Extracting and Integrating Meta-data from online sources

A set of examples leading to improved course and instructor selection

Andreas Kokkalis
and
Adrian C. Prelipcean

June 7, 2014

Project for IK2553 performed at Department of Communication Systems

Examiner and academic adviser
Professor Gerald Q. Maguire Jr.

KTH Royal Institute of Technology
School of Information and Communication Technology (ICT)
Department of Communication Systems
SE-100 44 Stockholm, Sweden

Abstract

The recent advances in Natural Language Processing (NLP), association based algorithms and recommendation systems enabled suppliers to personalize the content offered to their clients / subscribers. However, their use has been limited to either the industrial environment or to the research one, leaving the education environment lacking when it comes to such tools. This report focuses on exploring the use of recommendation systems in education. First, it presents methods that are suitable for extracting meta-data that can be used to derive certain associations and / or suggestions. Second, it proposes a method that suggests courses and / or tutors based on a student's interest. Third, it provides *DocAid*, which is an API suitable for extracting meta-data from different sources. Finally, a web application is built on top of *DocAid* to make the aforementioned functionality available for others to use.

Keywords: keyword extraction; keyphrases extraction; recommender systems; suggestion; algorithms;

Sammanfattning

De senaste framstegen inom Natural Language Processing (NLP), föreningsbaserade algoritmer och rekommendationssystem möjlig leverantörerna att anpassa det innehåll som erbjuds till sina kunder / abonnenter. Dremot har användningen varit begränsad till antingen den industriella miljön eller till forskning en, lämnar utbildningsmiljön saknas när det kommer till sådana verktyg. Denna rapport fokuserar på att utforska anvndningen av rekommendationssystem i utbildningen. Först presenterar det metoder som är lämpliga för extraktion av meta-data som kan användas fr att härleda vissa föreningar och / eller förslag. För det andra föreslås en metod som tyder på kurser och / eller handledare som bygger på en studerandes intresse. För det tredje ger det *DocAid*, vilket är ett API som lämpar sig för att extrahera metadata från olika källor. Slutligen är en webbapplikation byggd ovanpå *DocAid* för att göra den tidigare nämnda funktioner tillgängliga för andra att använda.

Nyckelord: nyckelord extraktion; nyckelfras extraktion; rekommenderat system; frslag; algoritmer

Contents

Abstract i

Sammanfattning ii

List of Acronyms and Abbreviations vii

1 Introduction 1
1.1 General introduction to the project . . . 1
1.2 Background . . . 1
1.3 Problem statement . . . 1
1.4 Problem . . . 2
1.5 Goal . . . 2
1.6 Structure of this report . . . 3

2 Methods 4
2.1 Data Collection . . . 4
2.2 Acronym extraction . . . 4
2.3 Keyphrase and keywords extraction . . . 5
2.3.1 Naive approach . . . 5
2.3.2 Automatic keyphrase extraction . . . 6
2.4 Course and instructor suggestion . . . 6
2.5 Evaluation . . . 7
2.5.1 Evaluation of acronym extraction . . . 7
2.5.2 Evaluation of keywords and keyphrases extraction . . . 7
2.5.3 Evaluation of course and instructor suggestion . . . 8

3 Implementation 9
3.1 System Architecture . . . 9
3.2 Modules . . . 11
3.2.1 Extract acronyms . . . 11
3.2.2 Extract frequently used (key)words . . . 11
3.2.3 Extract keyphrases . . . 12
3.2.4 Translate content . . . 12
3.2.5 Read documents . . . 12
3.2.6 Parse web pages . . . 13
3.2.7 Course XML parser . . . 13
3.2.8 Course web page parser . . . 14
3.2.9 Suggest courses . . . 14
3.2.10 Suggest tutors . . . 15
3.3 Web Application . . . 15

4 Results and discussion 26
4.1 Acronym extraction . . . 26
4.2 Keywords and keyphrases extraction . . . 28
4.3 Course and tutor suggestion . . . 32

5 Conclusions and future work **33**
5.1 Conclusions 33
5.2 Future work 33

List of Figures

1 *DocAid* System achitecture 10
2 *DocAid* web application architecture 16
3 *DocAid* web application - acronym extraction from a URL 16
4 *DocAid* web application - acronym extraction from a document . 17
5 *DocAid* web application - keywords extraction form 18
6 *DocAid* web application - keywords extraction from a document 18
7 *DocAid* web application - keywords extraction from a URL . . . 19
8 *DocAid* web application - filtered keywords extraction from a document 19
9 *DocAid* web application - unfitlered keywords extraction from a document 20
10 *DocAid* web application - keyphrase extraction form 21
11 *DocAid* web application - keyphrase extraction from a URL . . . 21
12 *DocAid* web application - keyphrase extraction from a document 22
13 *DocAid* web application - course advisor form 23
14 *DocAid* web application - suggested courses 24
15 *DocAid* web application - suggested tutors 24
16 *DocAid* web application - course recommendation details 25

List of Tables

1 Acronym extraction evaluation 26
2 Acronym spelling evaluation 27
3 Acronym spelling on first use evaluation 27
4 Evaluation of the keyword extraction - the naive approach - algorithm 29
5 Number of suggested keywords 30
6 Evaluation of the keyphrase extraction algorithm 31
7 Distribution of the suggested keyphrases 32

List of Acronyms and Abbreviations

AFP	Acronym Finding Program
Agg.	Aggregate
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CMS	Course Management System
DOC	Microsoft file extension for word processing documents
DOM	Document Object Model
HTML	HyperText Markup Language
KEA	Keyphrase Extraction Algorithm
KTH	Kungliga Tekniska högskolan
LMS	Learning Management System
NLP	Natural Language Processing
ODT	OpenDocument file extension for word processing documents
PDF	Portable Document Format
PL/pgSQL	SQL Procedural Language
SQL	Structured Query Language
TLA	Three Letter Acronym
TXT	text file
URL	uniform resource locator
XML	Extensible Markup Language

1 Introduction

This project concerns analyzing meta-data extraction techniques for obtaining information that can be used by recommender systems in an educational context. This chapter states the problem and the tasks we desire to achieve. Section 2 provides the methodology used by this project. Section 3 offers an overview of the implemented system. Section 4 discusses the results we obtained and Section 5 concludes the report.

1.1 General introduction to the project

This project began with the goal of building a course suggestion mechanism based upon user's interests and existing information from the university's course management and learning management systems. Because of privacy reasons which limited access to other students' record, this problem was generalized to provide support for a much larger range of activities. Specifically the problem to be addressed is "Extracting and Integrating Meta-data from on-line sources". However, the examples that will be used to illustrate many of the key functionalities will be functions that could lead to improved course and instructor selection

1.2 Background

Course Management Systems (CMS) and Learning Management Systems (LMS) play an important role in modern education. CMS focuses on the automatic administration of courses, tasks which may include editing course content, course registrations, tracking learning progress, and homework submission [34]. On the other hand, LMS can include the CMS tasks but also includes features for online collaboration (i.e., conversations, feedback) and may provide personalized content [25, 34]. KTH Royal Institute of Technology [10] uses different educational tools such as KTH Social [12], Bilda [3], Daisy [7], and LADOK [9]. These systems are supported by large databases and storage systems which contain information regarding course material, academic records of users, and registration history.

1.3 Problem statement

The most commonly used platform for course selection at KTH is KTH Social, which provides a search engine for finding and retrieving course information. The search engine has two main functions: searching by course name and advanced search. The latter filters results based upon the department that offers a course and when the course is offered. However, this search engine requires input from the user, which implies that the user already has knowledge of the courses offered by the different departments. An alternative approach would be to assist the student in course selection based upon personalized content acquired from existing and relevant data (for example from CMS, LMS, and

other sources). The underlying information that resides in these systems can be exploited to provide a set of suggestions tailored to a particular user's interests, prior registration history, and in keeping with the student's own study program (i.e., the student's plan of study leading to a particular degree).

1.4 Problem

This project investigates the different components for building a course suggestion mechanism. The first component is the input, which varies from user specified academic interests to user registration history extracted from the existing CMS and LMS systems. The second component is responsible for pre-processing the input to provide a set of relevant keywords. Such input can be acquired from course web pages, documents (i.e ,transcripts of records) and other CMS / LMS data. The final component takes advantages of user and processed input and produces a set of suggestions using association rules. The major components are:

- User input (e.g., URL of an HTML web page, document),
- Parse input to text source,
- Meta data extraction from text source, and
- Course and instructor suggestions built from meta data similarities of text source and course data.

1.5 Goal

As stated above, the problem that will be addressed in this project is "Extracting and Integrating Meta-data from on-line sources". The goal of this project is to analyze and implement the components mentioned in section 1.4. The following sub-goals correspond to the steps modules that implement these components.

- Acronym extraction module
  for a text source - URL or uploaded document.
- Keyword extraction module
  for a text source - URL or uploaded document.
- Key phrase extraction module
  for a text source - URL or uploaded document.
- Document analysis module combines the previous modules for a given text source.
- Course Advisor - uses the document analysis module to extract meta data from various sources (i.e documents, registration history, user's interests) and outputs a list of course and instructor recommendations.

1.6 Structure of this report

The remainder of this report is organized as follows. Section 2 describes what methods will be used to implement solutions for the tasks presented in Section 1.3. Section 3 shows an overview of the developed system that provides the desired functionality, its modules and *DocAid*, which is the web application we implemented for this project. Section 4 discusses the system's benefits and limitations. Section 5 provides discusses possible extensions and concludes the report.

2 Methods

In this section, we will present the details about how the tasks defined in Section 1.3 can be approached and evaluated. First, we present the data that was collected for this report. Second, we present the subgoals – together with a literature review of corresponding papers that have treated similar problems – in the following order: acronym extraction, keyword extraction, keyphrase extraction, and the course advisor. The section concludes with the methods used for evaluating the developed algorithms.

2.1 Data Collection

The data used in this project consisted of data specific to each course. This data will be extracted from the KTH public Application Programming Interface (API)[11]. The KTH web services can provide information regarding any course's status (e.g., is the course available, the course's grade scale, the course's status, the course's title, number of credits, name of the department responsible for the course, the academic level of the course, recruitment code, etc.). Additionally, information can be derived from each course's web page and from the documents accessible from this web page. This should enable us to learn what acronyms are used in the course, the most commonly used words, etc. We decided to merge both sources of information, as we found that in practice the course description provided by the KTH API was limited and / or absent in some cases.

2.2 Acronym extraction

Different methods to extract acronyms from texts have been developed. An example of an implementation of one method is the Acronym Finding Program (AFP) [35]. AFP uses inexact pattern matching applied to the text surrounding a potential acronym that computes probability for the word of being an acronym. Another example is the Three Letter Acronym (TLA) [37], which uses a set of heuristics for acronym detection while comparing the first three letters of each word with the possible acronym output. A third approach[14] again uses a set of heuristics to develop a rule-based method. Finally, Yeates, Bainbridge, and Witten [38] propose a new algorithm for identifying acronyms that uses a compression-based identification method.

Due to limited time and the fact that we could not find an (open-source) implementation for any of the methods mentioned above, we devised an algorithm similar to [14] and [37], which is based on a set of heuristics, namely:

- acronyms are in upper case,
- acronyms contain initials of most of the words in their definition,

- stop words * are present in the definition of an acronym only when their first letter is upper case (e.g. MOD can stand for Ministry Of Defense or $m$ean $o$ptical $d$ensity but can not stand for Ministry of Defense†), and
- in order for a sequence of characters to be an acronym, the number of letters has to be strictly greater than the number of digits (we arrived at this consideration after closely analyzing documents that contained physical measurements such as geographical measurements, 45°N, or physical measurements 120VA).

After identifying the acronyms, the next step is identifying their spelling (if present) and whether the acronym was spelled out when it was first used or later in the document. To do this, the algorithm searches for n words (that are not stop words) before the acronym and for n words after the acronym, where n represents the number of letters contained by the acronym. This implies that the algorithm checks if any ith word in the list (while ignoring stop words) corresponds to the ith letter of the acronym and, if this is true for all elements of either of the pre- or post-acronym word list, it implies that the acronym is spelled.

2.3 Keyphrase and keywords extraction

Keyphrases summarize a document's contents and are usually assigned by the author(s) of the document. As noted in [36], there are two different approaches to keyphrases, namely keyphrase assignment, which selects phrases from a controlled vocabulary that describes a document, and keyphrase extraction, which uses lexical and information retrieval techniques to extract keyphrases from the document's content.

Keywords, are single words used by authors in a paper that describe the paper's content.

2.3.1 Naive approach

A naive approach to identifying keywords is to separate a text into words and retrieve the most frequently mentioned ones. However, given the fact that words often are derived from a stem‡, we decided on obtaining the most commonly used stems and classify the words derived from them as keywords. Several methods to extract stems have been developed (e.g., Hammarström [17] extracts stems in an unsupervised manner, Porter [28] uses a minimal length based on the number of consonant-vowel-consonant string remaining after the removal of a suffix and is optimized for the English language, and Snowball[29], which is a system where stemming rules can be specified) out of which we decided to use

*We define stop words as words that are frequently occurring but meaningless in terms of information retrieval [21].

†In this case, "of" is only a stop word when its first letter is in lower case.

‡A stem is a form to which affixes can be attached [33]

Porter's stemming algorithm as it is implemented and available from Apache Lucene [1]

2.3.2 Automatic keyphrase extraction

Several domain specific keyword extraction algorithms have been developed. One of these, KEA [36, 15] is a practical automatic keyphrase extraction algorithm. MAUI[22] is freely available and it offers features such as topic indexing with Wikipedia [23]. The heuristics behind KEA and MAUI allow for keyphrases to have a maximum number of three words and minimum of one word, which implies that a keyphrase can be a keyword but it is not limited to this aspect. The algorithm is based on two features: (1) a phrases' frequency in a document compared to how much it is generally used and (2) the distance into the document until the phrase appeared for the first time.

2.4 Course and instructor suggestion

Although suggestion systems have been thoroughly analyzed and explored (see [6, 5, 16] as examples), to the authors' knowledge, the use of such systems has not been analyzed and / or evaluated in an educational context. We wish to generate suggestions for one or more courses that best fits a student's interests and to identify suitable faculty members based upon a student's interest. The suggestions for courses and faculty members should be limited to those in the area relevant to the student's interest and their study plan.

We propose a system that receives as input either a list of keywords or a set of documents that are of interest to the user and / or the registration history of a student. The system outputs a list of courses that the students might find interesting, together with a list of faculty members that teach in the area relevant to the student.

The algorithm first generates a list of relevant keywords, a list of relevant keyphrases, and a list of acronyms from the input and then searches in the database for courses that have similar acronyms, keywords, and / or keyphrases. We use a similarity computation based upon a variation of the Trigram algorithm [20], which groups every three consecutive characters from a string (in this case a trigram) and measures the similarity between two strings by counting the number of trigrams they share. Alternatives to the Trigram algorithm include calculating the normalized Levenshtein distance [39] (which is the number of minimum-weight series of edit transformations that transforms the first string into the second one) and the Ratcliff/Obershelp pattern recognition algorithm [31] (which computes the similarity of two strings as the number of matching characters divided by the total number of characters).

For each keyword from the user's interest list, the algorithm finds the most similar match in the list that contains keywords from all courses and the similarity measures is assigned to that keyword. The similarity measures are summed up and divided by the number of items in the user's list of keywords to compute the keyword weight. Similar operations are performed to obtain the keyphrase

weight. The acronym weight is obtained by dividing the number of exactly matched acronyms[§] by the number of items in the user's list of acronyms. These three weights are summed up to obtain the total weight, which is then used to order the courses by relevance (i.e., based upon this score).

2.5 Evaluation

This subsection describes the procedures that will be used to evaluate the precision of the acronym extraction algorithm and the keyword and keyphrase extraction algorithm. It concludes with a hypothetical description of how the recommendation system could be evaluated.

2.5.1 Evaluation of acronym extraction

For the analysis of the acronym extraction algorithm, the acronyms from 10 technical reports have been manually extracted from the "List of acronyms and abbreviations" preface page of each report[¶]. These acronyms have been put in a list which is considered the "ground truth". The automated acronym extraction is performed on the same 10 technical reports (excluding the list of acronyms and abbreviations preface page(s)) and the following measurements are performed: the number of acronyms correctly identified as acronyms, the number of words falsely identified as acronyms and the number of acronyms falsely identified as words. The proposed model takes a sequence of characters and classifies it as either an acronym or not. The percentage of acronyms in a technical paper or report is small and a basic model that always classifies a sequence of words as not being an acronym (in our initial measurements, the accuracy of the baseline model was higher than 99%) would seem to have good performance. To measure the proportion of correct measurements beyond this baseline, we perform the measurements proposed by Taghva and Gilbreth [35], namely *recall*[‖] and *precision* [**]. Furthermore, we use the F_1 measure, initially introduced by van Rijsbergen [32], which combines recall and precision with an equal weight by using their harmonic mean.

2.5.2 Evaluation of keywords and keyphrases extraction

To perform this evaluation, we used 20 research papers from various research areas, all of which had a list keywords[††]. For each paper, the keywords that have been specified by the author(s) have been used to produce a list that is considered the ground truth for this paper. The keyword extraction algorithm was used to extract a list of keywords and the keyphrase extraction algorithm

[§] An exact match occurs if the string exactly matched in spelling, if the spelling is present.

[¶] The list of these 10 documents is given in Appendix A

[‖] Recall is defined as the number of correct acronym definitions that were found by the algorithm divided by total number of acronyms found in the document

[**] Precision is defined as the number of correct acronyms found by the algorithm divided by the total number of acronyms found by the algorithm

[††] The list of these 10 documents is given in Appendix B

was used to extract a list of keyphrases. To measure the utility of the keyword extraction algorithm, we count the number of exact matches between the ground truth list and the list generated by the algorithm. Furthermore, we define partial matches as a measure extrapolated from the generated list with regard to the ground truth. This partial match measures how many of the keywords from the ground truth list can be derived from a combination of any several items in the algorithm generated list. The same two metrics are computed for the keyphrase extraction algorithm.

2.5.3 Evaluation of course and instructor suggestion

Due to the lack of time, this step was not be performed, but we would suggest designing a form / questionnaire to be filled in by a student when he / she uses the web application. The questionnaire should inquire about his / her degree of satisfaction with the proposed course and / or instructor. An ideal solution would be to replicate a longitudinal study by recruiting a group of master's students who have completed their first semester to test this system. They would receive suggestions (based on their existing registration history) about courses and / or instructors and fill in a form designed to measure the acceptance rate of the system (number of courses selected based on the suggested courses) and the satisfaction of students (number of decisions that they did not regret making based on the provided suggestions).

3 Implementation

This section offers an overview of the system architecture, its modules, and web application. First, we discuss the system's architecture and our considerations while choosing this architecture. Second, we present the main modules and then we illustrate several examples on how they can be used. The section concludes with a presentation of *DocAid*, which is the web application developed for this project using the modules that have been implemented.

3.1 System Architecture

We named the underlying system that contains the meta data extraction modules the *DocAid* Engine. This engine depends on various libraries to implement each module:

- Apache Tika [2] is used to read input documents (PDF, TXT, ODT, DOC) and web pages (HTML web pages) and extract the title and the content of the source as text.
- Palladian [27] is used for extracting keywords from a text source.
- MAUI indexer [22] is used to extract key phrases from a document source.
- Jsoup Java HTML Parser [19] is used to strip HTML tags from various text sources.
- jlangdetect - Language detection API for Java [18] is used to define the language of a text source.

All modules mentioned in section 3.2 are available in the *DocAid* Engine via an application layer. In Figure 1 we present the API wrappers for each module, along with its input, output, and the libraries used. The following list maps the wrappers to each module.

- *CourseIndexBuilderWrp.* Uses modules the modules "Course Web Page Parser"3.2.8 and "Course XML parser" 3.2.7 to populate the database with course meta data.
- *URL/Document parser.* Represents the modules "Parse Web Pages"3.2.6 and "Read Documents"3.2.5. It uses Apache Tika to parse web pages or documents and provide the text output as input to the rest of the modules.
- *KeywordExtractorWrp.* Uses module "Extract frequently used (key)words"3.2.2 along with Palladian to extract keywords, then uses a keyword filter module which removes from the result stop words and character sequences that were falsely defined as keywords (e.g., words that contain numbers such as LAB1).
- *KeyphraseExtractorWrp.* Uses module "Extract keyphrases"3.2.3 along with MAUI to extract keyphrases.

- *AcronymExtractorWrp.* Uses module "Extract Acronyms"3.2.1 to identify acronyms.
- *DocAnalyzerWrp.* Document Analyzer is just a wrapper that combines the the previous three wrappers and provides a summary of the results.
- *CourseAdvisorWrp.* Takes as input a list of text sources (multiple documents) and collects their output from the DocAnalyzerWrp. Optionally takes as input the registration history of a KTH student, extracts the course codes and provides them along with the keyphrases, keywords, and acronyms as input to the recommender, which finally outputs course and tutor recommendations. The recommender is build upon modules "Suggest courses"3.2.9 and "Suggest tutors"3.2.10.

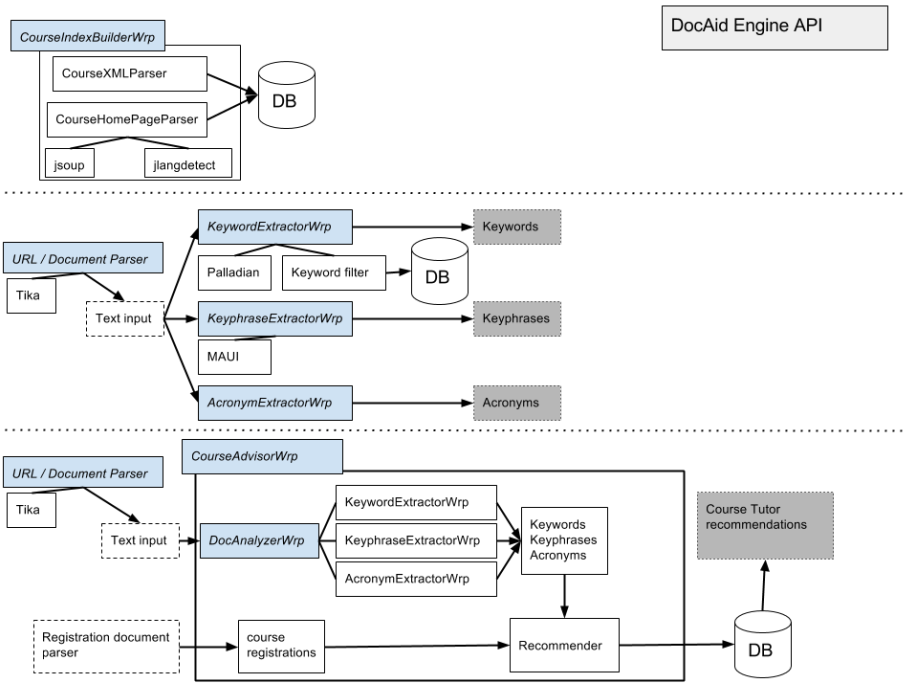


Figure 1: *DocAid* System achitecture

3.2 Modules

This section will introduce the developed modules, the input, output and usage of each module.

3.2.1 Extract acronyms

Data input This module takes as input text as a string.

Data output The module generates a list of acronyms, which contains information regarding each acronym's spelling and whether it was spelled out on first use.

Usage and example To extract acronyms, there are two classes used: the *Acronym* class, which is the object that represents the acronym and it contains the acronym itself, its spelling, and whether it was spelled out on first use, and the *AcronymDetector* class, which is a class that contains two static methods, the *detectAcronyms*(*Strings*) and *checkAcronymsOnSight*(*Strings*).

The *detectAcronyms*(*Strings*) method takes as input a string representing a text, it removes the non-ASCII characters and the punctuation, and then extracts a list of strings, where each string is an acronym (the extraction is performed according to the method mentioned in Section 2.2.

```
LinkedList<String> listOfUncheckedAcronyms =
AcronymDetector.detectAcronyms(String input);
```

The *checkAcronymsOnSight*(*Strings*) method takes as input a text, removes the non-ASCII characters and the punctuation, and then extracts a list of *Acronym* objects, each containing more information about each acronym.

```
LinkedList<Acronym> listOfUncheckedAcronyms =
AcronymDetector.checkAcronymsOnSight(String input);
```

3.2.2 Extract frequently used (key)words

Data input This module takes as input a text as a string.

Data output The module generates a list of stems, together with the frequency with which they are used in the text and the words that the stem was subtracted from.

Usage and example This module uses two classes: *Keyword* and the utility *KeywordExtractor*. The *Keyword* class is the object that represents a frequently used (key)word and it contains the stem that is / was processed by the keyword extraction algorithm, the frequency, which represents the number of times that stem was encountered in the text, and a set of terms from which the stem was derived. The *KeywordExtractor* takes a text as input and generates a list of keyword objects.

```
ArrayList<Keyword> listOfKeywords =
KeywordExtractor.guessFromString(String input)
```

3.2.3 Extract keyphrases

This algorithm uses the MAUI library to perform the necessary operations to extract keyphrases.

Data input This module takes as input a text as a string.

Data output The module generates a list of keyphrases, together with the frequency with which they are used in the text.

Usage and example This module uses two classes: *Keyphrase* and the utility *KeyphraseExtractor*. The *Keyphrase* class stores the phrase as a string, the words in the phrase as a list of strings, and the stems of the words that were used as a list of strings. The *KeyphraseExtractor* takes a text as input and generates a list of keyphrase objects.

```
ArrayList<Keyphrase> listOfKeyphrases =
KeyphraseExtractor.getKeyphrases(String input)
```

3.2.4 Translate content

This functionality is based on MyMemory[24], which is a service that provides an API for translating terms.

Data input This module takes text as a string as input.

Data output The module generates a string representing the translation of the input string in the desired language. The implemented application provides translation services between English and Swedish only.

Usage and example The module uses one class, namely *Translator*, which takes as input three strings, the first string representing the language from which the text should be translated, the second string representing the language to which the text should be translated, and the third one represents the text that should be translated.

```
String translatedText =
Translator.translate("en", "sv", String input)
```

3.2.5 Read documents

This functionality is based on the Apache Tika library, which provides information (meta-data) regarding the type of the uploaded file.

Data input This module takes a file as input.

Data output The module generates an *InputDocument* object, which is an object representation of the file that contains information regarding the document's type, its title, its content, and its number of pages.

Usage and example The module uses one class, namely *UtilClass*, which takes as input a file and returns an *InputDocument* object instance of the input.

```
InputDocument doc =
UtilClass.getInstance().getInputDocument(
new File(String filePath));
```

3.2.6 Parse web pages

This functionality is based on the Palladian library, which provides suitable tools for retrieving and parsing the a web page's HTML content.

Data input This module takes as input a string that represents the URL of a desired web page.

Data output The module generates an *WebDocument* object, which is an object representation of the content received from parsing the web page located at the URL that contains information regarding the document's title, content, and its URL.

```
WebDocument docWeb = new WebDocument(String url);
```

3.2.7 Course XML parser

This module depends on the KTH API "för kurs- och programinformation" [11]. It extracts course codes for a given year and course round. It also allows us to read the XML page of each course and then extract meta data.

Data input This module takes as input a DOM (Document Object Model) that corresponds to the XML page of the course list for a given round.

Data output The module generates a Map with instances of the class *Course*. Each *Course* instance contains meta deta extracted from its' XML page.

Usage and example This course XML parser extracts a list of course codes using the API using a URL of the form http://www.kth.se/api/kopps/v1/courseRounds/{year}:{round}.

It extracts course meta data using the API using a URL of the form http://www.kth.se/api/kopps/v1/course/{courseCode}.

```
javax.xml.parsers.DocumentBuilder dBuilder =
    DocumentBuilderFactory.newInstance().
    newDocumentBuilder();
org.w3c.dom.Document doc = dBuilder.parse(
"http://www.kth.se/api/kopps/v1/courseRounds/2014:2");
HashMap<String, Course> courses = CourseXMLPageParser.
    retrieveCourseCodes(doc);
CourseXMLPageParser.
    updateCourseContent(courses.get("ID2203"));
```

The function *updateCourseContent* builds the URL for the course XML page based on the course code, and then from each DOM node extracts meta data.

3.2.8 Course web page parser

This module parses a course's home page to extract keywords, keyphrases, and acronyms.

Data input An instance of the *Course* class.

Data output Updagtes *Course* with updated information regarding keywords, keyphrases, and acronyms.

Usage and example First it creates the URL of the course page, using the following patern "http://www.kth.se/student/kurser/kurs/courseCode?l=en. The courseCode is extracted from *Course*. It uses the module "Course web page parser"3.2.6. It also detects and stores the language of the web page.

```
CourseHomePageParser.updateCourseInfo(course, stopwords);
```

3.2.9 Suggest courses

Data input This module takes as input four lists of strings: one list contains course codes for courses taken by the user, a list of acronyms that was previously extracted from either a web page or from several documents, a list of keywords extracted from the same documents, and / or a list of keyphrases extracted from the documents or from the specified list of interests by the user.

Data output The module generates a list of suggested courses, together with the acronyms, keyphrases, and keywords that were used in assigning the suggestion weight.

Usage and example This module depends on the course data set that was extracted from KTH's course web and it has been implemented as a PG/SQL function.

```
select * from suggestcoursesfinal
(text[] course_ids, text[] acronym, text[] keyword,
text[] keyphrases)
```

3.2.10 Suggest tutors

Data input This module takes as input a four lists of string: one list contains course codes for courses taken by the user, a list of acronyms that was previously extracted from either a web page or from several documents, a list of keywords extracted from the same documents, and / or a list of keyphrases extracted from the documents or from the specified list of interests by the user.

Data output The module generates a list of suggested tutors, together with the acronyms, keyphrases, and keywords that were used in assigning the suggestion weight.

Usage and example This module depends on the course data set that was extracted from KTH's course web and it has been implemented as a PG/SQL function.

```
select * from suggesttutorfinal
(text[] course_ids, text[] acronym, text[] keyword,
text[] keyphrases)
```

3.3 Web Application

The aforementioned modules have been implemented as part of the *DocAid* Engine. To illustrate the efficiency of this set of models, we implemented a web application built on top of the *DocAid* Engine. The web application is functional and it can be accessed at the *DocAid* homepage: http://kthtest-docaid.rhcloud.com/docaid/index.jsp [8]. Its architecture is shown in Figure 1. The web application is hosted on Openshift [26], a cloud computing platform as a service, that has full support for the components required by *DocAid*. First, the *DocAid* Engine uses a PostgreSQL [30] database as a back-end to store course meta-data and dictionaries of stop words.

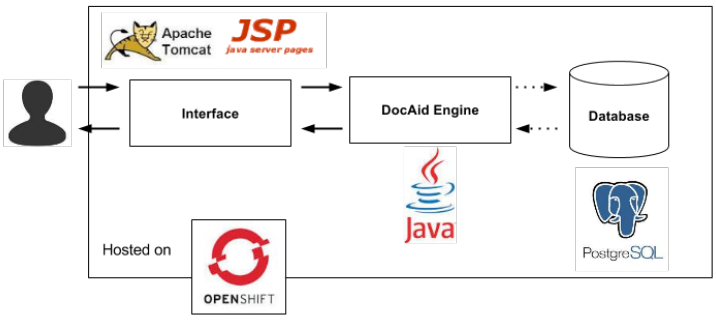


Figure 2: *DocAid* web application architecture

The *DocAid* functionality is available in the web application via the following pages:

Acronym extraction The input form of this page is identical to the key phrase extraction. See Figures 3 and 4 for examples of results. The three columns of the table are the acronym, the acronym spelled out, and a boolean field that indicates if the acronym was spelled out on first use in the source text.

List of extracted acronyms — Number of Acronyms: 17

Acronym	Spell out	Spelled out on first use
ITS	intelligent transport systems	true
DSRC	dedicated short range communications	true
EFC	electronic fee collection	true
RHS	road hazard signaling	true
ATS	abstract test suite	true
ICT	not spelled out	false
848KB	not spelled out	false
CALM	not spelled out	false
GSM	not spelled out	false
ETSI	not spelled out	false
CAM	not spelled out	false
ENAP	not spelled out	false
STF	not spelled out	false
DCC	not spelled out	false
DENM	decentralized environmental notification messages	false
PICS	protocol implementation conformance statement	false
ERM	not spelled out	false

Figure 3: Acronyms extracted from a web page URL (in this case, the URL is the first item of Appendix C). The first column is the acronym. The second column shows whether the acronym is spelled out. The final column is a boolean value that indicates if the acronym was spelled out on first use.

List of extracted acronyms — Number of Acronyms: 89

Acronym	Spell out	Spelled out on first use
IUIPC	internet users information privacy concerns	true
DMA	direct marketing association	true
DTI	department trade industry	true
ESM	experience sampling method	true
GPS	global positioning system	true
HCI	human computer interaction	true
HGI	home gateway initiative	true
LBS	location based services	true
MSA	measure sampling adequacy	true
OCR	optical character recognition	true
PLS	partial least square	true
SEM	structural equation modeling	true
TAM	technology acceptance model	true
USD	united states dollar	true
WWW	world wide web	true
KTH	not spelled out	false
ICT	information communication technology	false
SICS	swedish institute computer sciences	false
KMO	kaiser meyer olkin	false
ACM	association computing machinery	false
CFIP	not spelled out	false
DASH7	not spelled out	false

Figure 4: Acronyms from a document (in this case the 3rd item of Appendix C). The first column is the acronym. The second column is the acronym spelled out (if possible). The final column is a boolean value that indicates if the acronym was spelled out on first use.

Keyword extraction The user has the option to provide the URL of an HTML web page or to upload a document to the server. If the format of the file is not supported, an appropriate message is returned to the user. Additionally, there are several input parameters that can influence the keyword extraction output, such as enable or disable filtering of stop words, words that contain numbers (e.g., LAB1, or numeric text), minimum number of occurrences for a stem to be considered a keyword and the minimum keyword length in characters. Finally, when the input text source does not contain sufficient data to extract keywords a corresponding message is returned.

Each row of the resulting table contains the stem, the frequency of the stem, how many times this stem appeared in the source text, and the words that are mapped to the particular stem. The table, by default, is sorted on frequency, but it offers the option to sort the results on any column. Figures5 and 6 show the input forms and Figures 7, 8, and 9 show the results extracted from the document and / or the URL (as appropriate).

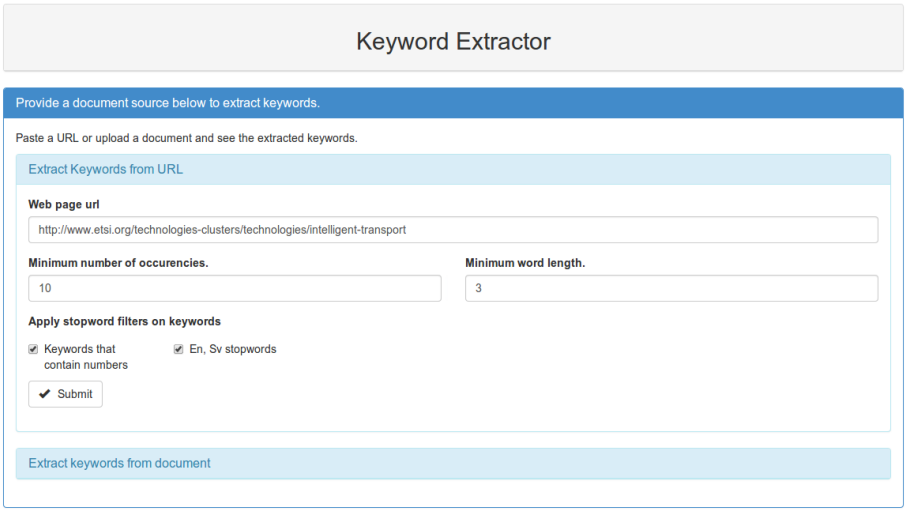


Figure 5: Keywords form. Takes input the URL of an HTML page, the minimum number of occurrences of a keyword, the minimum length of a word to be considered as a keyword. It contains a checkbox to enable stop word filtering and filtering of words that contain numbers.

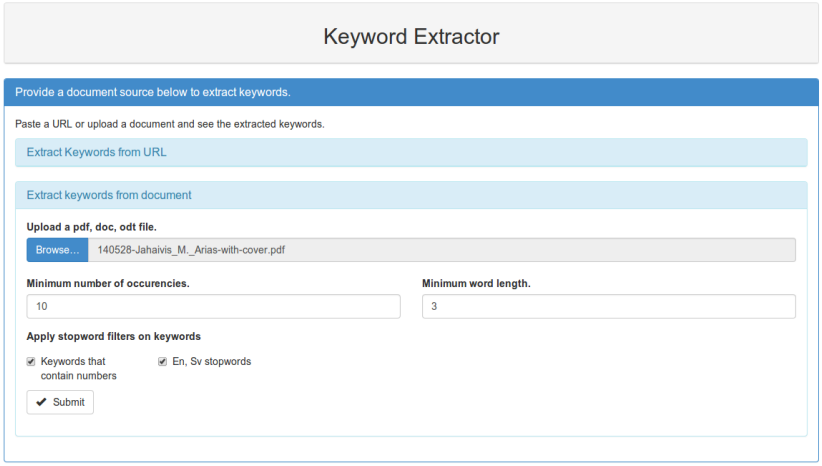
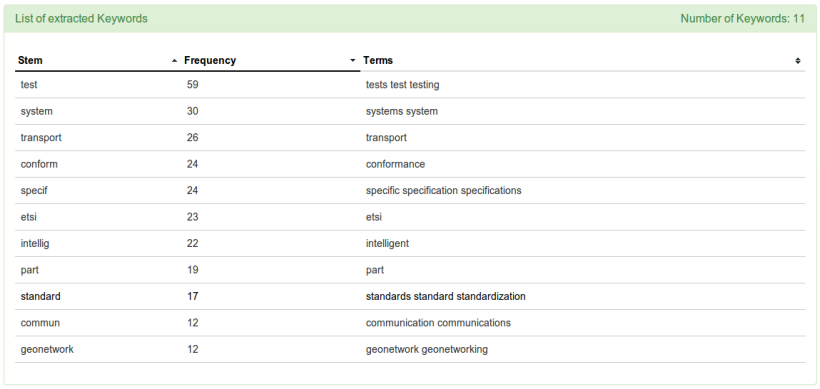


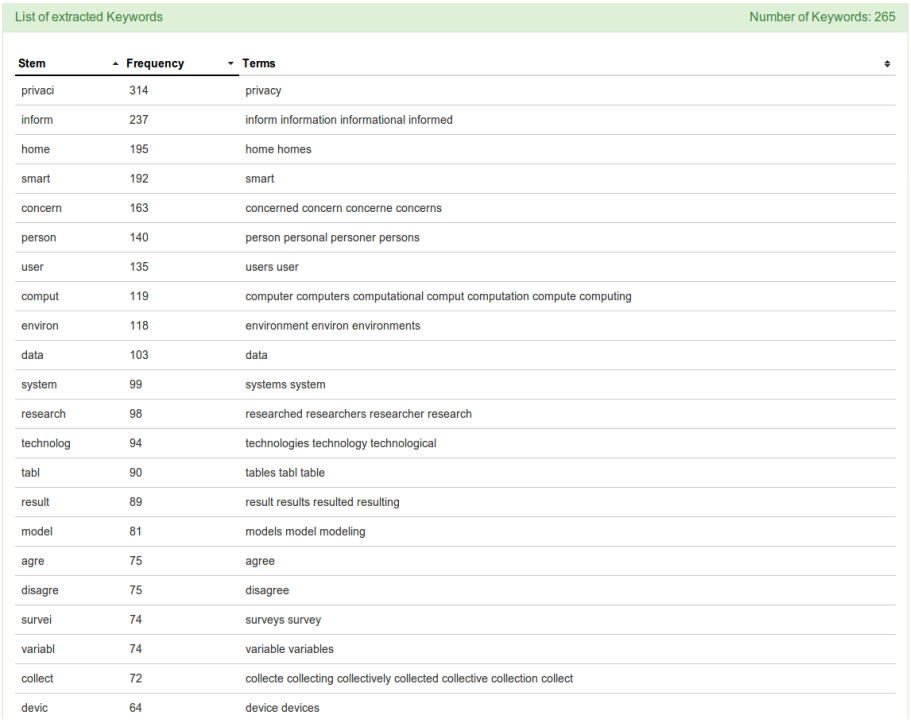
Figure 6: Keywords from document. Takes input the document to upload, the minimum number of occurrences of a keyword, the minimum length of a word to be considered as a keyword. It contains a checkbox to enable stop word filtering and filtering of words that contain numbers.



List of extracted Keywords — Number of Keywords: 11

Stem	Frequency	Terms
test	59	tests test testing
system	30	systems system
transport	26	transport
conform	24	conformance
specif	24	specific specification specifications
etsi	23	etsi
intellig	22	intelligent
part	19	part
standard	17	standards standard standardization
commun	12	communication communications
geonetwork	12	geonetwork geonetworking

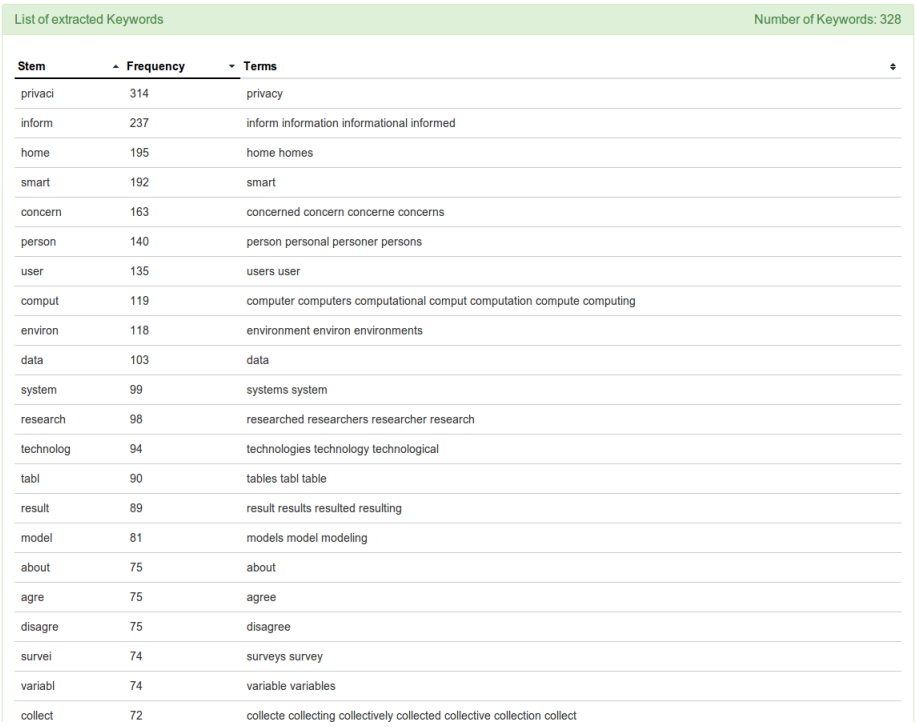
Figure 7: Keywords extracted from the web page URL shown in Figure 5. The first column is the keyword stem, the second column is the frequency of occurrence and the final column holds the words that reference the stem.



List of extracted Keywords — Number of Keywords: 265

Stem	Frequency	Terms
privaci	314	privacy
inform	237	inform information informational informed
home	195	home homes
smart	192	smart
concern	163	concerned concern concerne concerns
person	140	person personal personer persons
user	135	users user
comput	119	computer computers computational comput computation compute computing
environ	118	environment environ environments
data	103	data
system	99	systems system
research	98	researched researchers researcher research
technolog	94	technologies technology technological
tabl	90	tables tabl table
result	89	result results resulted resulting
model	81	models model modeling
agre	75	agree
disagre	75	disagree
survei	74	surveys survey
variabl	74	variable variables
collect	72	collecte collecting collectively collected collective collection collect
devic	64	device devices

Figure 8: Keywords extracted from an uploaded document (in this case the 3rd item of Appendix C) with filtered stop words. The first column is the keyword stem, the second column is the frequency of occurrence, and the final column is the words that reference the stem.



List of extracted Keywords — Number of Keywords: 328

Stem	Frequency	Terms
privaci	314	privacy
inform	237	inform information informational informed
home	195	home homes
smart	192	smart
concern	163	concerned concern concerne concerns
person	140	person personal personer persons
user	135	users user
comput	119	computer computers computational comput computation compute computing
environ	118	environment environ environments
data	103	data
system	99	systems system
research	98	researched researchers researcher research
technolog	94	technologies technology technological
tabl	90	tables tabl table
result	89	result results resulted resulting
model	81	models model modeling
about	75	about
agre	75	agree
disagre	75	disagree
survei	74	surveys survey
variabl	74	variable variables
collect	72	collecte collecting collectively collected collective collection collect

Figure 9: *DocAid* Keywords from an uploaded document (in this case the 3rd item of Appendix C), unfiltered. The first column is the keyword stem, the second column is the frequency of occurrence, and the final column is the words that reference the stem.

Keyphrase extraction This page tales as input a URL or a text source. No additional parameters are available to the user. The resulting table contains two columns: the keyphrase and a factor. Figure 10 shows the input form for a URL text source to extract key phrases, while Figures 11 and 12 show the extracted results for this input.

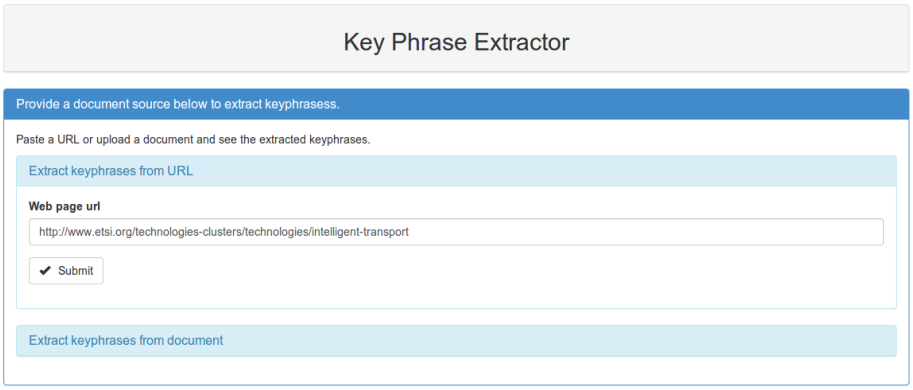
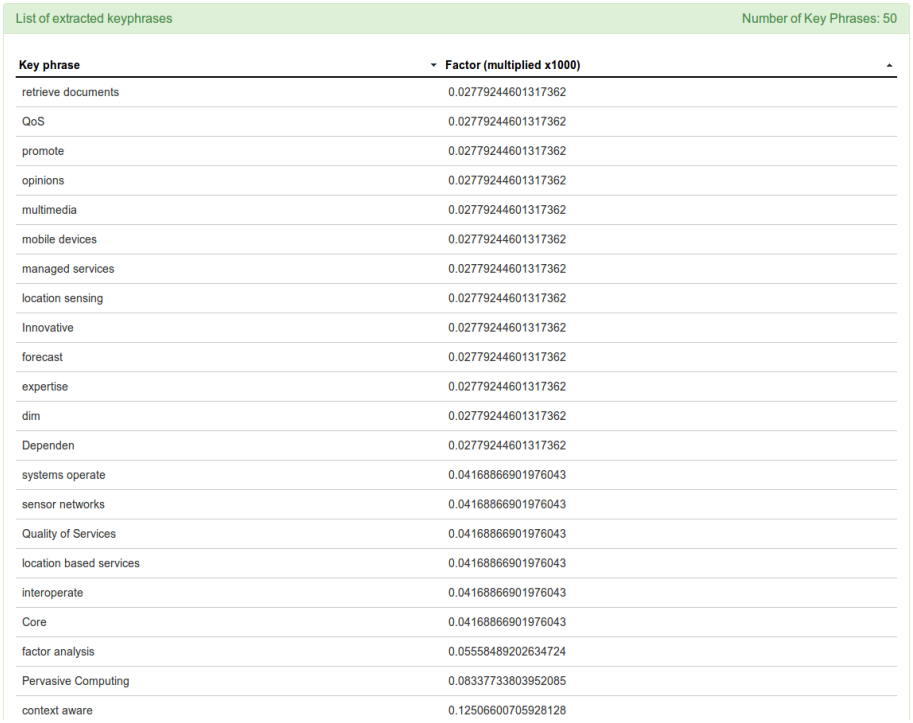


Figure 10: Key phrase form. Takes input the URL of an HTML page.

List of extracted keyphrases — Number of Key Phrases: 50

Key phrase	Factor (multiplied x1000)
vehicle and the roadside	0.4351610095735422
tolling	0.4351610095735422
Systems dedicated to Intelligent Transport	0.4351610095735422
Systems and Road Transport	0.4351610095735422
standards development	0.4351610095735422
speed railways	0.4351610095735422
roadside	0.4351610095735422
Road Transport	0.4351610095735422
road safety	0.4351610095735422
regulatory	0.4351610095735422
passengers	0.4351610095735422
including navigation	0.4351610095735422
high speed railways	0.4351610095735422
high speed	0.4351610095735422
Harmonized	0.4351610095735422
DSRC	0.4351610095735422
CALM	0.4351610095735422
automotive	0.4351610095735422
test specifications for Transmission	0.6527415143603134
test specifications for GeoNetworking	0.6527415143603134
test specification for DENM	0.6527415143603134
specifications for Transmission	0.6527415143603134

Figure 11: Key phrases extracted from the web page URL shown in Figure 10. The first column is the key phrase. The second column is the weight of the keyphrase in ascending order. The smaller the factor the more relevant the phrase.



List of extracted keyphrases — Number of Key Phrases: 50

Key phrase	Factor (multiplied x1000)
retrieve documents	0.0277924460131736 2
QoS	0.0277924460131736 2
promote	0.0277924460131736 2
opinions	0.0277924460131736 2
multimedia	0.0277924460131736 2
mobile devices	0.0277924460131736 2
managed services	0.0277924460131736 2
location sensing	0.0277924460131736 2
Innovative	0.0277924460131736 2
forecast	0.0277924460131736 2
expertise	0.0277924460131736 2
dim	0.0277924460131736 2
Dependen	0.0277924460131736 2
systems operate	0.0416886690197604 3
sensor networks	0.0416886690197604 3
Quality of Services	0.0416886690197604 3
location based services	0.0416886690197604 3
interoperate	0.0416886690197604 3
Core	0.0416886690197604 3
factor analysis	0.0555848920263472 4
Pervasive Computing	0.0833773380395208 5
context aware	0.1250660070592812 8

Figure 12: Key phrases from an uploaded document (in this case the 3rd item of Appendix C). The first column is the key phrase. The second column is the weight of the keyphrase in ascending order. The smaller the factor the more relevant the phrase.

Document analysis This page combines the functionality of the afore mentioned pages. The input parameters are the same as for keywords page and the results are a table from each module.

Course advisor This page takes input from various sources. The user can choose to use some or all of the fields. First, the user can specify a set of keywords and key phrases delimited by commas which correspond to his / her interest list. Second, the can upload his / her registration history. Our parse has proven to work better with the transcript of records printed as a PDF file from the KTH Social web site. Third, there is a document upload field which allows for multiple files to be selected. From these files meta data are extracted and used in the recommendation algorithm. Finally, all the keyword extraction options are available for the user to customize along with two extra options which limit the course and tutor suggestions.

The course suggestions table contain the course codes, the course title, the recommendation weight (i.e., the sum of the other columns), acronym, keyword,

and keyphrase weight. By default the table is sorted on the recommendation weight, but the user can select the other weights for sorting.

Since the *DocAid* database is populated with detailed meta data for each course, additional information can be presented to the user (i.e., the acronym, keyword and key phrase matches from the document sources to the course meta data).

Figure 13 provides the input form, and Figures 14 and 15 show the results of the advisor module, and Figure 16 shows the detailed list of acronyms, keywords and keyphrases used to determine the weight for the course "AG2417 Web and Mobile GIS".

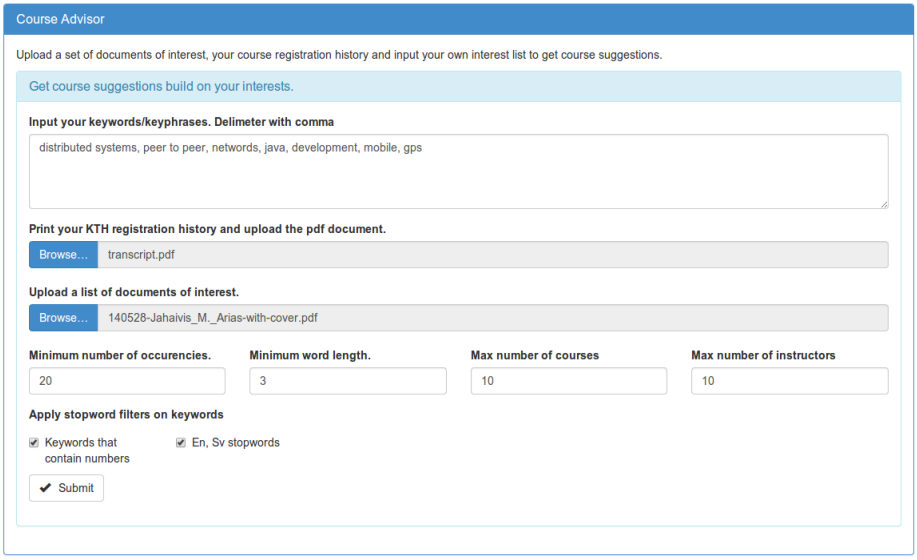


Figure 13: Course Advisor form. Keywords and key phrases provided in the user preference text area. A PDF document containing the transcript of records of a KTH student has been uploaded (in this case the 2nd item of Appendix C). Also, the user enters a PDF document with content of interest to the student (in this case the 3rd item of Appendix C). The keyword extraction parameters are shown and can be modified by the user. Finally the number of recommendations per course and tutor can be limited.

List of course recommendations — Number of recommendations: 10

Course code	Title	Reccommendation weight	Acronyms weight	Keywords weight	Key phrases weight
LL219U	Technology for Teachers, Grade 7-9, 45 Credits (1-45). Part of Lärarlyftet II	0.573	0.218	0.0	0.356
AG2429	Geovisualization	0.507	0.218	0.0	0.289
MF1049	Product Realization for Teachers in Technology	0.504	0.174	0.0	0.33
AK2050	Theory and Methodology of Science with Applications (Medical Ethics)	0.5	0.174	0.0	0.327
AG2417	Web and Mobile GIS	0.473	0.261	0.0	0.212
MJ2496	Innovation and Entrepreneurship in Sustainable Energy Technology	0.47	0.131	0.0	0.34
SK2521	Fluorescence Spectroscopy for Biomolecular Studies	0.468	0.174	0.0	0.294
MF1061	Introduction to Design and Product Realisation	0.464	0.131	0.0	0.334
ML2111	Distribution System	0.458	0.131	0.0	0.327
AG2414	Spatial Analysis	0.456	0.261	0.0	0.196

Figure 14: Course suggestions for input used in Figure 13.

List of tutor recommendations — Number of recommendations: 10

Instructor name	Reccommendation weight	Acronyms weight	Keywords weight	Key phrases weight
kurser.tamos@abe.kth.se	1.464	1.087	0.0	0.377
registrar@sses.se	1.311	1.131	0.0	0.18
Göran Baurne	1.153	0.653	0.0	0.501
Mats Bengtsson	1.049	0.696	0.0	0.354
Per Alvfors	0.835	0.218	0.0	0.618
kurser.juridik@abe.kth.se	0.831	0.653	0.0	0.179
Nils Brandt	0.813	0.348	0.0	0.465
Tomas Karlsson	0.803	0.479	0.0	0.325
Catharina Erlich	0.799	0.218	0.0	0.582
Viktoria Fodor	0.749	0.392	0.0	0.358

Figure 15: Tutor suggestions for the input used in Figure 13.

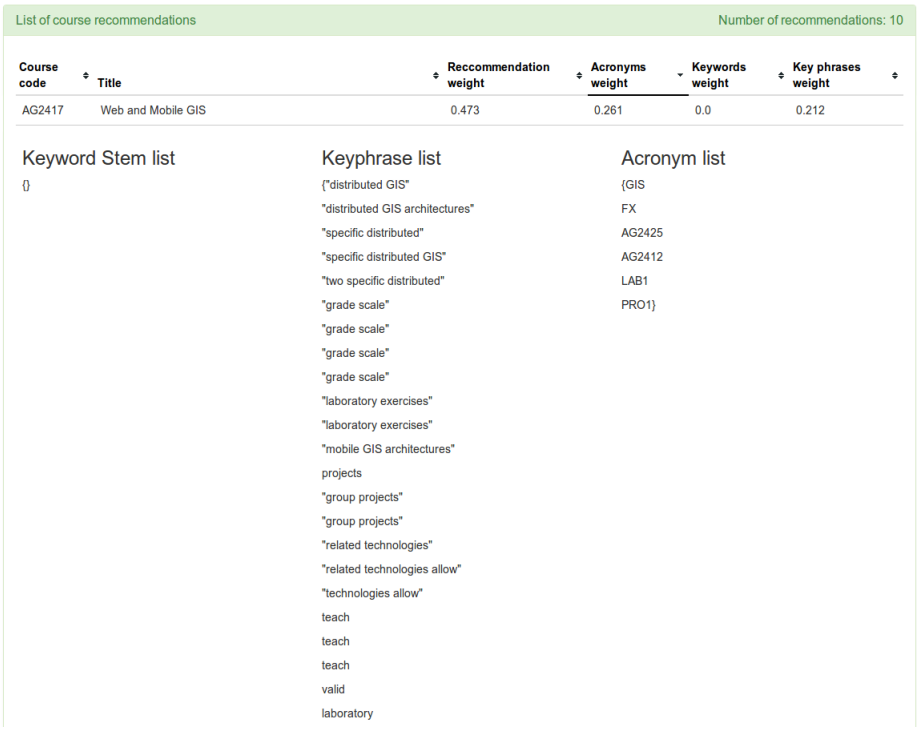


Figure 16: The keyphrases and acronyms that led to suggesting course "AG2417 Web and Mobile GIS".

Performance issues The web application appears to have high latency. This occurs because the available resources are shared with other users of Openshift, and only 512 megabytes are dedicated to the application. Also, if the application is not accessed for a specific amount of time, on the next access, it is redeployed.

4 Results and discussion

In this section, we present the results of the proposed methods and we discuss what can be deduced from them.

4.1 Acronym extraction

The algorithm recognized 940 acronym-definition pairs of which 196 were incorrectly identified from the set of 10 documents. The results obtained were 79.1% precision (with a standard deviation of 13.2%) and 89.2% recall (with a standard deviation of 5.6%), yielding an F_1 score of 83.9% (with a standard deviation of 7.9%). The results are shown in Table 1.

Table 1: Measurements performed to evaluate the acronym extraction algorithm. The first column represents the document id. The next four columns represent the number of acronyms inside a document, the number of correctly identified acronyms, the total number of acronyms, and the number of incorrectly classified words as acronyms. The final three columns represent the precision, recall and $F1$ measure of the algorithm.

Doc_id	Declared	Correct	Total	Incorrect	Prec. %	Rec. %	F1 %
1	32	26	37	11	70.3	81.3	75.4
2	89	83	136	53	61.0	93.3	73.8
3	26	22	32	10	68.8	84.6	75.9
4	33	31	57	26	54.4	93.9	68.9
5	97	93	141	48	66.0	95.9	78.2
6	59	51	61	10	83.6	86.4	85.0
7	53	50	56	6	89.3	94.3	91.7
8	97	86	97	11	88.7	88.7	88.7
9	67	66	78	12	84.6	98.5	91.0
10	281	236	245	9	96.3	84.0	89.7
Agg.	834	744	940	196	79.1	89.2	83.9

When analyzing the misclassified instances, we noticed that our algorithm failed to distinguish between words written in upper case by authors not as acronyms, but as a mean to emphasize their importance. Furthermore, all the acronyms that contained (any) small letters were missed because of the heuristic we employed. Another confusion was caused by identifying codes assigned by authors to certain elements as objects (e.g., a paper proposed the code COD for a certain species of fish) or by identifying institutions in the bibliography / references section as acronyms. We noticed that a large percentage of the falsely classified words as acronyms were due to the employed heuristic, which allows for two-letter upper case words to be considered as acronyms.

The algorithm identified the correct spelling out of the acronyms in 71.2%

of the cases, and the correct spelling out of the acronyms on the first use was identified in 68.8% of the cases. These results are shown in Tables 2 and 3.

Table 2: The output of tests performed for evaluating the acronym spelling detection module. The first column represents the document id. The next two columns represent the number of acronyms inside a document and the number of correctly spelled acronyms identified by the algorithm.

Doc_id	Declared	Identified	Prec.%
1	32	24	75.0
2	89	62	69.7
3	26	16	61.5
4	33	19	57.6
5	97	78	80.4
6	59	43	72.9
7	53	36	67.9
8	97	74	76.3
9	67	58	86.6
10	281	184	65.5
Agg.	834	594	71.2

Table 3: Measurements performed for evaluating the ability to correctly identify an instance of an acronym being spelled out on first use. The first column represents the document id. The next two columns represent the number of acronyms inside a document and the number of acronyms correctly spelled on first use based upon the acronyms identified by the algorithm.

Doc_id	Declared	Identified	Prec.%
1	32	19	59.4
2	89	57	64.0
3	26	15	57.7
4	33	17	51.5
5	97	76	78.4
6	59	43	72.9
7	53	35	66.0
8	97	70	72.2
9	67	58	86.6
10	281	184	65.5
Agg.	834	574	68.8

Those acronyms whose spelling out could not be identified even if it was present had a structure that did not match our heuristics (e.g., *LIDAR* was

spelled as “light detection and ranging”, where “and” was treated as a stop word and ignored). In the 5.6% of the cases were the spelling was correct but it could not be identified on the first check, the error was caused either by an extra comment for the first use (e.g., putting an extra note in parentheses when the acronym is spelled out for the first time) or one of the words was split into two syllables due to automatic hyphenation.

4.2 Keywords and keyphrases extraction

We extracted 100 keywords and phrases out of 20 documents, out of which the naive approach exactly identified 20% of the cases and, if merged with the partial identification, it managed to identify 52% of the cases. However, the value of the standard deviation was high (22.4% for the exact identification and 30.6% for the merged approach) due to the fact that this method is solely based on the frequency of the words used (see Table 4 for the results). We also noticed that the number of suggestions depends on the frequency we set as a minimum (see Table 5 for the number of suggestions per document, given a minimum frequency of 20).

Table 4: The results of measurements performed to evaluate the naive approach of keyword extraction. The first column represents the document id. The next columns represent the number of declared keywords in the abstract, the number of exactly matched keywords, the number of approximately matched keywords, the merged results, and the respective precisions.

Doc_id	Keyword	Exact	Approx.	Merged	Prec. Ex	Prec. Merg
1	5	0	2	2	0.0	40.0
2	5	1	3	4	20.0	80.0
3	4	2	2	4	50.0	100.0
4	4	0	0	0	0.0	0.0
5	5	2	3	5	40.0	100.0
6	5	0	1	1	0.0	20.0
7	5	1	0	1	20.0	20.0
8	9	3	2	5	33.3	55.6
9	6	0	4	4	0.0	66.7
10	5	0	3	3	0.0	60.0
11	5	1	0	1	20.0	20.0
12	9	0	2	2	0.0	22.2
13	7	1	5	6	14.3	85.7
14	6	1	0	1	16.7	16.7
15	4	2	1	3	50.0	75.0
16	4	2	0	2	50.0	50.0
17	3	2	0	2	66.7	66.7
18	2	0	1	1	0.0	50.0
19	4	2	0	2	50.0	50.0
20	3	0	3	3	0.0	100.0
Agg.	100	20	32	52	20.0	52.0

Table 5: These measurements were performed to check the number of suggestions considering only those keywords that occur at least 20 times. The last column represents the number of suggestions.

Doc_id	Keyword	Merged	Suggestions
1	5	2	62
2	5	4	35
3	4	4	92
4	4	0	26
5	5	5	31
6	5	1	16
7	5	1	69
8	9	5	38
9	6	4	191
10	5	3	37
11	5	1	71
12	9	2	55
13	7	6	132
14	6	1	14
15	4	0	89
16	4	3	21
17	3	2	79
18	2	2	71
19	4	1	73
20	3	2	86
Agg.	100	49	1288

The keyphrases extraction performed with MAUI clearly outperformed the naive approach, exactly identifying 65.0% of the keywords and merging the exactly identified keywords with the partially identified ones, the classifier reached an accuracy of 81.0%. However, the value of the standard deviation is still high, 19.7% for the exact approach and 17.5% for the merged approach (see Table 6 for the results). When verifying if most of the results are in a certain interval (e.g., given a maximum number of 50 keyphrases recommended per document, how many of the suggested keywords are in the first 10 recommendations), we noticed that the matches tend to be in either the first suggested keyphrases or the last suggested ones (when considering the weight), with a gap in the middle (see Table 7 for the relevant measurements).

Table 6: The results of measurements performed to evaluate the keyphrase extraction. The first column represents the document id. The next columns represent the number of declared keywords in the abstract, the number of exactly matched keyphrases, the number of approximately matched keyphrases, the merged results, and the respective precisions.

Doc_id	Keyword	Exact	Approx.	Merged	Prec. Ex	Prec. Merg
1	5	3	2	5	60.0	100.0
2	5	4	0	4	80.0	80.0
3	4	2	2	4	50.0	100.0
4	4	3	0	3	75.0	75.0
5	5	4	1	5	80.0	100.0
6	5	3	1	4	60.0	80.0
7	5	2	1	3	40.0	60.0
8	9	7	0	7	77.8	77.8
9	6	3	2	5	50.0	83.3
10	5	3	0	3	60.0	60.0
11	5	3	1	4	60.0	80.0
12	9	7	1	8	77.8	88.9
13	7	4	2	6	57.1	85.7
14	6	1	1	2	16.7	33.3
15	4	3	0	3	75.0	75.0
16	4	3	1	4	75.0	100.0
17	3	3	0	3	100.0	100.0
18	2	1	1	2	50.0	100.0
19	4	3	0	3	75.0	75.0
20	3	3	0	3	100.0	100.0
Agg.	100	65	16	81	65.0	81.0

Table 7: This table contains information regarding the distribution of the suggested keyphrases, considering a maximum of 50 keyphrase recommendations per document. Columns 2-5 represent the number of recommend keyphrases that were found in their adjacent recommendation interval. It is noticeable that for these tests, the distribution appears to have the form of a reversed Gauss bell.

Doc_id	Keyword	Merged	1-10	11-20	21-30	31-40	41-50
1	5	5	1	1	2	0	1
2	5	4	0	1	1	2	0
3	4	4	2	1	0	0	1
4	4	3	0	3	0	0	0
5	5	5	2	1	1	1	0
6	5	4	2	0	0	0	2
7	5	3	1	0	0	1	1
8	9	7	2	2	0	2	1
9	6	5	2	0	1	2	0
10	5	3	2	0	0	0	1
11	5	4	1	0	1	0	2
12	9	8	0	2	0	3	3
13	7	6	0	2	1	2	1
14	6	2	0	1	0	0	2
15	4	3	0	3	0	0	0
16	4	4	2	0	0	1	1
17	3	3	2	0	0	0	1
18	2	2	1	0	1	0	0
19	4	3	2	0	0	1	0
20	3	3	0	1	0	2	0
Agg.	100	81	22	18	8	17	17

The only observation we could made was that in some cases the authors specify keywords or keyphrases that are not written in the text in the same way as in the list of keywords in the abstract (e.g., if the author writes “sensor fusion” as a keyword but in the text he / she uses “fused data from sensors”).

These tests should be performed on a larger data set to obtain more statistically significant results.

4.3 Course and tutor suggestion

We have noticed that if we base the recommendation algorithm on naive association rules, some courses and / or tutors are present in most of the performed suggestions. This is mostly due to the level of detail of each course description, which is automatically associated with the tutor as well, and varies across all courses (some courses only have the time schedule in their content,

while others provide information regarding the studied literature and a list of commonly used acronyms and abbreviations of the course).

5 Conclusions and future work

This section concludes the report and presents the authors' views regarding what could be implemented to improve the presented system.

5.1 Conclusions

In this report, we have shown that it is possible to collect meta-data from on-line sources and presented a set of tools that can be used for this task. Furthermore, we have explored the possibility of recommending courses based on a user's predefined set of interest, such as manual input of keywords and / or keyphrases, uploading documents of interest, and updating his / her registration history. We have designed a web application that supports the proposed subgoals and we made it publicly available at the *DocAid* homepage [8].

5.2 Future work

In this report, we implemented the course and tutor recommendation functionality based solely on static content, where a course is considered to be relevant to a document or interest list if it has similar keywords, keyphrases, or acronyms. To make the recommending system more robust, it would be useful to consider each course's prerequisites and future learning perspective (e.g., the "Modern Database Systems" course requires the "Introduction to Databases" course). Finally. a great addition would be to take into account the registration trend of students (if 90% of the students that took courses A and B continued with C, the recommender system would tend to suggest C). However, users might not be willing to share their registration history or other education related information because of privacy concerns. It would be interesting to take into account personalized content that adapts to the user's preference. Such a system would probably be similar to the one proposed by Billsus and Pazzani [4]), which suggests a daily news program based on the preference data derived from what the user reads in a predefined time interval.

References

[1] "Apache Lucene" http://lucene.apache.org/core/ Last accessed on 4 Jun. 2014

[2] "Apache Tika - a content analysis toolkit" http://tika.apache.org/ Last accessed on 4 Jun. 2014

[3] "Bilda " http://intra.kth.se/utbildning/bilda/vad-ar-bilda-1.79559 Last accessed on 4 Jun. 2014

[4] Billsus, D., and Pazzani, M. J. "A hybrid user model for news story classification." In Courses and Lectures - International Center for Mechanical Sciences 99: 108, 1999, doi=10.1.1.44.8942.

[5] Burke, R. "Knowledge-based recommender systems." Encyclopedia of library and information systems 69, no. Supplement 32: 175-186, 2000, doi=10.1.1.21.6029.

[6] Chen, Y., Xue G.R., and Yu Y. "Advertising keyword suggestion based on concept hierarchy." In Proceedings of the 2008 international conference on web search and data mining: 251-260. ACM, 2008, doi=10.1145/1341531.1341564.

[7] "Daisy ICT " http://www.kth.se/en/student/studok/daisy-ict-1.48170 Last accessed on 4 Jun. 2014

[8] "DocAid " http://kthtest-docaid.rhcloud.com/docaid/index.jsp Last accessed on 4 Jun. 2014

[9] "LADOK (student registry)" http://www.kth.se/en/student/studentliv/studentratt/ladok-och-offentlighetsprincipen-1.374759 Last accessed on 4 Jun. 2014

[10] "KTH - Royal Institute of Technology" http://www.kth.se/en Last accessed on 4 Jun. 2014

[11] "KTH public APIs" http://www.kth.se/en/api/anvand-data-fran-kth-1.57059 Last accessed on 4 Jun. 2014

[12] "KTH Social " https://www.kth.se/social/ Last accessed on 4 Jun. 2014.

[13] "KNIME" http://www.knime.org/ Last accessed on 4 Jun. 2014

[14] Dannélls, D. "Automatic acronym recognition." In Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations, pp. 167-170. Association for Computational Linguistics, 2006,doi=10.1.1.103.4496

[15] Eibe, F., Paynter, G.W., Witten, I.H., Gutwin C., and Nevill-Manning C. G. "Domain-specific keyphrase extraction.", Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence: 668-673, 1999, doi=10.1.1.148.3598.

[16] Gleich, D., and Zhukov, L.. "SVD based term suggestion and ranking system." In Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on: 391-394. IEEE, 2004, doi=10.1.1.135.5254.

[17] Hammarström, H.. "Poor man's stemming: Unsupervised recognition of same-stem words." In Information Retrieval Technology, pp. 323-337. Springer Berlin Heidelberg, 2006, doi=10.1007/11880592_25

[18] "jlangdetect : Language detection API for Java" `https://code.google.com/p/jlangdetect/` Last accessed on 4 Jun. 2014

[19] "jsoup: Java HTML Parser" `http://jsoup.org/` Last accessed on 4 Jun. 2014

[20] Lin, D. "An information-theoretic definition of similarity." In ICML, vol. 98: 296-304, 1998, doi=10.1.1.55.1832.

[21] Lo, R. T. W., He B., and Ounis I. "Automatically building a stopword list for an information retrieval system." In Journal on Digital Information Management: Special Issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR), vol. 5: 17-24. 2005, doi=10.1.1.111.3041.

[22] "Maui - Multi-purpose automatic topic indexing " `https://code.google.com/p/maui-indexer/` Last accessed on 4 Jun. 2014

[23] Medelyan, O., Witten, I. H., and Milne D. "Topic indexing with Wikipedia." In Proceedings of the AAAI WikiAI workshop: 19-24. 2008, doi=10.1.1.147.7850.

[24] "MyMemory " `http://mymemory.translated.net/doc/spec.php` Last accessed on 4 Jun. 2014

[25] Ninoriya, S., Chawan, P. M., and Meshram, B. B. "CMS, LMS and LCMS For eLearning." International Journal of Computer Science Issues (IJCSI) 8, no. 2, 2011, doi=10.1.1.403.175.

[26] "OpenShift" `https://www.openshift.com/` Last Accessed on 5 Jun. 2014

[27] "Palladian Nodes for KNIME " Last accessed on 4 Jun. 2014 `http://tech.knime.org/community/palladian`

[28] Porter, M. F. "An algorithm for suffix stripping." Readings in Information Retrieval: 313-316, isbn = 1-55860-454-5

[29] Porter, M. F. "Snowball: A language for stemming algorithms.", 2001, http://snowball.tartarus.org/texts/introduction.html Last accessed on 4 Jun. 2014

[30] "PostgreSQL" http://www.postgresql.org/ Last accessed on 5 Jun. 2014

[31] Ratcliff, J.W., Metzener, D.E.: "Pattern-matching-the gestalt approach." Dr. Dobbs Journal, 13(7): 46, 1988 http://collaboration.cmc.ec.gc.ca/science/rpn/biblio/ddj/Website/articles/DDJ/1988/8807/8807c/8807c.htm Last accessed on 4 Jun. 2014

[32] van Rijsbergen, C.J. "Information Retrieval." Butterworths, London, 1979, doi=10.1.1.36.2325

[33] Sampson, G. "The 'Language Instinct' Debate", The Modern Language Journal 91, no 3: 486-487, 2007, doi=10.1111/j.1540-4781.2007.00593_14.x.

[34] Simonson, M. "Course management systems." Quarterly Review of Distance Education 8, no. 1: 7-9, 2011.

[35] Taghva, K., and Gilbreth, J.. "Recognizing acronyms and their definitions." International Journal on Document Analysis and Recognition 1, no. 4 : 191-198, 1999, doi=10.1.1.80.9429.

[36] Hall, M., et al. "KEA: Practical automatic keyphrase extraction." In Proceedings of the fourth ACM conference on Digital libraries: 254-255. ACM, 1999, doi=10.1.1.148.450.

[37] Yeates, S. "Automatic Extraction of Acronyms from Text." In New Zealand Computer Science Research Students' Conference: 117-124, 1999, doi=10.1.1.41.1738.

[38] Yeates, S., Bainbridge, D. , and Witten, I. H. "Using compression to identify acronyms in text." omputer Science, University of Waikato, 2000, doi=10.1.1.341.7587

[39] Yujian, L., and Bo L.. "A normalized Levenshtein distance metric." Pattern Analysis and Machine Intelligence, IEEE Transactions on 29, no. 6: 1091-1095, 2007, doi=10.1109/TPAMI.2007.1078;

Appendix A: Documents used for testing the acronym extraction algorithm

1. "Medicine Prices, Availability, Affordability and Price Components in Oman" http://www.haiweb.org/medicineprices/surveys/200710OM/sdocs/survey_report.pdf Last accessed June 7, 2014.

2. "Handbook on geographic information systems and digital mapping" http://unstats.un.org/unsd/publication/SeriesF/SeriesF_79E.pdf Last accessed June 7, 2014.

3. "MODIS Collection 5 Burned Area Product - MCD45" http://modis-fire.umd.edu/Documents/MODIS_Burned_Area_Collection5_User_Guide_2.0.pdf Last accessed June 7, 2014.

4. "Spawning and nursery grounds of selected fish species in UK waters" http://www.cefas.defra.gov.uk/publications/techrep/TechRep147.pdf Last accessed June 7, 2014.

5. "ACRP Report 88: Guidebook on Integrating GIS in Emergency Management at Airports" http://onlinepubs.trb.org/onlinepubs/acrp/acrp_rpt_088.pdf Last accessed June 7, 2014.

6. "List of acronyms and abbreviations - Guidelines for safe recreational water environment" http://www.who.int/water_sanitation_health/bathing/srwe1-pref.pdf Last accessed June 7, 2014.

7. "List of acronyms and abbreviations - Environmental Protection Agency" http://www.epa.gov/hudson/CIP_07_section5.pdf Last accessed June 7, 2014.

8. "Operational Assessment of 16 Campuses" https://www.ecu.edu/facility_serv/energy/Operational%20Assessment%20of%2016%20UNC%20Campuses-Final%20%20Report.pdf Last accessed June 7, 2014.

9. "Annex I: Glossary, Acronyms, Chemical Symbols and Prefixes. In IPCC Special Report on Renewable Energy Sources and Climate Change Mitigation" http://srren.ipcc-wg3.de/report/IPCC_SRREN_Annex_I.pdf Last accessed June 7, 2014.

10. "Annual review of DFID support to the anti-corruption commission phase 2 in Sierra Leone" http://s4rsa.wikispaces.com/file/view/DFID+Support+to+the+Anti-Corruption+Commission.pdf Last accessed June 7, 2014.

Appendix B: Documents used for testing the keyword and keyphrase extraction algorithm

1. Benz, U. C., Hofmann, P., Willhauck, G., Lingenfelder, I., and Heynen, M. Multi-resolution, object-oriented fuzzy analysis of remote sensing data for GIS-ready information. ISPRS Journal of photogrammetry and remote sensing, 58(3), 239-258, 2004 doi = 10.1.1.200.7311

2. Lu, D., Mausel, P., Brondizio, E., and Moran, E. Change detection techniques. International journal of remote sensing, 25(12), 2365-2401, 2004, doi=10.1080/0143116031000139863

3. Weng, Q. Land use change analysis in the Zhujiang Delta of China using satellite remote sensing, GIS and stochastic modelling. Journal of environmental management, 64(3), 273-284, 2002, doi = DOI:10.1006/jema.2001.0509

4. Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., and Tan, K. L. Private queries in location based services: anonymizers are not necessary. In Proceedings of the 2008 ACM SIGMOD international conference on Management of data (pp. 121-132), ACM, 2008, doi=10.1.1.226.121

5. Schmidt-Belz, B., Laamanen, H., Poslad, S., and Zipf, A. Location-based mobile tourist services-first user experiences. Information and communication technologies in tourism, 2003, 115-123, ISBN: 3-211-83910-0

6. Jiang, B., and Yao, X. Location-based services and GIS in perspective. Computers, Environment and Urban Systems, 30(6), 712-725, 2006, doi = 10.1016/j.compenvurbsys.2006.02.003

7. Goodchild, M. F. Citizens as sensors: the world of volunteered geography. GeoJournal, 69(4), 211-221, 2007, doi=10.1007/s10708-007-9111-y

8. Zheng, Y., Liu, L., Wang, L., and Xie, X. Learning transportation mode from raw gps data for geographic applications on the web. In Proceedings of the 17th international conference on World Wide Web (pp. 247-256), ACM, 2008, doi=10.1145/1367497.1367532

9. Stenneth, L., Wolfson, O., Yu, P. S., and Xu, B. Transportation mode detection using mobile phones and GIS information. In Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (pp. 54-63), ACM, 2011, doi=10.1145/2093973.2093982

10. Gonzalez, P. A., Weinstein, J. S., Barbeau, S. J., Labrador, M., Winters, P. L., Georggi, N. L., and Perez, R. Automating mode detection for travel behaviour analysis by using global positioning systemsenabled mobile phones and neural networks. Intelligent Transport Systems, IET, 4(1), 37-49, 2010, doi=10.1.1.391.7759

11. Hjaltason, G. R., and Samet, H. Distance browsing in spatial databases. ACM Transactions on Database Systems (TODS), 24(2), 265-318, 1999, doi=10.1145/320248.320255

12. Fischer, M. J., Lynch, N. A., and Paterson, M. S. Impossibility of distributed consensus with one faulty process. Journal of the ACM (JACM), 32(2), 374-382, 1985, doi=10.1.1.13.6760

13. Slocum, T. A., Blok, C., Jiang, B., Koussoulakou, A., Montello, D. R., Fuhrmann, S., and Hedley, N. R. Cognitive and usability issues in geovisualization. Cartography and Geographic Information Science, 28(1), 61-75, 2001, doi=10.1.1.23.6912

14. Elwood, S. Geographic Information Science: new geovisualization technologiesemerging questions and linkages with GIScience research. Progress in Human Geography, 2008, doi=10.1177/0309132508094076

15. Cuervo, E., Balasubramanian, A., Cho, D. K., Wolman, A., Saroiu, S., Chandra, R., and Bahl, P. MAUI: making smartphones last longer with code offload. In Proceedings of the 8th international conference on Mobile systems, applications, and services (pp. 49-62), ACM, 2010, doi=10.1.1.174.5663

16. Nauman, M., Khan, S., and Zhang, X. Apex: extending android permission model and enforcement with user-defined runtime constraints. In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security (pp. 328-332), ACM, 2010, doi=10.1145/1755688.1755732

17. Ongtang, M., McLaughlin, S., Enck, W., and McDaniel, P. Semantically rich applicationcentric security in Android. Security and Communication Networks, 5(6), 658-673, 2012, doi=10.1109/ACSAC.2009.39

18. Zhou, Y., and Jiang, X. Dissecting android malware: Characterization and evolution. In Security and Privacy (SP), 2012 IEEE Symposium on (pp. 95-109). IEEE, 2012, doi=10.1109/SP.2012.16

19. Chin, E., Felt, A. P., Greenwood, K., and Wagner, D. Analyzing inter-application communication in Android. In Proceedings of the 9th international conference on Mobile systems, applications, and services (pp. 239-252). ACM, 2011, doi=10.1.1.228.2624

20. Camp, T., Boleng, J., and Davies, V. A survey of mobility models for ad hoc network research. Wireless communications and mobile computing, 2(5), 483-502, 2002, doi=10.1.1.13.268

Appendix C: Documents and URL used as input for the web application

1. "Intelligent Transport Systems" `http://www.etsi.org/technologies-clusters/technologies/intelligent-transport` Last accessed June 7, 2014

2. Registration history of the second author of this report. The document was generated on 28 January, 2014.

3. "Privacy in the context of Smart Home Environments - Based upon a survey of experts " `http://web.it.kth.se/~maguire/DEGREE-PROJECT-REPORTS/140528-Jahaivis_M._Arias-with-cover.pdf` Last accessed June 7, 2014