014.08.24





Detect duplicated items

Agenda

1 重复内容的定义

2 两两比较复杂度

3 相似性算法挑选

4 软件工程架构和优化

Definition

长度	基本相似或相等, 两者长度的平方根相差不超过1。
重复	在任意位置, 多个逗号, 空格, s字符等。
同义	全角半角编码。分隔符号不同。am, 'm。
顺序	内部句子位子换了，比如从连线题里抽取的数据。

原始字符 VS 分词: 文本越小，分词效果的差异越大。

相似性算法挑选

文本	Dice	重复度
AGoodnightGoodmorning勾选	分词(费时)	10/12 # => 83.33%
AGoodnightGoodmorning圈选	unicode	44/46 # => 95.65%

"两两比较"时间复杂度

一个朴素的问题

O(1)	O(log n)	O(n)	n log(n)	O(n2)	O(n!)
			x	√	

```
$ irb
2.1.1 :001 > cal = lambda {|n| n * (n - 1) / 2 }
 => #<Proc:0x0000010300b758@(irb):1 (lambda)>
2.1.1 :002 > cal[1000*1000]
 => 499999500000
2.1.1 :003 > cal[500*1000]
 => 124999750000
2.1.1 :004 > cal[100*1000]
 => 4999950000
2.1.1 :005 > cal[10*1000]
 => 49995000
2.1.1 :006 >
```

更精确的复杂度是: n(n-1) / 2

软件架构

API

Task

Core

ModelCache

1	2	3	4	5	...

Features-Trees

tree | tree | tree | tree

配置特征

通用	uniq_chars__len	sqrt_chars__len	sorted_freq_chars	
业务	options_uniq_chars__len	options_sorted_freq_chars	options__len	…

```
6
7  from detdup.features.default import DefaultFeatures
8  class PLFeature(DefaultFeatures):
9      """ programming language """
10     def post_init(self):
11         # 在特征数据库级别划分
12         self.typename      = 'pl'
13
14         self.custom_features = {
15             'desc' : str,
16           }
17
```

数据准备

操作	extract	build features-trees and model-cache
存储	cPickle	sqlite and ModelCache

Task

extract

预先排重

```sql

SELECT
  t1."sorted_freq_chars",
  group_concat(t1."item_id") AS item_ids
FROM
  "DefaultFeaturesTree" AS t1
GROUP BY
  t1."sorted_freq_chars";

```

1. 选出需要排重的item-ids

```sql

SELECT
  t1."id",
  t1."uniq_chars__len",
  t1."sqrt_chars__len",
  t1."sorted_freq_chars",
  t1."item_id",
  t1."desc"
FROM
  "Desc" AS t1
WHERE
  (((((t1."uniq_chars__len" >= 3)) AND
  (t1."uniq_chars__len" <= 9)) AND
  (t1."sorted_freq_chars" = 'hn')) AND
  (t1."desc" = 'programming language'));

```

2. 给每一个item划分排重域

Task

train

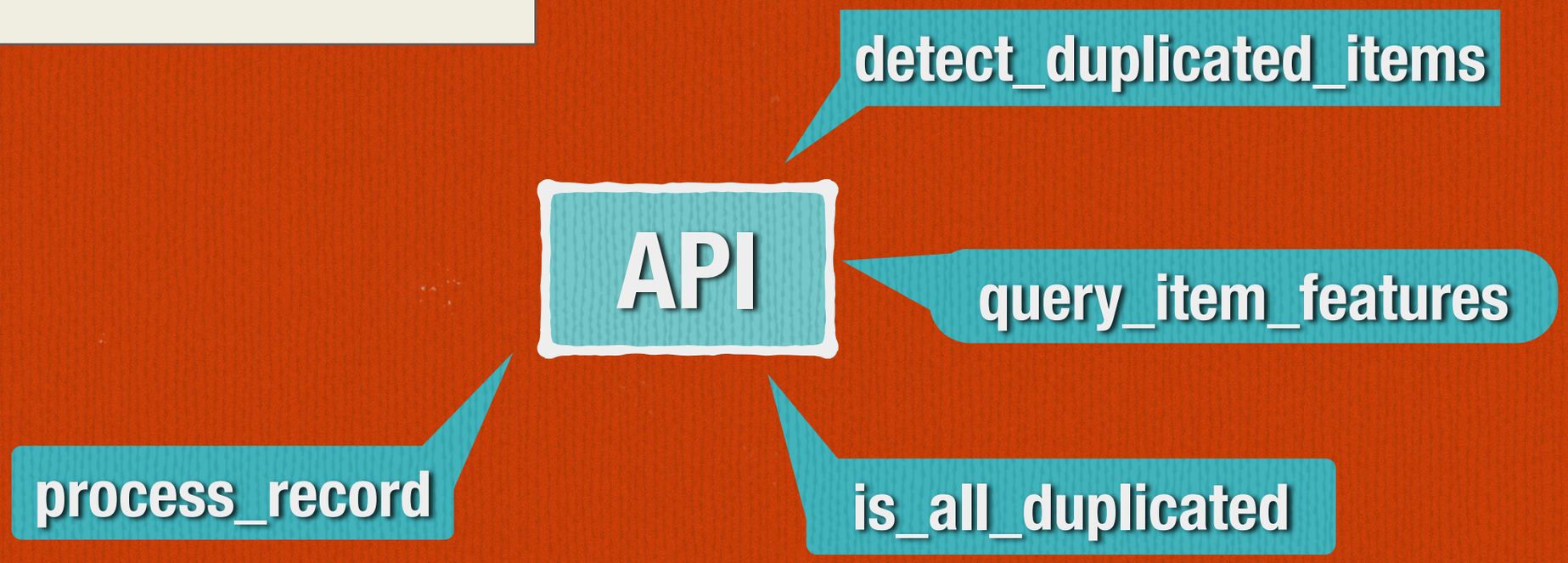
3. 排重缓存。

item1 => [item1, item2, item3]

item2 => 缓存命中(ItemsGroupAndIndexes)

实时排重

放入排重特征库中比对	
1	临时(FakeItemIds)
2	永久



软件工程优化

1	多进程数据清洗
2	**sqlitebck** 内存磁盘相互拷贝
3	动态定义特征数据库表
4	...总是还可以更好

性能数据

文本相似度	排重效果	重复元素	重复组
95%	几乎全部正确	3199个	1463组
90%	一点点错误	3297个	1507组

相当于重复元素多了98个, 重复组多了44个, 重复[组]90-95之间多了 44 / 1463.0 = 3.0%, 重复元素90-100%元素约为 7.4%。
在文本相似度为90%时，误判率大概在 重复元素 19 / 3297.0 = 0.57%, 重复组在 9 / 1507.0 = 0.59%;

性能和总数以及重复元素总量成线性增长关系。

特征库查找速度

Sqlite多维索引查找速度(IO, 查找树算法等优化方向)

其他开源项目

article_segment

fill_broken_words

etl_utils

model_cache

region_unit_recognizer

compare_word

tfidf

phrase_recognizer

split_block

https://github.com/17zuoye

pip install etl_utils

谢谢！

好身体才有好代码！

勤思考，挺直背，多喝水。

```
$ ruby -e 'loop { sleep 600; `open http://have-a-break` }'
```

内容如有错误，请指正！