Zend PHP Conference - Santa Clara, US

Derick Rethans - dr@ez.no

http://derickrethans.nl/talks.php

About Me

- Dutchman living in Norway
- eZ Systems A.S.
- eZ Components project lead
- PHP development
- mcrypt, input_filter, date/time support, unicode
- QA

Introduction

Indexing

Before you can search, you need to index.

Indexing requires:

- Finding the documents to index (crawl)
- Separate the documents into indexable units (tokenizing)
- Massage the found units (stemming)

Share your information

Crawling

- Domain specific: file system, CMS, Google
- Should indicate different fields of a document: title, description, meta-tags, body

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
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)

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□ 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,*
汎用      名詞,サ変名詞,*,*,汎用,はんよう,代表表記:汎用
```

Stemming

Stemming normalizes words:

- Porter stemming
- It's language dependent: snowball
- Several algorithms exist

```
arrival -> arrive
skies -> sky
riding -> ride
rides -> ride
horses -> hors
```

Alternatively, instead of word analysis you can use "sounds like" indexing, by using something like soundex or metaphone:

Word	Soundex	Metaphone
stemming	S355	STMNK
peas	P200	PS
peace	P200	PS
please	P420	PLS

Stemming

Japanese

踊る	odoru	dance
踊らない	odoranai	doesn't dance
踊った	odotta	danced
踊らなかった	odoranakatta	didn't dance
踊れる	odoreru	can dance
踊れない	odorenai	can't dance
踊れた	odoreta	could dance
踊れなかった	odorenakatta	couldn't dance
踊っている	odotteiru	is dancing
踊っていない	odotteinai	isn't dancing

Searching

Different types of searches

- Search words, phrases, boolean: airplane, "red wine", wine -red
- Field searches: title:tutorial desc:feed
- Facetted search:

<a href='http://dev.ezcomponents.org/search'>demo</a>

Different Methods

MySQL FULLTEXT

MySQL has support for full-text indexing and searching:

- A full-text index in MySQL is an index of type FULLTEXT.
- Full-text indexes can be used only with MyISAM tables, and can be created only for CHAR, VARCHAR, or TEXT columns.

MySQL FULLTEXT

Types

Boolean search:

```
mysql> SELECT * FROM articles WHERE MATCH (title,body)
    -> AGAINST ('+MySQL -YourSQL' IN BOOLEAN MODE);
+----+-----------------------+-------------------------------------+
| id | title                 | body                                |
+----+-----------------------+-------------------------------------+
| 1 | MySQL Tutorial        | DBMS stands for DataBase ...        |
| 2 | How To Use MySQL Well | After you went through a ...        |
| 3 | Optimizing MySQL     | In this tutorial we will show ... |
| 4 | 1001 MySQL Tricks     | 1. Never run mysqld as root. 2. ... |
| 6 | MySQL Security        | When configured properly, MySQL ... |
+----+-----------------------+-------------------------------------+
```

Natural search (with or without query expansion):

```
mysql> SELECT * FROM articles
    -> WHERE MATCH (title,body) AGAINST ('database');
+----+-------------------+------------------------------------------+
| id | title             | body                                     |
+----+-------------------+------------------------------------------+
| 5 | MySQL vs. YourSQL | In the following database comparison ... |
| 1 | MySQL Tutorial    | DBMS stands for DataBase ...             |
+----+-------------------+------------------------------------------+
```

MySQL FULLTEXT

- Full-text searches are supported for MyISAM tables only.
- Full-text searches can be used with multi-byte character sets. The exception is that for Unicode, the UTF-8 character set can be used, but not the ucs2 character set.
- Ideographic languages such as Chinese and Japanese do not have word delimiters. Therefore, the FULLTEXT parser cannot determine where words begin and end in these and other such languages.
- Although the use of multiple character sets within a single table is supported, all columns in a FULLTEXT index must use the same character set and collation.

From http://dev.mysql.com/doc/refman/5.0/en/fulltext-restrictions.html

Own Implementation

- We store content differently, so the FULLTEXT MySQL approach doesn't work
- We need support for CJK
- We needed support for other databases

Own Implementation

- Split text into tokens
- Store tokens in a table, uniquely - with frequency
- Store object / word location in a table, next/prev word, order

```
mysql> select * from ezsearch_word order by word limit 250, 4;
+------+--------------+--------------+
| id | object_count | word         |
+------+--------------+--------------+
| 1761 |            1 | associations |
| 2191 |            1 | assurance    |
| 349 |         37 | at          |
+------+--------------+--------------+

mysql> select word_id, word from ezsearch_object_word_link wl, ezsearch_word w
where wl.word_id = w.id and contentobject_id = 145 order by placement limit 4;
+---------+--------+
| word_id | word |
+---------+--------+
|    1576 | puts |
|     926 | europe |
|     349 | at     |
+---------+--------+
```

Own Implementation

Be careful to split with regular expressions - character set and locale issues:

```
<pre><?php
setlocale( LC_ALL, 'nb_NO.utf8');
var_dump( preg_split( '/\W/u', 'blårbærøl er greit' ) );
?>
```

Own Implementation

Be careful to split with regular expressions - character set and locale issues:

```
<pre>
<?php
$string = 'blårbærøl er greit';
$string = iconv( 'utf-8', 'latin1', $string );
setlocale( LC_ALL, 'nb_NO.iso-8859-1');
var_dump( preg_split( '/\W/', $string ) );
?>
```

Own Implementation

Doesn't work very well for huge amounts of content:

```
mysql> use ezno;
Database changed

mysql> select count(*) from ezsearch_word;
212291

mysql> select count(*) from ezsearch_object_word_link;
15551310
```

eZ®
Share your information

Apache Lucene

Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform.

- Implemented in Java
- Provides indexing and searching libraries
- Ranked searching -- best results returned first
- Many powerful query types: phrase queries, wildcard queries, proximity queries, range queries and more
- Fielded searching (e.g., title, author, contents)
- Date-range searching
- Sorting by any field

Zend Lucene

It's a port of Java Lucene to PHP

- Compatible with the Lucene index format
- Provides indexing and searching libraries
- Supports some of the lucene query language
- Support for indexing HTML documents: title, meta and body
- Has support for different field types:
- Keyword: Not tokenized
- UnIndexed: Not indexed
- Binary: Binary data
- Text: Tokenized
- UnStored: Tokenized, but only indexed

Zend Lucene
Indexing Example

Normal:

```
<?php
// Open existing index
$index = Zend_Search_Lucene::open('/data/my-index');

$doc = new Zend_Search_Lucene_Document();
// Store document URL to identify it in search result.
$doc->addField(Zend_Search_Lucene_Field::Text('url', $docUrl));
// Index document content
$doc->addField(Zend_Search_Lucene_Field::UnStored('contents', $docContent));

// Add document to the index.
$index->addDocument($doc);
?>
```

HTML document:

```
<?php
$doc = Zend_Search_Lucene_Document_Html::loadHTMLFile($filename);
$index->addDocument($doc);
?>
```

Zend Lucene
Search Example

With parser:

```
<?php
$index = Zend_Search_Lucene::open('/data/my-index');
$userQuery = Zend_Search_Lucene_Search_QueryParser::parse($queryStr);
$hits = $index->find($query);
?>
```

With API:

```
<?php
$userQuery = Zend_Search_Lucene_Search_QueryParser::parse($queryStr);

$pathTerm = new Zend_Search_Lucene_Index_Term('/data/doc_dir/' . $filename, 'path');
$pathQuery = new Zend_Search_Lucene_Search_Query_Term($pathTerm);

$query = new Zend_Search_Lucene_Search_Query_Boolean();
$query->addSubquery($userQuery);
$query->addSubquery($pathQuery);

$hits = $index->find($query);
?>
```

Apache Solr

Solr is a standalone enterprise search server with a web-services like API. It extends Lucene:

- Real schema, with numeric types, dynamic fields, unique keys
- Powerful extensions to the lucene query language
- Support for dynamic faceted browsing and filtering
- Advanced, configurable text analysis
- Highly configurable and user extensible caching
- Performance optimizations
- External configuration via xml
- An administration interface
- Monitorable logging
- Fast incremental updates and snapshot distribution
- XML and CSV/delimited-text update formats

<a href='http://localhost:8983/solr/admin/'>demo</a>

Marjory

Marjory is a webservice for indexing and searching for documents, utilizing a full-text search engine.

- It is somewhat similar to Solr, but is written in PHP and the underlying architecture allows for using search engines other than Lucene (no other adaptor is implemented yet, though).
- Marjory is based on the Zend Framework and uses Zend_Search_Lucene as the default search engine.

eZ Components' Search component

- As our current solution doesn't scale, we needed something new
- As Lucene is really good, the first attempt was to use the Java bridge
- Then Solr came out
- Which we integrated in "eZ Find"
- Next versions of eZ Publish will use eZ Components

- Support for multiple backends
- Abstract documents
- Support for datatypes
- Rich searching API, including facetted search
- Easy query interface

eZ Components' Search component
Document Definition

```php
<?php
static public function getDefinition()
{
    $n = new ezcSearchDocumentDefinition( 'ezcSearchSimpleArticle' );
    $n->idProperty = 'id';
    $n->fields['id'] =
        new ezcSearchDefinitionDocumentField(
            'id', ezcSearchDocumentDefinition::TEXT );

    $n->fields['title'] =
        new ezcSearchDefinitionDocumentField(
            'title', ezcSearchDocumentDefinition::TEXT,
            2, true, false, true );

    $n->fields['body'] =
        new ezcSearchDefinitionDocumentField(
            'body', ezcSearchDocumentDefinition::TEXT,
            1, false, false, true );

    $n->fields['published'] =
        new ezcSearchDefinitionDocumentField(
            'published', ezcSearchDocumentDefinition::DATE );

    $n->fields['url'] =
        new ezcSearchDefinitionDocumentField(
            'url', ezcSearchDocumentDefinition::STRING );

    $n->fields['type'] =
        new ezcSearchDefinitionDocumentField(
            'type', ezcSearchDocumentDefinition::STRING,
            0, true, false, false );

    return $n;
}
?>
```

eZ Components' Search component

Document Definition, in XML

```
<?xml version="1.0"?>
<document>
    <field type="id">id</field>
    <field type="text" boost="2">title</field>
    <field type="text">summary</field>
    <field inResult="false" type="html">body</field>
    <field type="date">published</field>
    <field type="string" multi="true">author</field>
</document>
```

Setting up the manager:

```
<?php
$backend = new ezcSearchSolrHandler;

$session = new ezcSearchSession(
    $backend,
    new ezcSearchXmlManager( $testFilesDir )
);
?>
```

eZ Components' Search component
Indexing

```
<?php
$session = new ezcSearchSession(
    $backend,
    new ezcSearchXmlManager( $testFilesDir )
);

$a = new Article(
    null, // id
    'Test Article', // title
    'This is an article to test', // description
    'the body of the article', // body
    time() // published
);
$session->index( $a );
?>
```

eZ Components' Search component
Search - API

```
<?php
$session = new ezcSearchSession(
    $backend,
    new ezcSearchXmlManager( $testFilesDir )
);

$q = $session->createFindQuery( 'Article' );
$q->where( $q->eq( 'title', 'Article' ) );
->limit( 5 );
->orderBy( 'id' );

$r = $session->find( $q );

foreach ( $r->documents as $document )
{
    echo $document['document']->title, "\n";
}
?>
```

Share your Information

eZ Components' Search component

Search - Query Builder

```
<?php
$session = new ezcSearchSession(
    $backend,
    new ezcSearchXmlManager( $testFilesDir )
);

$q = $session->createFindQuery( 'Article' );

new ezcSearchQueryBuilder(
    $q,
    'thunderball',
    array( 'fieldOne', 'fieldTwo' )
);

$q->facet( 'title' ); // keyword data field

$r = $session->find( $q );
foreach ( $r->documents as $document )
{
    echo $document['document']->title, "\n";
}
?>
```

eZ®
Share your information

Notes on Performance
Solr vs Zend Lucene

- Both engines are not tuned
- Solr indexes about 25% faster for small documents (one sentence)
- Solr indexes about 200% faster for big documents (64kb)

Future improvements

- More backends: google, marjory, sphinx, xapian, yahoo
- More features: SpellChecker, MoreLikeThis

Resources

Porter algoritihm: http://telemat.det.unifi.it/book/2001/wchange/download/stem_porter.html
Solr: http://lucene.apache.org/solr/
Zend Lucene: http://framework.zend.com/manual/en/zend.search.lucene.html
SnowBall: http://snowball.tartarus.org/
Mecab (In Japanese): http://mecab.sourceforge.net/
These Slides: http://derickrethans.nl/talks.php