Laboratory exercises for EITN01 WEB Intelligence and Information Retrieval

Anders Ardö
Department of Electrical and Information technology
Lund Institute of Technology

January 23, 2013

Contents

LUND INSTITUTE OF TECHNOLOGY
Lund University

Optional lab: PHP programming

0.1 Objectives

Simple and pragmatic introduction to PHP programming for EITN01.

0.2 Literature

- PHP tutorial: http://www.w3schools.com/php/
- PHP manual: http://www.php.net/manual/en/
- Regular expressions information and tutorial: http://www.regular-expressions.info/

0.3 Tools

All groups will get their own home account WebXX in the Windows group 'LAB' ("`Log on to:`"), where XX (01 ... 10) is your lab-group number. You will get your lab-group number a password for this at the first lab. *Change the password immediately!* This account will be yours for the duration of the course and is used for all the labs.

In the directory `S:\` there are some examples and test-data in `CodeExample.EITN01`, and `testdata.EITN01` respectively. These are also available at the course home-page see 'Laboratory Lessons' in the navigation panel to the left.

Make sure that you save your work between labs in `U:\` *- you need to reuse it later!*

The directory `P:\` contains some program packages like Emacs and Eclipse that you might find useful.

0.4 Assignments

Throughout these assignments use the PHP manual as reference!

0.4.1 Hello World

A PHP scripting block always starts with <?php and ends with ?>. Statements are ended with a ';'. In PHP there are two ways to make comments

- '//' or '#' to make a single-line comment
- '/*' and '*/' to make a large comment block

The trivial 'Hello World'-program looks like this in PHP:

```
<?php
  //printing Hello World
  echo "Hello World\n";
?>
```

Enter this in a file called `hello.php` and run it by:
Start a command window.
Run the program by entering `php hello.php`
What does it say?

0.4.2 Variables

All variables in PHP start with a $ sign symbol, like `$MyVariable` or `$txt`. Variables can be strings, numbers, object, or arrays.

The trivial 'Hello World'-program with variables looks like this in PHP:

```
<?php
  $txt = "Hello World\n";
  //printing Hello World
  echo $txt;
?>
```

Or using the string concatenation operator '.' and a numeric variable (`$num`):

```
<?php
  $num = 42;
  $t1 = "Hello";
  $t2 = "World";
  $txt = $t1 . ' ' . $t2;
  //printing Hello World
  echo $num . ': ' . $txt . "\n";
?>
```

Test these programs on your machine. Modify to use other texts and numbers.

PHP is a loosely typed language, so in PHP, a variable does not need to be declared before being used. And PHP automatically converts the variable to the correct data type, depending on its value.

Look up which operators are available using the manual. Look at what conditional statements are available and their syntax using the manual. There are also a lot of predefined functions available for your use, check out the string functions in the manual. Or use one of the tutorials above.

0.4.3 Arrays

PHP have three kinds of arrays:

- Numeric array - An array with a numeric index
  `$progLang = array('PHP', 'Perl', 'Java', 'C', 'Ada');`
  The variable `$progLang[2]` contains the value '`Java`'.

- Associative array - An array where each key is associated with a value (ie an ordered map `<key>` $\rightarrow$ `<value>`). Example:
  `$wordFreq = array('are' => 5, 'is' => 7, 'each' => 1, 'array' => 0);`
  The variable `$wordFreq['is']` contains the value 7,
  the variable `$wordFreq['array']` contains the value 0,
  the variable `$wordFreq['more']` is undefined.

- Multidimensional array - An array containing one or more arrays
  you can mix and use arrays freely, like `$docWords[$docID]['each']`

Basically an array is used like `$x[ <key> ] = <value> ;` where `<key>` indexes the array and can be an integer or a string. `<value>` can be of any type including a reference to another array.

Load and print out this data-set:

Word	Frequency
<key>	<value>
are	5
is	7
each	1
array	3

```
<?php
  $wordFreq = array('are' => 5, 'is' => 7, 'each' => 1, 'array' => 0);
  foreach ( $wordFreq as $k => $v ) {
    echo "Word $k has the value $v\n";
  }
?>
```

Test it! Add some words and frequencies - test it.

Can you add values to the array as separate statements like `$wordFreq['and']=17;`? Test!

Try the same program but use a numeric array instead of the associative array `$wordFreq`. What happens?

Change the loop-variable to be `( $wordFreq as $v )`. Try it on both types of arrays. What output do you get?

What is the difference between the two ways of looping through an array?

0.4.4 Classes and objects

The basic syntax in PHP object-oriented programming is basically the same as in a language such as Java. Classes (the code that creates an "object") are defined by the class keyword. It can contain functions and member variables.

Here is an example class called `Document` which contains three variables, `$id`, `$length`, and `$termFreq` (which is an array).

```
<?php
class Document {
  var $id;
  var $length;
  var $termFreq = array();
}
?>
```

To create a new object or instance of the class simply do `$myDoc1 = new Document;`. Let's use the class to instantiate two `Document` objects and fill them with some values.

```
<?php
class Document {
  var $id;
  var $length;
  var $termFreq = array();
```

```
}

$myDoc1 = new Document;
$myDoc1->id = 'D1';
$myDoc1->termFreq = array('are' => 5, 'is' => 7, 'each' => 2, 'array' => 1);
$myDoc1->length = 5+7+2+1;

$myDoc2 = new Document;
$myDoc2->id = 'D2';
$myDoc2->termFreq = array('are' => 9, 'and' => 7, 'more' => 3,
                          'associative' => 1, 'array' => 2);
$myDoc2->length = 9+7+3+1+2;
?>
```

Write some programs that, using the above class and instantiated documents:

- prints a list of all words (terms) in the documents
- prints a list of all words in the documents - without duplicates
- prints a list of all words in the documents - without duplicates, with the frequencies per document
- calculates the average length of the documents
- creates a new document instance which is the sum of the other instances
- contains a function that accepts a `Document` object instance and a word as parameters and adds that word the `$termFreq` array (if it's not there) with the frequency 1, or if the already is in the `$termFreq` array just increments the frequency. (Hints - look up function definitions (PHP manual/Language Reference/Functions) and the array function (PHP manual/Function Reference/Variable and Type Related Extensions/Array) `in_array` in the PHP manual.)

0.4.5 Functions and instantiated objects

Add the function `calcLength()` to the class `Document`

```
class Document {
  var $id;
  var $length;
  var $termFreq = array();

  function calcLength() {
    $this->length = 0;
    foreach ( $this->termFreq as $w => $freq ) {
      $this->length += $freq;
    }
  }
}
```

When this function is called for an instantiated object, `$this` is representing the instance and thus provides access to variables local to this instance.

Replace the lines setting document length above (`$myDoc1->length = 5+7+2+1;` etc) with calls to this function (like `$myDoc1->calcLength();`). Verify that you get the same result.

0.4.6 Regular expressions

Read up on regular expressions and how they are used in PHP.

We will use the function `preg_match()` (see PHP manual/Function Reference/Text Processing/Regular Expressions (Perl-Compatible)). It can be called like `preg_match($regexp, $txt, $matches)` where `$regexp` is the regular expression to matched against the text in `$txt` and the array `$matches` will hold all matches. `$matches[0]` will contain the text that matched the full pattern. `$matches[1]` will have the text that matched the first captured parenthesized sub-pattern, and so on. `preg_match()` returns either 0 (no match) or 1 (match).

What will the following program print?

```
<?php
  $txt = 'We are learning regular expressions in PHP';
  $regexp = '/ng (regular) expr/';
  if (preg_match($regexp, $txt, $matches)) {
    echo 'Full match is "' . $matches[0] . "\"\n";
    echo 'First subpattern is "' .  $matches[1] . "\"\n";
  } else {
    echo "No matches\n";
  }
?>
```

Change the word 'regular' to 'regularr'. What will the program print now?

Change the pattern to `'/ng (regular*) expr/'`. What will the program print now?

Look at the program

```
<?php
$lines = file('./Collection.xml',FILE_IGNORE_NEW_LINES);

//Regular expression patterns
$pRid = '/^<id>(.*)<\/id>/';
$pTitle = '/^<title>(.*)<\/title>/';

// Loop through our array, get titles etc
foreach ($lines as $line_num => $line) {
    #preg_match uses regular expressions to match and extract part of a line
    if (preg_match($pRid, $line, $matches)) {
     $rid = $matches[1]; $d='';
     if ($maxrec++ > 5) {break;}
    } elseif (preg_match($pTitle, $line, $matches)) {
      $title = $matches[1];
      echo "The title of doc $rid is '$title'.\n";
    }
}
?>
```

Copy the file `Collection.xml` from `S:\testdata.EITN01` and look inside it.
What do think the program will do? Test what happens when you run the program!
Does the program behave as expected?

0.4.7 Start with Lab 1 ...

1 Information retrieval basics: similarity, tf-idf, recall/precision

1.1 Objectives

The purpose is to become familiar with basic IR models and concepts. After the lab you should be able to implement indexing of document collections and simple searches using Boolean and Vector models. You should also be able to evaluate search results in terms of recall and precision. Experience and understanding of the LSI technique for dimension reduction and concept indexing will be gained.

1.2 Literature

- The course book: 'Modern Information Retrieval', second edition, Chapters 1, 3-4. Equation numbers below refer to this book.
- PHP
  - PHP tutorial: http://www.w3schools.com/php/
  - PHP manual: http://www.php.net/manual/en/
- ... or your favorite programming language manual.
- Regular expressions information and tutorial: http://www.regular-expressions.info/

1.3 Home assignments

In the directory `S:\` there are some examples and test-data in `CodeExample.EITN01`, and `testdata.EITN01` respectively. These are also available at the course home-page see 'Laboratory Lessons' in the navigation panel to the left.

- Documents in our test collection are parsed Web-pages with a simple XML-like formatting that looks like the example below:

  ```
  <id>9401B53573E4C876D85D7158C196E1E8</id>
  <url>http://www.sarracenia.com/faq/faq3045.html</url>
  <title>The Carnivorous Plant...</title>
  <abstract>The Carnivorous Plant FAQ: What can I easily grow in a ... 2005</abstract>
  <links>F3E7C4B1F7F04C8640C26AF993C2CBBC,8BCFA98EB076C151D24BA6A29619F18E,</links>
  ```

  We will use record-number - '<id>' as document identifier and title - '<title>' as document text. Later document text will be complemented by adding abstract - '<abstract>'. The web linking structure (out-links) are recorded in '<links>' using document identifiers.

- Study and familiarize yourself with the example code in `CodeExample.EITN01/main.php` and `CodeExample.EITN01/IR.php.inc`.

  The PHP program `main.php` reads a file with documents like the one in the example above and extracts record identifier and title. Then it goes on to calculate and print the term-document frequency matrix.

main.php:

```
<?php

include_once('./IR.php.inc');
$coll = new Collection;

$lines = file('./Collection.xml',FILE_IGNORE_NEW_LINES);

//Regular expression patterns
$pRid = '/^<id>(.*)<\/id>/';
$pTitle = '/^<title>(.*)<\/title>/';
//$pAbstract = '/^<abstract>(.*)<\/abstract>/';

// Loop through our array, get titles etc
foreach ($lines as $line_num => $line) {
    #preg_match uses regular expressions to match and extract part of a line
    if (preg_match($pRid, $line, $matches)) {
     $rid = $matches[1]; $d='';
     if ($maxrec++ > 5) {break;}
    } elseif (preg_match($pTitle, $line, $matches)) {
      $title = $matches[1];
      $d = $coll->addDoc($rid, $title, $title);
    }
}

$coll->calcTermDocFreq();
$coll->printFreqMatrix();
?>
```

- What does the function 'file' do?
- What does the line `include_once('./IR.php.inc');` do?
- What does the PHP statement
  `$words=preg_split('/[^a-zA-Z\'-]+/', $text, -1, PREG_SPLIT_NO_EMPTY);`
  do?
- What does the PHP function 'strtolower' do?
- What can you use as array indexes in PHP?
- Write a PHP program that uses the data-structures in the above example (including IR.php.inc) to calculate the tf-idf factor, according to table 3.6 in the course book, for each term in all documents in the collection. Store the calculated tf-idf factors in the vector `$weights` in the `Document` class. Use the provided classes 'Document' and 'Collection' as inspiration for structuring your code and data.

1.4 Tools

All groups will get their own home account WebXX, where XX is your group number. You will get a password for this at the first lab. *Change the password immediately!* This account will be yours for the duration of the course and is used for all the labs.

In the directory `S:\` there are some examples and test-data in `CodeExample.EITN01`, and `testdata.EITN01` respectively.

PHP and/or Perl is recommended and supported. (If you would like to use another programming language (like C or Java) you are free to do so, but with limited support.)

The directory `P:\` contains some program packages like Emacs and Eclipse that you might find useful.

Make sure that you save your work between labs in `U:\` *- you need to reuse it later!*

1.5 Lab assignments

1.5.1 Login and Change your password

I repeat Change your password!!! (by pressing Ctrl+Alt+Del)
This account will be yours for the duration of the course.

Programs developed in one lab will be used later in other labs.

1.5.2 Run your first PHP program

Copy the program `S:\CodeExample.EITN01\hello.php` to your home catalog (`U:\`)

Start a command window.

Run the program by entering `php hello.php`

What does it say?

1.5.3 Indexing

- Make a program that reads 5 documents from the collection in `testdata.EITN01/Collection.xml`, calculates the term frequencies (Document->termFreq) - (this is already done in the function 'Document->importWords' - make sure you understand the code!), and the number of documents containing the term (term-document frequencies Collection->termDocFreq) vectors. Use the example code provided in `CodeExample.EITN01/main.php` to get started. For the initial assignments we will only use the title as document text. Use '<id>' as document identifier.

- Make a print-out (on the screen) of the term/frequency-matrix (words vs documents) for the 5 documents from the collection plus the term-document frequencies vector.

- Print a list of indexed documents (id and title).

- Verify that the program behaves correctly by comparing the document list to the calculated term/frequency matrix.

-

1.5.4 tf-idf weights

- Test your tf-idf program from the home-assignments. Print a term/tf-idf matrix for the 5 document collection.
- Why is the tf-idf weight so high for 'classic' and so low for 'terrarium'? Compare with the term/frequency matrix above.

1.5.5 Similarity

Use the query *'What Nepenthes propagation methods are best?'* in the assignments below. (Hint treat the query as a document in a separate collection.)

- Change the indexing to use the first 20 documents from the collection as the test database initially.

We will implement and test 3 different models in this lab. Choose a reasonable way of displaying results. Start by reviewing the course literature on how to calculate similarities.

Model 1 - Boolean model OPTIONAL *Needed for grades 4 and 5*: Write a program that calculates the similarity between a query and all documents in your collection according to the Boolean model. (If you find it hard to express the above query in the Boolean model, you can use a simplified version of the query.)

Model 2 - Vector model with Boolean weights Write a program that calculates the similarity between a query and all documents in your collection according to the Vector model. Use '1' (term present in document) and '0' (term not present in document) as term weights.

Model 3 - Vector model with tf-idf weights Redo the above but with tf-idf weights for document terms and query terms according to scheme 1 (or 3) in table 3.6.

- Comparison
  - Compare the 3 (2) ways of calculating similarities - observations?
  - Which model compares best with your intuitive feeling for good similarity between documents and query? You have to read some of the documents to be able answer this question!
- Larger documents

  Modify your program to index also the abstract in addition to the title. Now look at your results and answer the questions:
  - Are there any changes in the results? Why?
  - Can you get an intuitive feeling for good similarity between the extended documents and the query based on just looking at the collection file?
- Larger collection

  Modify your program to index all documents in the file `Collection.xml`. Now look at your results and answer the questions:
  - Are there any changes in the results? Why?

– Which model compares best with your intuitive feeling for good similarity between documents and query?

Discuss your answers with the lab-assistant.

1.5.6 Evaluation

In order to get a handle on which model performs best recall/precision values are most often used. In the file `queryRelevance.xml` a few queries and corresponding relevant documents are given for the collection. Use all documents in the collection for these assignments. *Hint if you get problems with memory allocation use a line like* `ini_set("memory_limit","100M");` *in your PHP program.*

- Calculate absolute recall and precision for each of the 3 (2) models above and the collection. (Make sure you get reasonable results (=apply sanity checks) before using your results!)
- Limit the number of results for a query to maximum 10 hits. Now calculate precision and recall for the models
- Limit the number of hits by requiring that the similarity is above a certain limit in order for the document to be included among the hits. Again calculate precision and recall.
- Are there any differences in the values of precision and recall for the above 3 ways of determining the result set?
- Are there any difference in ordering between our indexing models using the above 3 ways of determining the result set?
- Calculate the F-score for the 3 (2) models. Why is that good for comparing search algorithms?

Save your program it will be reused in lab 3.4.

1.6 Conclusions

Save your programs - assignments in the following labs will require you to extend them.

In order to pass the lab, talk to the lab instructor and answer the individual questions included in the lab exercises.

Give a good recipe for determining result-sets based on your experiments in this lab.

2 Link-based ranking, Query languages

2.1 Objectives

The purpose of this lab is to improve your understanding of ranking including how different types of ranking can be combined.

Furthermore the lab will increase your familiarity with different types of query languages from standardized ones like SQL and SRU/CQL to ad-hoc ones used in Web search engines.

2.2 Literature

In order to be able to solve the exercises, you could use the following resources:

- The course book: 'Modern Information Retrieval', second edition, Chapters 5, 7, 11.5. Equation numbers below refer to this book.
- "The PageRank Citation Ranking: Bringing Order to the Web", Page, Lawrence and Brin, Sergey and Motwani, Rajeev and Winograd, Terry (1999).
  http://ilpubs.stanford.edu:8090/422/
- "PageRank", http://en.wikipedia.org/wiki/PageRank
- "Authoritative Sources in a Hyperlinked Environment", Kleinberg, Jon (1999).
  http://www.cs.cornell.edu/home/kleinber/auth.pdf or
  http://portal.acm.org/citation.cfm?id=324140
- Search/Retrieve via URL - SRU and CQL:
  - http://www.loc.gov/standards/sru/
  - http://www.loc.gov/standards/sru/specs/cql.html

  (Beware - database servers may support only version 1.1 or a limited version 1.2)
- SQL: http://dev.mysql.com/doc/refman/5.1/en/index.html

2.3 Home assignments

Answer the following questions:

- Write a program that does a PageRank calculation for the collection from Lab1.
- What do these acronyms stand for: SRU, CQL? Read about them.
- Write an example of an SRU request.
- Write an example SQL question.

2.4 Lab assignments

2.4.1 Ranking

- How do you calculate a tf-idf based ranking?
- Modify your program from Lab1 (or make a new) to also read link-information into your data structure. The Web linking structure (document out-links) is stored in the records in '<links>' as a list of document identifiers of the documents linked to.

  Use the link-information to calculate a PageRank based ranking.

  Hints for implementing PageRank.

```
#Read collection and generate link matrix as a 2dimensional array
# similar to earlier read collection
#   $linkMatrix[$from][$to]
#   skip all links that goes to a page outside the collection
#   count no of non-skiped outlinks per page => used later as L(p)

#Calculate PageRank according to the iterative algorithm in Wikipedia
# PR(p) = (1-d)/N + d * sum_all_pages_linking_to_p(PR1(pl)/L(pl))
#  where d=0.85
#        N=Total no of documents ($tot)
#        PR(p) = new PageRank for page p
#        PR1(p) = old PageRank for page p
#        L(p) = no of outlinks from page p
#        sum is taken over all pages (pl) that links to page p

#Initialize PR1 to 1/N for all p; i.e. SUM(PR)=1

#Iteratively caculate PR until change in values is small enough
while ($change > 0.001) {
    foreach ($linkMatrix as $from => $tolist) {
  #  if ( L(p) == 0 ) { #rank sink - dangling pages!
  #    => random jump to random page (simulate links p to all pages)
  #    => distribute PR(p) over all pages
  #  else use algorithm above - loop through all links from $from
      foreach ($tolist as $to => $tmp) {
        ...
      }
    }

  #  change is sqrt(sum_over_all_p( (PR(p) - PR1(p))*(PR(p) - PR1(p)) ))
  #     (Euclidian length)

  #  Do a sanity check that SUM(PR)=1

  #  Setup for new iteration: move PR -> PR1
  #    0.0 -> PR
}
```

```
#Result in PR1

#Hint - test/debug your implementation on really small artificial graphs where you
#       know what the PageRank should be
```

Does does ranking with PageRank improve your search results?

- OPTIONAL (*Needed for grade 5*): Implement the HITs algorithm and run it on your collection. Do you find any clear hubs or authorities?
- Calculate PageRank (and optionally HITs) for the larger collection in `testdata.EITN01/linkCollection.xml`. If you get memory problems then skip word indexing.
- Compare the rankings between the two collections. Any changes?
- How can you combine tf-idf and PageRank based rankings into one common ranking? Would this be better than either of the other two?

2.4.2 Simple searching

Searching can be done in a standardized way using SRU/CQL. An example working query is (all on one line, without spaces):
`http://lup.lub.lu.se/luurSru/?version=1.1&operation=searchRetrieve&startRecord=1&maximumRecords=2&query=title%3Ddata`

It searches for the word 'data' in the 'title' field, retrieves at maximum 2 records starting from record number 1.

- What does the '`%3D`' in the URL translate to? Help on URL-encodings can be found in http://www.w3schools.com/tags/ref_urlencode.asp.

A target database (LUP) is available at http://lup.lub.lu.se/luurSru. LUP is the acronym for Lund University Publications. In LUP you will find research publications, refereed and un-refereed, and doctoral dissertations from 1996 onwards.

Most of SRU/CQL is supported. For details see http://lup.lub.lu.se/documents/luurSruInfo.html

- Use your browser to test the above query. What do you get in return?
- Develop and test a query that finds all publications written by 'Anders Ardö'. How many are they?
- Develop and test a query that finds all publications with the word 'antenna' in the title, written during the period 2002-2007.
- Develop and test a query that finds all publications about antennas written during the period 2002-2007 (ie not only publications with 'antenna' in the title).

2.4.3 SQL

Device a hypothetical SQL-database structure for a LUP like database. How would the above CQL queries look like when written in SQL for your hypothetical database?

What is the main difference between SQL and CQL?

2.4.4 Search Engine query languages

- Look up the query languages for Google and Bing (MSN Live search). Make a short list of syntax and search possibilities for each.
- Compare them. Expressiveness, possibilities, compliance to any standard, etc.
- Are you missing something? Or wishing for other extended possibilities?

2.5 Conclusions

In order to pass the lab, talk to the lab instructor and show that you have done and understood the applications in the assignments.

3 Text pre-processing for indexing

3.1 Objectives

In this lab you will implement and test a number of text pre-processing techniques in order to see their effect on retrieval performance. Furthermore you will learn about feature extraction.

3.2 Literature

- Read up on text pre-processing techniques in the course book: 'Modern Information Retrieval', second edition, Chapters 6, 8. Equation numbers below refer to this book.

3.3 Home assignments

- Why is text pre-processing used?
- Modify your program from Lab 1 to use a stop-word list.
  - What is a stop-word list and what is it used for?
  - Find a suitable stop-word list on the Internet.
  - Where in your code is it best located?
  - Prepare so that several other text pre-processing filters can be done in a pipeline.
- What is stemming?
- Make a list of as many text pre-processing methods you can think of. Here is a start (in addition to the above): spell checking, noun phrase extraction, word bigrams and trigrams, synonyms, named entities. Look up the main idea behind each of the methods in your list.

3.4 Lab assignments

For these assignments you should use the larger collection `cran600_Collection.xml` together with it's associated queries `cran_queryRelevance.xml` (you can still develop and test your programs using the smaller collection, but remember to make measurements with the larger collection!).

- Start by filling in table 3.4.3 below with values using your program from lab 1.5.6. Redo your measurements using the larger collection.

3.4.1 Stop-word list

- Test your stop-word list filter from the home assignments.
- How does it affect retrieval performance (and how do you test that?)?
- Fill in table 3.4.3.

3.4.2 Other text pre-processing methods

- Select and implement at least 2 other text pre-processing methods from the list compiled in your home assignments.
- Test your retrieval performance for each method separately and together (at least for some combinations). Fill in table 3.4.3.

3.4.3 Performance evaluation table

Recall/Precision score for the top X results. Select a reasonable value for X.

Ranking method	Text pre-processing methods	Query 1	Query 2	Query 3	Query 4
tf idf	-				

- Draw a recall/precision curve (similar to figure 4.2 in the course book) for one of the queries in the table. What can you learn from this curve?

3.5 Conclusions

Save your programs - assignments in the following labs will require you to extend them.

In order to pass the lab, talk to the lab instructor and answer the individual questions included in the lab exercises.

Summarize the advantages and disadvantages of the text pre-processing techniques you have implemented.

Why did you choose not to implement some techniques?

4 Concepts using LSI; Document classification using SVM

4.1 Objectives

Detailed look at interpretation of LSI concept space. To gain experience with trained document classification using Support Vector Machines (SVM) and its performance.

4.2 Literature

- LSI http://en.wikipedia.org/wiki/Latent_semantic_analysis
- SVM http://en.wikipedia.org/wiki/Support_vector_machine
- T. Joachims, Text Categorization with Support Vector Machines: Learning with Many Relevant Features. Proceedings of the European Conference on Machine Learning, Springer, 1998. http://www.joachims.org/publications/joachims_98a.pdf

4.3 Home assignments

- How do you calculate the similarity of a query and a document in a LSI reduced dimension space?
- How can you find concepts in LSI?
- Why does SVM need to be trained before it can do classifications?
- Read up on the use of LIBSVM. Decide what you will use as features for SVM training - compare with lecture notes.

4.4 Tools

- LIBSVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/ and Appendix A.
- `svd` is a C program for computing the Singular Value Decomposition (SVD) of large sparse matrices. It is available on the Lab computers and based on SVDLIBC (for documentation and download see http://tedlab.mit.edu/~dr/SVDLIBC/). A PHP function to use SVD from a PHP program is available. See `S:\CodeExample.EITN01/`.)

4.5 Lab assignments

4.5.1 Document classification with Support Vector Machines - SVM

What term weights are best used as feature values for SVM?

- Check what input format LIBSVM needs and convert data from the collections to this input format.
- Use LIBSVM to make a simple SVM classifier trained (use SVM parameters from the lectures) with positive examples from your collection `testdata.EITN01/CP.xml` and negative examples from `testdata.EITN01/nonCP.xml`. You have to have positive and negative examples in the same file when training.
- Evaluate how good your classifier is using the documents in `testdata.EITN01/testSVM.xml`.

4.5.2 For the ambitious student - Latent Semantic Indexing

(*Needed for grades 4 and 5*).

- Make a reduced dimension (use 10 dimensions) model for your collection using LSI. Use the small collection while developing your code.

  Hints: Look at "Latent Semantic Indexing (LSI) A Fast Track Tutorial" (http://www.miislita.com/information-retrieval-tutorial/latent-semantic-indexing-fast-track-tutorial.pdf).
  Also consider the need for utility functions like `matrixMultiply` and `invertDiagonalMatrix`.

  The program `svd` is available on your Lab computers and can be used with the routines provided in the PHP classes 'IRutils' and 'Collection'. Study the functions `runSVD, readS, readUt, saveWeightMatrixCompressed,` and `mapDocsTerms2numbers` first.

- Calculate cosine similarity between queries and docs as before, but use the reduced concept vectors from the LSI model. Remember that both the document vectors and query vectors have to be transformed into the concept space.

- Using the larger collection experiment with the number of dimensions in the range 50 - 150.

- How does results compare to the best results of earlier models? Complement the table in lab 3.4.3 with your results here.

- Is it possible to determine which is best?

4.5.3 For the ambitious student - Concepts using LSI

(*Needed for grade 5*).

- Try to determine a few concepts (expressed in indexed words) from your LSI analysis earlier.

- Try to put names to your concepts?

- Are the concepts meaningful?

4.6 Conclusions

Save your programs - assignments in the following labs will require you to extend them.

In order to pass the lab, talk to the lab instructor and answer the individual questions included in the lab exercises.

5 Browsing vs searching, Search Engines vs meta-search

5.1 Objectives

Learn when to use browsing and when to use searching based on what you are looking for. Understand how a meta-search engines work compared to normal search engines like Google and Yahoo. Compare performance of different retrieval systems.

5.2 Literature

- The course book: 'Modern Information Retrieval', second edition, Chapter 11 - 12.
- Meta-search engines:
  `http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/MetaSearch.html`

5.3 Home assignments

None.

5.4 Lab assignments

5.4.1 Compare Search Engines to meta-search engines

Compare search engines - select one query and send it to these search engines

Google `http://www.google.com/`

Yahoo `http://www.yahoo.com/`

MSN/Bing `http://www.msn.com/`

- Compare overlap and relevance ordering for all the search engines. Hint: You might have some help from
  `http://www.thumbshots.com/Products/ThumbshotsImages/Ranking.aspx`
- Test the meta-search engines

  Dogpile `http://www.dogpile.com/`

  DuckDuckGo `https://duckduckgo.com/`

  WolframAlpha `http://www.wolframalpha.com/`

  Include them in the comparison you did to individual search engines above.
- Which is the best search engine?
- (How many of the hits did you use? Why?)

5.4.2 Compare browsing to searching

- Find "software for automatic assembly code optimization on Intel CPU's" using browsing at Dmoz and searching via a search machine. Which was easiest/best?
- Find "documentation for the Zebra Z39.50 Search Engine" using browsing at Dmoz and searching via a search machine. Which was easiest/best?
- Find "public domain software for doing data mining" using browsing at Dmoz and searching via a search machine. Which was easiest/best?
- For what type of problems would you use browsing and when searching?

5.4.3 Who does the best retrieval engine

- Test your best combination of methods to do retrieval from our test collection (`collection.xml`). Enter your top 10 results including their document ids in the Web-page `http://dja.eit.lth.se/EITN01/BestIR.php`

  Use the query your lab-assistant gives you.
- Which group gets the best result?

5.5 Conclusions

In order to pass the lab, talk to the lab instructor and answer the individual questions included in the lab exercises.

Can query re-formulation (but still keeping the intended question) be used to improve results?

A How to use LIBSVM tools

Extensive documentation for LIBSVM is available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/

LIBSVM have a learning module (svm-train) and a classification module (svm-predict). The classification module can be used to apply the learned model to new examples. See also the examples below for how to use svm-train and svm-predict.

svm-train is called with the following parameters:

```
svm-train [options] training_set_file model_file
options:
-s svm_type : set type of SVM (default 0)
        0 -- C-SVC
        1 -- nu-SVC
        2 -- one-class SVM
        3 -- epsilon-SVR
        4 -- nu-SVR
-t kernel_type : set type of kernel function (default 2)
        0 -- linear: u'*v
        1 -- polynomial: (gamma*u'*v + coef0)^degree
        2 -- radial basis function: exp(-gamma*|u-v|^2)
        3 -- sigmoid: tanh(gamma*u'*v + coef0)
        4 -- precomputed kernel (kernel values in training_set_file)
-d degree : set degree in kernel function (default 3)
-g gamma : set gamma in kernel function (default 1/k)
-r coef0 : set coef0 in kernel function (default 0)
-c cost : set the parameter C of C-SVC, epsilon-SVR, and nu-SVR (default 1)
-n nu : set the parameter nu of nu-SVC, one-class SVM, and nu-SVR (default 0.5)
-p epsilon : set the epsilon in loss function of epsilon-SVR (default 0.1)
-m cachesize : set cache memory size in MB (default 100)
-e epsilon : set tolerance of termination criterion (default 0.001)
-h shrinking: whether to use the shrinking heuristics, 0 or 1 (default 1)
-b probability_estimates: whether to train a SVC or SVR model for probability
   estimates, 0 or 1 (default 0)
-wi weight: set the parameter C of class i to weight*C, for C-SVC (default 1)
-v n: n-fold cross validation mode
```

The input file `training_set_file` contains the training examples. The first lines may contain comments and are ignored if they start with #. Each of the following lines represents one training example and is of the following format:

```
<line> .=. <target> <feature>:<value> <feature>:<value> ... <feature>:<value>
<target> .=. +1 | -1 | 0 | <float>
<feature> .=. <integer> | "qid"
<value> .=. <float>
```

The target value and each of the feature/value pairs are separated by a space character. Feature/value pairs MUST be ordered by increasing feature number. Features with value zero can be skipped.

In classification mode, the target value denotes the class of the example. +1 as the target value marks a positive example, -1 a negative example respectively. So, for example, the line

-1 1:0.43 3:0.12 9284:0.2

specifies a negative example for which feature number 1 has the value 0.43, feature number 3 has the value 0.12, feature number 9284 has the value 0.2, and all the other features have value 0.

The result of svm-train is the model which is learned from the training data in `training_set_file`. The model is written to `model_file`. To make predictions on test examples, svm-predict reads this file. svm-predict is called with the following parameters:

```
svm-predict [options] test_file model_file output_file

options:
-b probability_estimates: whether to predict probability estimates,
   0 or 1 (default 0); for one-class SVM only 0 is supported
```

The test examples in `test_file` are given in the same format as the training examples (possibly with 0 as class label). For all test examples in `test_file` the predicted values are written to `output_file`. There is one line per test example in `output_file` containing the value of the decision function on that example. For classification, the sign of this value determines the predicted class.