(How NOT to (parse "search queries"))

Tim Retout

London Perl Workshop
30th November 2013
Who is this guy?

- Development Manager, CV-Library
  (We do exciting Perl stuff! We're hiring!)
- Southampton.pm
- Debian Developer (incl. pkg-perl team)
- Perl user since 2004
- Lots of other interesting things
Aim

- Interface like popular web search engines
  - Easy to learn for simple queries
  - Sufficiently flexible for advanced users
- Handle weird user input
  - Don't blow up
  - Be tolerant, DWIM, and try to serve results
- Laziness, a.k.a. maintainability
Backend

• Solr
  – Java, based on Lucene
  – HTTP API

• CPAN modules
  – WebService::Solr
  – (though keep an eye on Apache::Solr)
SolrQuerySyntax

- https://wiki.apache.org/solr/SolrQuerySyntax
- Derived from Lucene syntax

```((foo bar))
>>> 400 Bad Request - The response sent by the client was syntactically incorrect```
WebService::Solr::Query

- "SQL::Abstract but for Solr"
- Handles all escaping

```perl
my $query = WebService::Solr::Query->new(
    { -default => ['foo', 'bar'] }
);
```

But how do you turn "(foo OR bar)" into this?
Parsing Search Queries

- Plain keyword searches:
  - [Perl] [Perl developer]
- Boolean logic:
  - [Perl OR developer] [Perl AND developer]
- Phrase searches:
  - ["Perl developer"]
- Grouping:
  - [(Perl AND developer) OR (Python AND Ruby)]
Search::QueryParser

# See also Search::Query::Parser – very similar.
my $qp = Search::QueryParser->new();
my $s = '(foo OR bar) AND baz';

# Parses a query string into a data structure
# suitable for external search engines
my $query = $qp->parse($s)
    or die "Error in query: " . $qp->err;
That output format

{
  "+" => [
    {
      field => "",
      op => "()",
      value => {
        "" => [
          {
            field => "",
            op => ":",
            value => "foo"
          }
        ...
      }
    }
  ...
}
sub _escape {
    my ( $self, $query ) = @_;

    for my $key ( keys %$query ) {
        for my $field ( @{$ $query->{$key} } ) {
            if ( ref $field->{value} ) {
                $field->{value} = $self->_escape($field->{value});
            } else {
                $field->{value} =
                    WebService::Solr::Query->escape( $field->{value} );
                $self->_modify_field($field);
            }
        }
    }

    return $query;
}
..then unparse it...

sub _unparse_query {
    my $self = shift;
    my $q    = shift;

    my @subQ;
    for my $prefix ( '+', '', '-' ) {
        next if not $q->{$prefix};
        push @subQ, $prefix . $self->_unparse_subQ($_) for @{ $q->{$prefix} };
    }
    return join q{ }, @subQ;
}
sub _unparse_subQ {
    my $self = shift;
    my $subQ = shift;

    return '(.{$self->_unparse_query($subQ->{value}).}'.
        if $subQ->{op} eq '()';

    my $quote = $subQ->{quote} || q{);
    return ( $subQ->{field} ? $subQ->{field} . ':' : '' )
        . "$quote$subQ->{value}$quote";
}
Oh wait

Error in query: [foo OR bar AND baz]:
cannot mix AND/OR in requests; use parentheses

Error in query: [([foo OR bar])]:
no matching )

(And have you seen the code?!)
Can we do better?

- Learn about parsers
  - Udacity CS262
- Parse::Yapp
  - Parser code becomes readable
  - More powerful than Parse::RecDescent
- Simple precedence rule for AND/OR/NOT
  - NOT beats AND beats OR
Lexer

```perl
$pseudo->YYData->{INPUT} =~ s/^\s*//;
for ($pseudo->YYData->{INPUT}) {
    s/^\(// and return ('LPAREN', '(');
    s/^\)// and return ('RPAREN', ')');
    s/\1// and return ('TERM', $1);
    s/\1\1// and return ('TERM', $1);
}
return ('', undef);
```
Grammar

%left OR
%left AND
%left NOT

%%

search:  LPAREN search RPAREN  { "( $_[2] )" }
|   search OR search  { "( $_[1] OR $_[3] )" }
|   search AND search  { "( $_[1] AND $_[3] )" }
|   search search %prec AND  { "( $_[1] AND $_[2] )" }
|   NOT search  { "(!$_[2])" }
|   TERM  { "$_[1]" }
;

It still sucks

$ perl -MSearch -e 'Search->new->run()'

foo AND bar OR baz

$ ( ( foo AND bar ) OR baz )

(((foo bar))

>>> Syntax error.

This stuff is designed for compilers, not humans.
Solr's edismax parser

- https://wiki.apache.org/solr/ExtendedDisMax
- Robust, tolerant – designed for user input
- Can be configured to search multiple fields by default

Stop munging stuff in Perl – it might have been necessary before Solr 3.1, but it's not any more.
my $solr = WebService::Solr->new($url);

# WebService::Solr::Query objects are useful for # 'fq' params, but avoid them for main 'q' param.
my $options = {
    fq => [WebService::Solr::Query->new(...)];
};

$solr->search($s, \%options);
((a OR b) - lucene

```java
localhost:8080/solr/select/?q=(a%20OR%20b))&version=2.2&start=0&rows=1
```

**HTTP Status 400 - null**

<table>
<thead>
<tr>
<th>type</th>
<th>Status report</th>
</tr>
</thead>
<tbody>
<tr>
<td>message</td>
<td>null</td>
</tr>
<tr>
<td>description</td>
<td>The request sent by the client was syntactically incorrect.</td>
</tr>
</tbody>
</table>

**Apache Tomcat/6.0.37**
This XML file does not appear to have any style information associated with it. The document tree is shown below:

```xml
<response>
  <lst name="responseHeader">
    <int name="status">0</int>
    <int name="QTime">2</int>
  </lst>
  <lst name="params">
    <str name="mm">0</str>
    <str name="indent">on</str>
    <str name="start">0</str>
    <str name="q">(a OR b)</str>
    <str name="qf">text</str>
    <str name="version">2.2</str>
    <str name="rows">10</str>
    <str name="defType">edismax</str>
  </lst>
</response>
```
Questions?

Tim Retout
<tim@retout.co.uk>