

Information Retrieval

Exercise 2

„Relevance Feedback system for text documents“

System Documentation

Wolfgang Ziegler 0425861

SS 2008

Notes

Unfortunately my group member XXXX has abruptly stopped participating, so I've implemented the system and finished the exercise on my own.

I decided to make this work available to the drupal community, see http://more.zites.net/relevance_feedback. So everyone interested in it, has access to it.

1 Overview

As described in the concept it has been chosen to implement the web-based relevance feedback system for the search of the popular CMS drupal¹. The system is capable of using

- **Implicit Feedback:** Documents viewed by the user are treated as relevant.
- **Explicit Feedback:** The user has the possibility to rate documents as relevant or non relevant.

When the user starts searching feedback is gathered either implicit or explicit. When enough feedback is available the search query is refined and the new results are presented to the user.

¹ <http://drupal.org>

This shows how the search interface looks like when the system is set to use „Explicit feedback“.

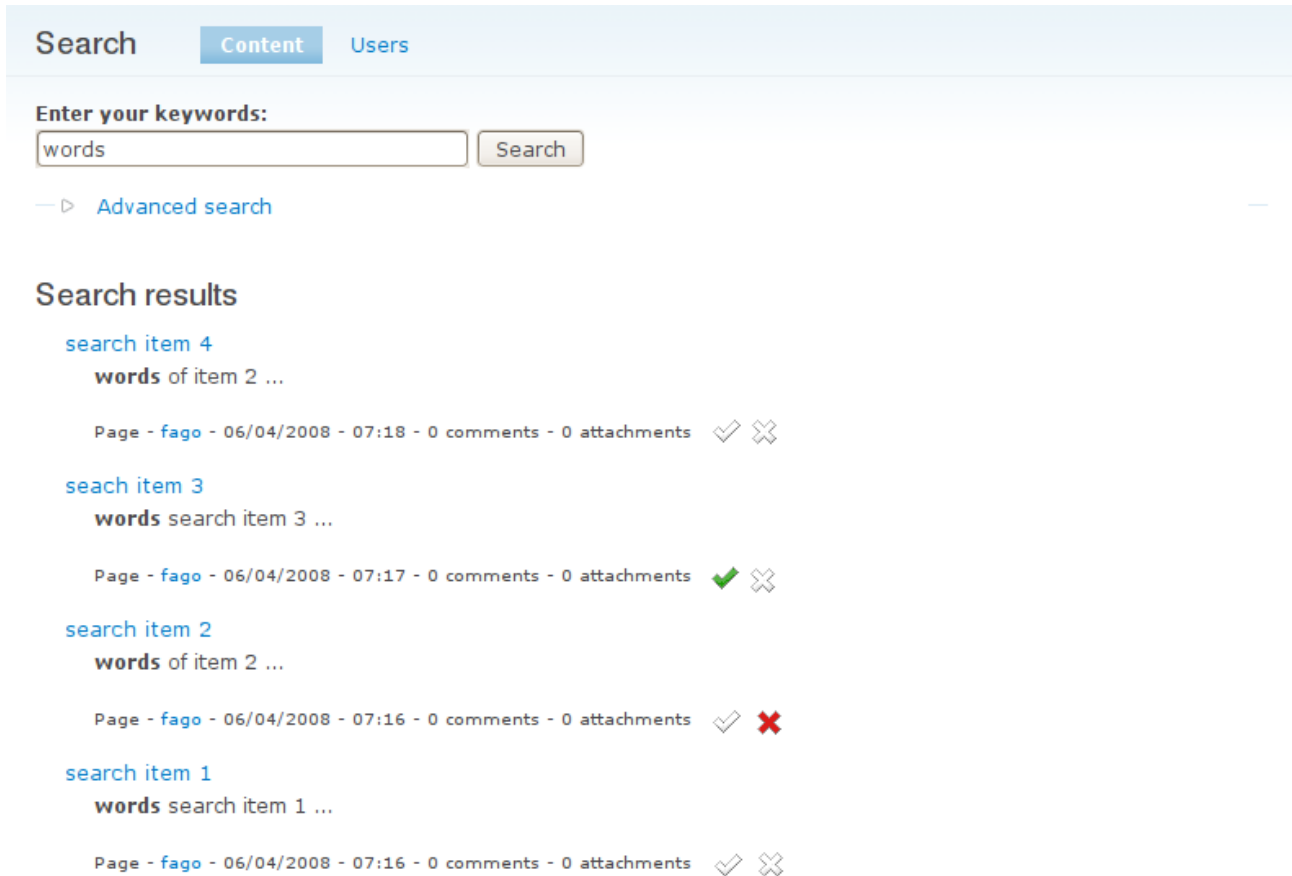


Figure 1: UI when the system is set to use „Explicit feedback“

The feedback information is sent to the system with the help of AJAX, which avoids unnecessary page loads and ensures a good user experience. Once the user has given enough feedback the system refines the search query and redirects the user to the new search results.

The system has been tested with two test collections: First off posts from drupal's support forum² were aggregated by RSS and so imported into a drupal installation. For the second test collection the somlib demo collection has been used.³

2 Technical details

As described in the concept the system relies on the search index built by drupal, which assigns a score to each term. Let's have a closer look how drupal calculates this score.

² <http://drupal.org/forum>

³ <http://www.ifs.tuwien.ac.at/~andi/somlib/download/download/democollection.tar.gz>

About drupal's search index

When drupal indexes a document (aka „Node“ in drupal's terminology) it splits its text up into its words and calculates a score for each word. First off it looks at the HTML tags, in which the word is appearing. For each nested tag the score of the word is increased by a value depending on the tag:

```
<?php
$tags = array('h1' => 25,
              'h2' => 18,
              'h3' => 15,
              'h4' => 12,
              'h5' => 9,
              'h6' => 6,
              'u'  => 3,
              'b'  => 3,
              'i'  => 3,
              'strong' => 3,
              'em' => 3,
              'a'  => 10);
?>
```

Figure 2: HTML tags that increase the score of a word [<http://acquia.com/blog/drupal-search-how-indexing-works>]

Then the absolute position of the word within the document is taken into account. The lower in the document the word is found, the less it will increase the score. This “focus” is calculated by this formula:

$$focus = \min\left(1, \frac{0.01 + 3.5}{2 + position * .015}\right)$$

So it's a decaying value in terms of the amount of unique words up to this point. From 100 words and more, it decays, to e.g. 0.5 at 500 words and 0.3 at 1000 words. [acquia]

The so far calculated score is weighted by this focus. Furthermore the scores of multiple occurrences of a word in a document are accumulated to one value:

$$documentscore(document, word) = \sum score * fokus$$

Additionally drupal analyses links to other documents and indexes the words linked to another

document for the other document. As this is not relevant for the exercise the details are left out here.

Apart from the “documentscore” drupal calculates also a total score, a value which expresses the value of a word throughout the whole index:

$$totalscore(word) = \log_{10} \left(1 + \frac{1}{\max(1, \sum documentscore(document, word))} \right)$$

So the “documentscore” has been used as term frequency and the “totalscore” as inverse document frequency throughout the exercise. Like drupal does the term weight was calculated using the tfidf model throughout the exercise:

$$weight(document, term) = documentscore(document, term) * totalscore(term)$$

Refining the search

The system refines the search when a given number of positive feedback is available. This number is configurable and can be set in the configuration options as shown in figure 3:

Relevance Feedback

How to get feedback:

- Users can mark search results as (non)relevant.
- Treat results viewed by users as relevant.

Refine search when feedback for how many results is available:

After the user has provided positive feedback for e.g. 5 results, his search will be refined.

Algorithm:

- Rocchio algorithm

Threshold for adding new terms to the query:

Enable debug mode

Rocchio algorithm

Alpha value:

Beta value:

Gamma value:

Figure 3: System configuration options

Then an optimised search query is calculated with the help of the feedback and an algorithm. The system makes use of the “Rocchio algorithm”:

$$\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{\vec{d}_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{\vec{d}_j \in D_{nr}} \vec{d}_j$$

Figure 4: Rocchio algorithm [introduction]

q_m is the modified query, q_0 the original query vector; D_r and D_{nr} are the set of known relevant and nonrelevant documents, α , β , and γ are weights. The weights can also be configured through the configuration interface. [introduction]

The modified search query vector holds a weight for each word occurring in the examined documents.

To map from this query vector back to a drupal search request a threshold value has been used, which is also configurable with the help of the configuration options. Words with a weight higher than the threshold are taken into account and accumulated to a new drupal search request. To map from the original drupal search request to the query vector q_0 , each word is initiated with the threshold value.

Once the system has calculated the new modified search query the user is redirected to a new page, showing the search results of the modified query and a notification that the search results have been refined.

3 Installation

The system can be found and tested online at <http://demo.zites.net/search>. The system runs there in “explicit feedback” mode, a threshold of 0.1, three required positive feedbacks and the default settings for the rocchio algorithm.

For running the system with settings it can be installed. To do so, follow these steps:

1. Make sure you have a working environment as required for drupal, e.g. apache + php + mysql as provided by the XAMPP project⁴.

⁴ <http://www.apachefriends.org/de/xampp-windows.html>

2. Unpack drupal-relevance.tar.gz in your webdir and install it like a usual drupal installation. Look at INSTALL.txt to do so.
3. After installation, overwrite your drupal database with the database from the provided mysql dump. Before you do so, you have to gunzip the dump.
4. Once you have done so, you are ready and can log into the site with the user “admin” and the password “admin”. You can find the modules settings at “/admin/settings/relevance_feedback” and drupal's search settings at “/admin/settings/search”. The search interface can be found at “/search”.

The module written for the system can be found in the directory drupal-6/sites/all/modules/relevance_feedback_6. Furthermore it's available in the drupal CVS in my sandbox: http://cvs.drupal.org/viewvc.py/drupal/contributions/sandbox/fago/relevance_feedback

4 Results

The system was tested with both text collections and returned quite good results. It turned out that the common settings for the rocchio algorithm as seen in figure 3 work best. It wasn't as easy to determine the best value for the query threshold value in general, the more similar the relevant results are the better is it to choose higher values. It turned out that 0.1 is a good value if the relevant results are quite similar. If not, a value of 0.5 worked best.

Let's show an example with a threshold of 0.1. We search for “drupal” and restrict the search to drupal forum posts – this unprecise search returns a lot of results. We are interested in results related to the installation of drupal, so we provide positive feedback for such results. After the third given feedback the system presents as the refined search results, as configured:

Search **Content** Users

Your search query was refined for better search results.

Enter your keywords:
 Search

—▷ [Advanced search](#)

Search results

STICKY: Installing Drupal - README
... Installing Modules and Themes Check out Beyond the **basics** for more community contributed documentation, videos, tips and tricks. ... The **Drupal Cookbook** (for **beginners**) **Drupal 6 - Overview**, Installation & Upgrading **Drupal 5 - Overview**, Installation ...

Drupal Forum Post - [fago](#) - 07/01/2008 - 17:28 - 0 comments - ✓ ✕

STICKY: Before you start - README
... of **Drupal** sites check out these links. Success **Stories Drupal** showcase forum **Buytaert - Drupal** sites **Drupal** sites read more ...

Drupal Forum Post - [fago](#) - 07/01/2008 - 17:28 - 0 comments - ✓ ✕

Drupal 6.4 Installation Failure
I get the following error when trying to install **Drupal** 6.4: user warning: You have an error in your SQL syntax; check the ... Files\Apache Software Foundation\Apache2.2\htdocs\drupal-6.4\includes\menu.inc on line 315. user warning: You have an error in ...

Drupal Forum Post - [fago](#) - 09/03/2008 - 13:38 - 0 comments - ✓ ✕

Figure 5: Showing refined search results

As you can see in figure 5, the relevant search results are now at the top of the list – so the results have been improved, great! One can see some other terms added to the (internal) search query highlighted.

When one steps ahead and marks further results as relevant, the system usually reverts the original search query. This probably occurs as the results marked as relevant are too different – so there are no terms appearing in enough results. I think it's hard for a relevance feedback system to give good results when the user marks rather different results as relevant – in contrast, if the user knows exactly what to look for and the relevant results are quite similar, then I think a relevance feedback system can do a good job improving search results.

The influence of the sort order

The drupal search module allows one to alter the ordering of the returned search results.

Content ranking

The following numbers control which properties the content search should favor when ordering the results. Higher numbers mean more influence, zero means the property is ignored. Changing these numbers does not require the search index to be rebuilt. Changes take effect immediately.

Factor	Weight
Keyword relevance	10
Recently posted	0
Number of comments	0
Number of views	0

Figure 6: Search ordering settings offered by drupal

The above results were retrieved with the settings as shown in figure 6. These settings ensure that results with a better keyword match are shown first. It has been tried to add a factor of 3 for the factor “Recently posted”, which works fine as keyword relevance is still more important. However if one increases “Recently posted” and decreases “Keyword relevance” so that the keyword relevance value is lower than the value of “Recently posted”, then the relevance feedback system fails to improve the results. This happens, because then the system prefers recently posted results to better matched results.

5 References

[acquia]

Drupal Search: How indexing works; Robert Douglass; May 9, 2008; <http://acquia.com/blog/drupal-search-how-indexing-works>

[introduction]

Introduction to Information Retrieval; Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze; May 27, 2008