Textwrangling and Wordsmithing

Derick Rethans

derick@mongodb.com – @derickr

http://joind.in/14372

Examples

- Find all the blog postings that contain a comment mentioning “OpenStreetMap”
- Find all the profiles where the description is relevant to “databases”, return the profile date
- Find all the product metadata with description relevant to “Nikon S9500”

Text is difficult

```
<?php
echo substr("Åmsterdam PHP", 0, 1), "\n";
?>
```

Result:

```
?
```

```
<?php
var_dump("Å" === "Å");
?>
```

Result:

```
boolean false
```

Agenda

- First Steps
- Theory: tokenizing and stemming
- MongoDB text search
- Sorting
- Using ElasticSearch

Text indexing, the easy way (1)

```
Xdebug 2.1.0beta3 released
==========================

.. articleMetaData::
   :Where: London, UK
   :Date: 2010-02-27 23:57 Europe/London
   :Tags: blog, php, xdebug, extensions

I've just released Xdebug 2.1.0beta3 which includes a few crash bugs as well as
the issue that headers sent from PHP scripts are not actually set.

You can find the full changelog here_ and get the latest version from the
`download page`_.

.. _here: http://xdebug.org/updates.php#x_2_1_0beta3
.. _`download page`: http://xdebug.org/download.php
```

Text indexing, the easy way (2)

```
<?php
$m = new MongoClient;
$m->demo->articles->drop();

foreach ( glob( '201*rst' ) as $file )
{
        $c = file_get_contents( $file );
        $c = str_replace('\\n', "\n", $c);
        preg_match( '/^.*/', $c, $match );

        $m->demo->articles->insert( [
                '_id' => $file,
                'subject' => $match[ 0 ],
                'text' => $c,
                'simple_index' => preg_split( '/[ \n]/', $c ),
        ] );
}
```

Text indexing, the easy way (3)

```
> db.articles.find({_id: '201002272352-xdebug-210beta3.rst'}).pretty();
{
  "_id" : "201002272352-xdebug-210beta3.rst",
  "subject" : "Xdebug 2.1.0beta3 released",
  "text" : "Xdebug 2.1.0beta3 released
==========================

.. articleMetaData::
   :Where: London, UK
   :Date: 2010-02-27 23:57 Europe/London
   :Tags: blog, php, xdebug, extensions

I've just released Xdebug 2.1.0beta3 which includes a few crash bugs as well as
the issue that headers sent from PHP scripts are not actually set.

You can find the full changelog here_ and get the latest version from the
`download page`_.

.. _here: http://xdebug.org/updates.php#x_2_1_0beta3
.. _`download page`: http://xdebug.org/download.php
",
  "simple_index" : [
    "Xdebug", "2.1.0beta3", "released", "==========================", "",
    "..", "articleMetaData::", "", "", "", ":Where:", "London,", "UK", "",
    "", "", ":Date:", "2010-02-27", "23:57", "Europe/London", "", "", "",
    ":Tags:", "blog,", "php,", "xdebug,", "extensions", "", "I've", "just",
    "released", "Xdebug", "2.1.0beta3", "which", "includes", "a", "few",
    "crash", "bugs", "as", "well", "as", "the", "issue", "that", "headers",
```

Text indexing, the easy way (4)

```
es:PRIMARY> db.articles.find(
        { simple_index: { $all: [ 'Advent', 'Xdebug'] } },
        { subject: 1, _id: 0 }
).pretty();

{ "subject" : "Contributing Advent 1: Xdebug and hidden properties" }
{ "subject" : "Contributing Advent 8: The magic __FILE__ constant" }
{ "subject" : "Contributing Advent 15: Xdebug connection timeout" }
{ "subject" : "Contributing Advent 17: Printing stacks" }
{ "subject" : "Contributing Advent 20: Xdebug halting on error" }
{ "subject" : "Contributing Advent 23: Reproducing issues" }
{ "subject" : "Contributing Advent 24: Wrapping up!" }

es:PRIMARY> es:PRIMARY> db.articles.find(
        { simple_index: { $all: [ 'advent', 'Xdebug'] } },
        { subject: 1, _id: 0 }
).pretty();

es:PRIMARY>
```

Tokenizing

Making indexing parts out of text.

Text

```
"This standard was developed from ISO/IEC 9075:1989"

Whitespace:
"This" "standard" "was" "developed" "from" "ISO/IEC" "9075:1989"

Continuous letters:
"This" "standard" "was" "developed" "from" "ISO" "IEC"
```

HTML

```
"<li><em>If it exists</em>, the STATUS of the W3C document.</li>"

"If" "it" "exists" "the" "status" "of" "the" "w3c" "document"
```

Tokenizing

Japanese

There is little interpunction:

辞書,コーパスに依存しない汎用的な設計

You need special techniques to split it up into bits.
Tools like Kakasi and Mecab.

Output from mecab:

```
辞書，コーパスに依存しない汎用的な設計

辞書        名詞,普通名詞,*,*,辞書,じしょ,代表表記:辞書
，          特殊,記号,*,*,*,*,*
コーパス    名詞,普通名詞,*,*,*,*,*
に          助詞,格助詞,*,*,に,に,*
依存        名詞,サ変名詞,*,*,依存,いぞん,代表表記:依存
し          動詞,*,サ変動詞,基本連用形,する,し,付属動詞候補（基本）　代表表記:する
ない        接尾辞,形容詞性述語接尾辞,イ形容詞アウオ段,基本形,ない,ない,*
```

Tokenizing

Domain specific

Tokenization is domain specific

- You don't always want to split up letters from numbers - f.e. in product numbers.
- You might want to exclude words (stop words)
- You might want to filter out words that are short, or just long
- You might want to define synonyms
- You might want to normalize text (remove accents, Unicode forms)

Stemming Examples

- { walk, walked, walking, walks } ⇒ walk
- { magazine, magazines, magazine's } ⇒ magazine
- { runs, running, run, ran } ⇒ { run, ran }

Stop words

words that are too common to contribute to the document relevance

Examples of English Stop Words

{ am, themselves, of, before, here, while, what's, myself, ought, me, the, into, about, this, do, can't, a, ... }

When to Use MongoDB Text Search

- Adding text search to an existing db application
- Simplify application search-db architecture
- Not a substitute for a dedicated search application (ElasticSearch)

Create a Text Search Index

```
<?php
$m = new MongoClient;

$m->demo->articles->ensureIndex(
        [ 'subject' => 'text', 'post' => 'text' ],
        [ 'weights' => [ 'subject' => 10, 'post' => 5 ] ]
);
?>
```

Do a Text Search Query

```
<?php
$m = new MongoClient;
$cursor = $m->demo->articles->find(
        [ '$text' => [ '$search' => '"advent" "xdebug"' ] ],
        [ 'subject' => 1, '_id' => 0 ]
);
foreach ( $cursor as $result ) {
        echo $result['subject'], "\n";
}
?>
```

Result:

```
Contributing Advent 15: Xdebug connection timeout
Contributing Advent 8: The magic __FILE__ constant
Contributing Advent
Contributing Advent 24: Wrapping up!
Whisky Advent: part 4
Contributing Advent 23: Reproducing issues
Contributing Advent 22: Documenting changes
Contributing Advent 1: Xdebug and hidden properties
Whisky Advent: part 3
Contributing Advent 20: Xdebug halting on error
Contributing Advent 17: Printing stacks
```

Indexing Options

Basic, default weights:

```
db.articles.ensureIndex( { subject: "text", post: "text" } );
```

Explicit weights:

```
db.articles.ensureIndex( { subject: "text", post: "text" },
                         { weights: { "subject": 10 } } );
```

Wildcard field: text at any depth, default weights:

```
db.articles.ensureIndex( { "$**": "text" } );
```

Wildcard field: override default weights and explicit weights:

```
db.articles.ensureIndex( { "$**": "text" },
                         { weights: {"$**": 10, post: 5 } } );
```

Language-Specific Stemming

```
t = db.search; t.drop();
t.save( { _id: 1, title: "mi blog", post: "Este es un blog de prueba" } );
t.save( { _id: 2, title: "cuchillos son divertido", post: "Es mi tercer blog stemmed" } );
t.save( { _id: 3, title: "My fourth blog", post: "This stemmed blog is in english" } );
t.ensureIndex( { title: "text", post: "text" }, { default_language: "spanish" } );
```

Text Search Query Syntax

summer **OR** olympics (or strongly preferred: both)

```
db.articles.find( { '$text' : { '$search' : "Summer Olympics" } } );
```

phrase "Summer Olympics"

```
db.articles.find( { '$text' : { '$search' : "\"Summer Olympics\"" } } );
```

phrase "wild flowers" **OR** Sydney

```
db.articles.find( { '$text' : { '$search' : "\"wild flowers\" Sydney" } } );
```

wild **AND** flowers

```
db.articles.find( { '$text' : { '$search' : "\"wild\" \"flowers\"" } } );
```

industry **ANDNOT** Melbourne **ANDNOT** Physics

```
db.articles.find( { '$text' : { '$search' : "industry -Melbourne -Physics" } } );
```

Query Options and Sorting

```
<?php
$m = new MongoClient;

$cursor = $m->demo->articles->find(
        [
                '$text' => [ '$search' => '"advent" "xdebug"' ],
                'subject' => new MongoRegex( '/an/' )
        ],
        [
                '_id' => 0, 'subject' => 1,
                'score' => [ '$meta' => 'textScore' ]
        ]
)->limit( 4 )->sort( [ 'score' => [ '$meta' => 'textScore' ] ] );

foreach ( $cursor as $record )
{
        printf( "Score: %4.2f, Title: %s\n", $record['score'], $record['subject'] );
}
?>
```

Result:

```
Score: 20.62, Title: Contributing Advent 1: Xdebug and hidden properties
Score: 13.90, Title: Contributing Advent 22: Documenting changes
Score: 13.40, Title: Contributing Advent 8: The magic __FILE__ constant
```

Sorting Strings

How would you sort: **côté** (side), **côte** (coast), **cote** (dimension), **coté** (with dimensions)?

The french are not the only ones with "weird" sorting!

- In Lithuanian, **y** is sorted between **i** and **k**.
- In traditional Spanish **ch** is treated as a single letter, and sorted between **c** and **d**.
- In Swedish **v** and **w** are considered variant forms of the same letter.
- In German dictionaries, **öf** would come before **of**. In phone books the situation is the exact opposite.

Locales — example

Comparing strings:

```
<?php
$coll = new Collator("fr_CA");
if ($coll->compare("côte", "coté") < 0) {
        echo "less\n";
} else {
        echo "greater\n";
}
?>
```

Result:

```
less
```

Ignore case and accents:

```
<?php
$coll = new Collator("fr_CA");
$coll->setStrength(Collator::PRIMARY);
if ($coll->compare("côte", "cOTé") == 0) {
        echo "same\n";
} else {
        echo "different\n";
}
```

Array Sorting Example

```
<pre>#  orig  norm  loc   trad
--------------------------
<?php
$d = $c = $b = $a = array('mapa', 'kilo', 'libro', 'llave', 'loca');
sort($b);
$col1 = new Collator('es_VE');
$col1->sort($c);
$col2 = new Collator('es_VE@collation=traditional');
$col2->sort($d);

for ($i = 0; $i < 5; ++$i) {
        echo sprintf('%d. %-5s %-5s %-5s %-5s<br/>',
                $i + 1, $a[$i], $b[$i], $c[$i], $d[$i]);
}
?>
```

Result:

```
#  orig  norm  loc   trad
---------------------------
1. mapa  kilo  kilo  kilo
2. kilo  libro libro libro
3. libro llave llave loca
4. llave loca  loca  llave
5. loca  mapa  mapa  mapa
```

Sorting with MongoDB

We have the following words in a field:

```
{ "_id" : ObjectId("53fc721844670a35498b4567"), "word" : "bailey" }
{ "_id" : ObjectId("53fc721844670a35498b4568"), "word" : "boffey" }
{ "_id" : ObjectId("53fc721844670a35498b4569"), "word" : "böhm" }
{ "_id" : ObjectId("53fc721844670a35498b456a"), "word" : "brown" }
{ "_id" : ObjectId("53fc721844670a35498b456b"), "word" : "сергéй" }
{ "_id" : ObjectId("53fc721844670a35498b456c"), "word" : "сергий" }
{ "_id" : ObjectId("53fc721844670a35498b456d"), "word" : "swag" }
{ "_id" : ObjectId("53fc721844670a35498b456e"), "word" : "svere" }
```

Let's sort them:

```
es:PRIMARY> db.collate.find( {}, { _id: 0, word: 1 }).sort( { word:1 } ).pretty();

{ "word" : "bailey" }
{ "word" : "boffey" }
{ "word" : "brown" }
{ "word" : "böhm" }
{ "word" : "svere" }
{ "word" : "swag" }
{ "word" : "сергéй" }
{ "word" : "сергий" }
```

Success?

Different Cultures → Different Sorting

- **English:**
  bailey boffey brown böhm svere swag cepréй сергий
- **German:**
  bailey böhm boffey brown svere swag cepréй сергий
- **Swedish (1):**
  bailey boffey brown böhm swag svere cepréй сергий
- **Swedish (2):**
  bailey boffey brown böhm svere swag cepréй сергий
- **Russian:**
  cepréй сергий bailey boffey böhm brown svere swag

MongoDB doesn't support this yet ☹

We can make it work!

```php
<?php
$words = [
    'bailey', 'boffey', 'böhm', 'brown', 'сергéй', 'сергий', 'swag', 'svere'
];

$collator = new Collator( 'sv@collation=standard' );
foreach ( $words as $word )
{
    $sortKey = $collator->getSortKey( $word );
    printf("%-26s: %-12s\n", bin2hex( $sortKey ), $word );
}
?>
```

Result:

```
2927373d2f57010a010a      : bailey
294331312f57010a010a      : boffey
295aa106353f01080108      : böhm
2949435141018487050109    : brown
5cba34b41a346601828d05010b: сергéй
5cba34b41a5a66010a010a    : сергий
4b512733018687060108      : swag
4b512f492f01090109        : svere
```

We can make it work!

```php
<?php
$words = [
    'bailey', 'boffey', 'böhm', 'brown', 'сергéй', 'сергий', 'swag', 'svere'
];

$collator = new Collator( 'sv@collation=default' );
foreach ( $words as $word )
{
    $sortKey = $collator->getSortKey( $word );
    printf("%-26s: %-12s\n", bin2hex( $sortKey ), $word );
}
?>
```

Result:

```
2927373d2f57010a010a      : bailey
294331312f57010a010a      : boffey
295aa106353f01080108      : böhm
29494353410109010 9       : brown
5cba34b41a346601828d05010b: сергéй
5cba34b41a5a66010a010a    : сергий
4b532733010801 08         : swag
4b512f492f01090109        : svere
```

Locale Based Sorting (1)

```php
<?php
$words = [ 'bailey', 'boffey', 'böhm', 'brown', 'сергéй', 'сергий', 'swag', 'svere' ];
$collations = [ 'de_DE@collation=phonebook', 'sv@collation=standard', 'ru' ];

$m = new MongoClient;
$c = $m->demo->collate;
$c->drop();

$collators = [];

foreach ( $collations as $collation )
{
    $c->createIndex( [ $collation => 1 ] );
    $collators[$collation] = new Collator( $collation );
}

foreach ( $words as $word )
{
    $doc = [ 'word' => $word ];
    foreach ( $collations as $collation )
    {
        $sortKey = $collators[$collation]->getSortKey( $word );
        $doc[$collation] = bin2hex( $sortKey );
    }
    $c->insert( $doc );
}
?>
```

Locale Based Sorting (2)

```
<?php
$collations = [ 'de_DE@collation=phonebook', 'sv@collation=standard', 'ru' ];

$m = new MongoClient;
$c = $m->demo->collate;

foreach ( $c->find() as $word )
{
        echo $word['word'], "\n";
        foreach ( $collations as $collation )
        {
                printf("  %-30s %s\n", $collation, $word[$collation]);
        }
        echo "\n";
}
?>
```

Result:

```
bailey
  de_DE@collation=phonebook      2927373d2f57010a010a
  sv@collation=standard          2927373d2f57010a010a
  ru                             2b29393f3159010a010a

boffey
  de_DE@collation=phonebook      294331312f57010a010a
  sv@collation=standard          294331312f57010a010a
  ru                             2b4533333159010a010a
```

Locale Based Sorting (3)

```
<?php
$m = new MongoClient;
$c = $m->demo->collate;

foreach ( $c->find()->sort( [ "de_DE@collation=phonebook" => 1 ] ) as $word )
{
        echo $word['word'], ' ';
}
echo "\n";
?>
```

Result:

```
bailey böhm boffey brown svere swag сергéй сергий
```

MongoDB and locale based sorting: conclusion

- Standard sort order is Unicode code point
- Collators from PHP can help
- We need locale support in indexes
- WiredTiger engine supports it
- Vote for:
  `https://jira.mongodb.org/browse/SERVER-1920`

Elasticsearch

- Elasticsearch is a full text search engine
- Support for custom stemming, tokenizing, stop-words, languages, etc.
- It's a separate easy to scale NoSQL solution on its own, focussed on text search

Elasticsearch + MongoDB

- A script feeds data from MongoDB into Elasticsearch
- Works as a replication slave
- Updates and inserts are nearly instantaneous
- Best of both worlds: MongoDB as a modern database, Elasticsearch as a first class search engine

Setting up Elasticsearch

- MongoDB 2.6.6, Elasticsearch 1.4.4
- Install a few plugins:

```
bin/plugin -install elasticsearch/elasticsearch-mapper-attachments/2.4.3
bin/plugin -install mobz/elasticsearch-head
```

- Configure MongoDB as a replica set
- Configure mappings

Configuring up Elasticsearch

Setup mappings:

```
curl -XPUT "localhost:9200/articles/_mapping/articles" -d '
{
  "articles" : {
    "_source" : { "enabled" : true },
    "properties" : {
      "subject" : { "type" : "string", "boost" : 2.0, "analyzer" : "english" },
      "text": { "type" : "string", "analyzer" : "english" }
    }
}'
```

Simple query from PHP

```
<?php
$url = 'http://localhost:9200/articles/articles/_search?q=subject:xdebug&fields=subject';
$a = json_decode( file_get_contents( $url ) );

foreach ( $a->hits->hits as $hit )
{
        printf("%.2f: %s\n", $hit->_score, $hit->fields->subject[0] );
}
?>
```

Result:

(!) Warning: file_get_contents(http://localhost:9200/articles/articles/_search?q=subject:xdebug&fields=subjec

Call Stack

#	Time	Memory	Function
1	0.0020	313960	{main}( )
2	0.0136	685104	Presentation->display($slideNr = '36')
3	0.0169	788912	ezcTemplate->process($location = 'slide.ezt', $config = ???)
4	0.0211	910304	ezcTemplateCompiledCode->execute( )
5	0.0230	958680	include('/tmp/template-cache/compiled_templates/xhtml-updqr0/slide-5eb6f7fc995a21e41e7fdbd1e486972(
6	0.0242	973896	ezcTemplate->process($location = 'slide-page.ezt', $config = class ezcTemplateConfiguration { private $pr(
7	0.0247	978584	ezcTemplateCompiledCode->execute( )
8	0.0254	996168	include('/tmp/template-cache/compiled_templates/xhtml-updqr0/slide-page-8d73e5f10c7184bfde5e4f68dc2
9	0.0259	1003640	ezcTemplate->process($location = 'slide-contents.ezt', $config = class ezcTemplateConfiguration { private
10	0.0264	1008528	ezcTemplateCompiledCode->execute( )
11	0.0270	1021808	include('/tmp/template-cache/compiled_templates/xhtml-updqr0/slide-contents-2669fd655b7b7f5d90e3c13
12	0.0289	1031736	ezcTemplate->process($location = 'example.ezt', $config = class ezcTemplateConfiguration { private $prop
13	0.0294	1036744	ezcTemplateCompiledCode->execute( )

Wrap-up

- First steps: Naïve indexing
- Theory: tokenizing and stemming
- MongoDB text search
- Sorting and collation — languages are difficult
- Using ElasticSearch with MongoDB — awesome match

enough — trough — bough
dough — sought — through
thorough — hiccough — hough — lough

http://joind.in/14372

derick@mongodb.com—@derickr