Proposal Matching

by

Omkar G. Gurav

Bachelor of Computer Engineering, University of Mumbai, 2011

Report Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Computer Science

in the Graduate Academic Unit of Computer Science

Supervisor:	Yevgen Biletskiy, Ph. D., Department of Electrical and Computer Engineering
Examining Board:	Weichang Du, Ph. D., Faculty of Computer Science
	Michael W. Fleming, Ph. D., Faculty of Computer Science

This report is accepted by the
Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

April 2015

©Omkar Gurav, 2015

Abstract

The wide use of semantic information technologies helps to extract information from vast data available in electronic documents from industrial as well as academic background. The structures of documents are very diverse; therefore, document extraction, processing and interpretation are very difficult procedures. There are many software tools available for these purposes, but they are very domain specific or scenario specific. Therefore, it is impossible to use these software tools for universal purposes and hence, it is essential to develop a tool that is very specific to the problem of information extraction from electronic documents and evaluation.

The present work is part of the final goal to create the "Research Map" of New Brunswick, which has meaningful information about the researchers and their domain of work from industrial and academic background. The purpose of the system is to identify research collaboration, which can indirectly lead to opportunities for funding. Accomplishment of the goal requires development of an application specific information extractor, classification algorithm and methods required to search information for the "Research Map". The work is inspired by the New Brunswick Innovation Foundation.

The present work involves classification of research proposals, extracting keywords based on their domain, and then comparing the same section of two research proposals by implementing comparison algorithm using TF-IDF and cosine similarity. The present work also implements the GUI using Java Swing to facilitate the ease of using the developed system.

Dedication

To my parents and beloved one for all the continued support.

Table of contents

Abstract ii
Dedication iii
Chapter 1: Introduction 1
Chapter 2: Objective 3
Chapter 3: Literature Review 4
3.1 Matchmaking 4
3.2 Search in Text Documents 5
3.3 Keyword Based Search 6
3.4 Term Frequency Inverse Document Frequency 6
3.5 Cosine Similarity 7
3.6 Data Comparison process 8
3.7 Information Extraction 8
Chapter 4: System Architecture 10
4.1 Conversion of HTML to XML 11
4.2 Classification of XML documents 12
Chapter 5: System Implementation 15
5.1 HTML to XML conversion 15
5.2 Keyword selection 18
5.3 Comparator 20
5.4 Graphic User Interface 26
Chapter 6: Quality Assurance 30
Chapter 7: Conclusion and Future Work 33
References 35
Appendix A: XML File of a Proposal 37
Appendix B: Software Source Code 42
Curriculum Vitae

Lists of Figures

Figure 1: Architecture Overview 10
Figure 2: Information Extractor 15
Figure 3: Keywords selection..... 19
Figure 4: Comparator 20
Figure 5: JWI and WordNet..... 21
Figure 6: Hypernym 22
Figure 7: GUI Trigger 27
Figure 8: Input Layout 28
Figure 9: Output layout 29

List of Tables

Table 1: Term frequency for document 1. 23
Table 2: Term frequency for document 2 23
Table 3: Normalized term frequency for document 1 24
Table 4: Normalized term frequency for document 2 24
Table 5: IDF of all terms 25
Table 6: TF-IDF of matching elements 25

Chapter 1: Introduction

Currently a vast amount of data is stored in the form of documents from both academic and industrial domains. The documents published, patented or presented enable us to find the information regarding the interested research area. Comparison of such documents is conducted to identify the common areas of the documents. Presently, this identification is performed manually, which is very time consuming, hectic and sometimes even practically impossible.

Data extracting software system and algorithm are available to automatically extract, process and interpret the information from the text. The problem with available software systems and algorithms is that they are specific to particular domain or application and do not consider the structure of documents. The existing systems cannot be used as a universal tool for information extraction and interpretation of text. Hence, they cannot be extended to universal research comparisons.

The documents to be automatically processed consider several tasks:

- Comparison with other documents;
- Information extraction;
- Interpretation of the data;
- Analysis of similarities in different contexts like academic aspect, industrial aspect, etc.

The present project is a part of ultimate goal to build a network of academic and industrial researchers in New Brunswick, who are working in same field, so called as

“Research Map”. It will allow researchers to identify the opportunities of collaboration between academia and industries. To achieve the ultimate goal, the initial requirement is to learn all the research proposals and compare how similar they are. The present work focuses on achieving this goal.

The present research work describes the software tool to extract specific components of the proposal according to the domain and compare pairs of the research proposals to build the “Research Map”.

Chapter 2: Objective

The ultimate goal of the project is to create a “Research Map” of New Brunswick. The research map is the information about the researcher in the industrial and academic domains. The present work is to extract information from the research proposals, which are submitted from industrial and academia about its domains of work done, and then map the information retrieved to find the documents from the same field. The implementation of this idea will result in an application that solves the information extraction problem for documents, which do not include explicit information about the domain of the contents. The information extraction in the present work identifies the domain specific words, which are important for comparing the documents. The common words are ignored or given less importance during comparison and then text analysis is carried out using natural language processing techniques to obtain information about the domain for those texts. The objectives for the present work are as follows:

1. Find the existing method to distinguish common words and domain specific words from the proposal documents, and extract only sections that have significance while comparing the proposals;
2. Find the way to use the existing methods and tools for information extraction to classify different significant sections from the research proposals;
3. Develop the system for information extraction from research proposals;
4. Find the way to compare the words extracted from information extractor - e.g. tf-idf or cosine function;
5. Find similarities between research proposal and create the “Research Map”.

Chapter 3: Literature Review

In this section, overview of the research carried out for development of the present work is discussed. Various information data extraction methods were reviewed. The present work uses the Information extraction technique discussed in research paper [8], and is discussed in more details in later part. The extracted information is then matched, so concepts for matching the text are used. Further, two comparison methods are discussed:

1. Keyword based comparison;
2. Concept based comparison.

Then proposed report discusses the "tf-idf" method for comparing the documents. The method blends both the Keyword-based and Concept-based methods of comparing documents.

3.1 Matchmaking

Matchmaking is to find the semantic similarity between the content of the same domain extracted from two different documents using the Information Extractor tool. There are various ways in which matchmaking can be implemented. Semantic similarity is one of the approaches from Natural Language Processing (NLP) as discussed in [3]. In this case, service description is considered as query, and information retrieval based on semantic similarity model is applied for the purpose of matchmaking. In the present work, this semantic type of matchmaking model is used for matching the domain of the research proposal.

The other approach is based on ontology for matchmaking task. Ontology-based matchmaking is discussed in the research paper [4]. In [4] author builds an e-Advising system. In this work, e-Advising system tries to automate the process of transferring the credit from one university to other. The system is built using ontology in RDFS in [5].

Paper [6] discussed another matchmaking solution. The system is provided with the buyer's and seller's needs. The needs are identified and then compared. The system finds the similarities with buyer and sellers requirements. In this work, buyer is considered as Learner and seller as Learning Object. The paper uses the concepts of Learning Object Metadata (LOM) and RuleML based similarity engine.

3.2 Search in Text Documents

The search in text documents is been studied since long time. The information content retrieved can be from a local repository or from the Web. Most parts needed for search on the Web and local repository are same with some exception. The exceptions are, firstly there is a huge amount of content on the Web, and hence, it is important for the web search to be efficient to handle such large amount of data. Secondly, the contents on the Web are very dynamic in nature. On the Web, the pages are published at fast rate. Also, editing of the old pages needs to be considered. Hence, a separate algorithm should take this into account for the flexibility to edit these existing pages. Thirdly, the web pages are interconnected, so the connected web pages add information to existing pages.

The common method to search the Web and local repositories are the keyword-based search and semantic-based search. In the keyword-based search, exact words are

searched in text, while in the semantic-based search we take semantic relations between words and documents into consideration.

3.3 Keyword Based Search

Many document search methods are based on finding the exact query word in the document or different forms of query word. In this type of search, the user enters a query with a number of words, and then the search method tries to find the exact words in documents or other forms of the words. For instance, if the query string is "test program", then it provides documents containing both the words exactly, as well as documents containing only one exact word. The system might also display the documents containing both the words, but also all the documents with different forms of those words. The different forms of the word "test" can be "testing", "tester", "testable", etc., and the different forms of the word "program" can be "programs", "programming", "programmer" and other.

3.4 Term Frequency Inverse Document Frequency

Term Frequency Inverse Document Frequency (TF-IDF) is a very commonly used methodology for comparing text documents. Term frequency is number of times the term or words appears in a document. Document Frequency is a number of documents where the words are present at least once.

The documents can be considered as vectors ($\vec{d} = (\, d^{(1)}, \cdots, \, d^{(|F|)})$), which contain keywords and the frequency of keywords in these documents. The vector is represented as TF(w, d). The inverse document frequency IDF(w) is evaluated using following formula:

$$IDF(w) = \log(\frac{|D|}{DF(w)}) \quad (1)$$

Where, |D| = total number of documents;

DF(W) = document frequency of keywords.

The IDF(w) is high if the word occurs in only one document, which is more important for distinguishing this particular documents. The value of keyword in a document is calculated by product of Term Frequency and Inverse Document Frequency and can be described as follows:

$$d^{(i)} = TF(w_i, d) * IDF(w_i) \quad (2)$$

3.5 Cosine Similarity

The similarity is calculated by using by calculating the cosine value between the vectors of words in two documents. The similarity can be calculated by comparing the vectors. The formula to calculate the cosine value of two vectors can be given as follows:

$$\cos(\vec{d_\iota}, \vec{q}) = \frac{(\vec{d_\iota} * \vec{q})}{(||\vec{d_\iota}|| * ||\vec{q}||)} \quad (3)$$

The documents are sorted in descending order of the value in the equation above.

3.6 Data Comparison process

To compare the domain specific test, the words, which have no meaning, are eliminated first, and then the words with similar meaning are found, and the number of words matched is calculated, and then the similarity between these domains is retrieved. Then the similarity between two documents is calculated using "if-idf".

The present work uses WordNet [10] to retrieve hierarchy of the concepts to perform finding the keyword and concept based similarity. WordNet is used for the present project because it gives open access and a very useful resource.

3.7 Information Extraction

Currently almost all the research work can be found on the Web, and hence, it is necessary to design a system, which considers the web format. So, the present system is working as a web information extraction (IE) system. The difference between the web information extraction and traditional information extraction is that the data fed to the traditional information extractors is usually unstructured, while the data for the web information extractors is usually semi-structured.

Classification of documents and IE methods is discussed in [4] and [7]. There are several approaches, which can be used for information extraction:

- Manual Intervention - manually writing an extraction program for each document or set of similar documents.

- Wrapper learning - First, the user manually labels text within a set of training documents and a learning system generates rules from the training pages. Then these rules are applied to extract target items from documents. This approach is

applied mostly to HTML, Word and Excel documents.

- Pattern recognition (automatic method) – aimed to find patterns/grammars from documents and then use them to extract data. Patterns can be defined manually or using machine learning algorithms.

- Natural language processing and machine learning to learn extraction rules from manually labeled (annotated) examples or common language patterns which express facts, relationships and rules.

The web pages are usually structured with Document Object Models (DOMs) of HTML documents like tables, list and other partially organized structures [1]. The Information extraction systems, which processes web documents, are called as 'Web Wrapper'. There are several ways of implementing the Web Wrapper. They can be broadly classified in four categories according to the research work [2]. The four categories mentioned are: manually constructed web wrapper, supervised, semi-supervised and unsupervised web wrapper [2]. There are different tools that can be utilized for building Web Wrapper such as HTML-aware tools, NLP-based tools, Wrapper induction tools, modeling-based tools and ontology-based tools [1].

In the present work, ontology-based tools are used. The web extraction uses the technique of converting HTML based documents to structured XML documents. The proposed Information Extractor converts the HTML documents into XML format, which has tags without styling and formatting data as in HTML documents. The data extracted is categorized in specific subject domains, the present work uses the concept described in the system for information extraction from syllabi [3].

Chapter 4: System Architecture

The top-level view of the system architecture can be described in Figure 1.

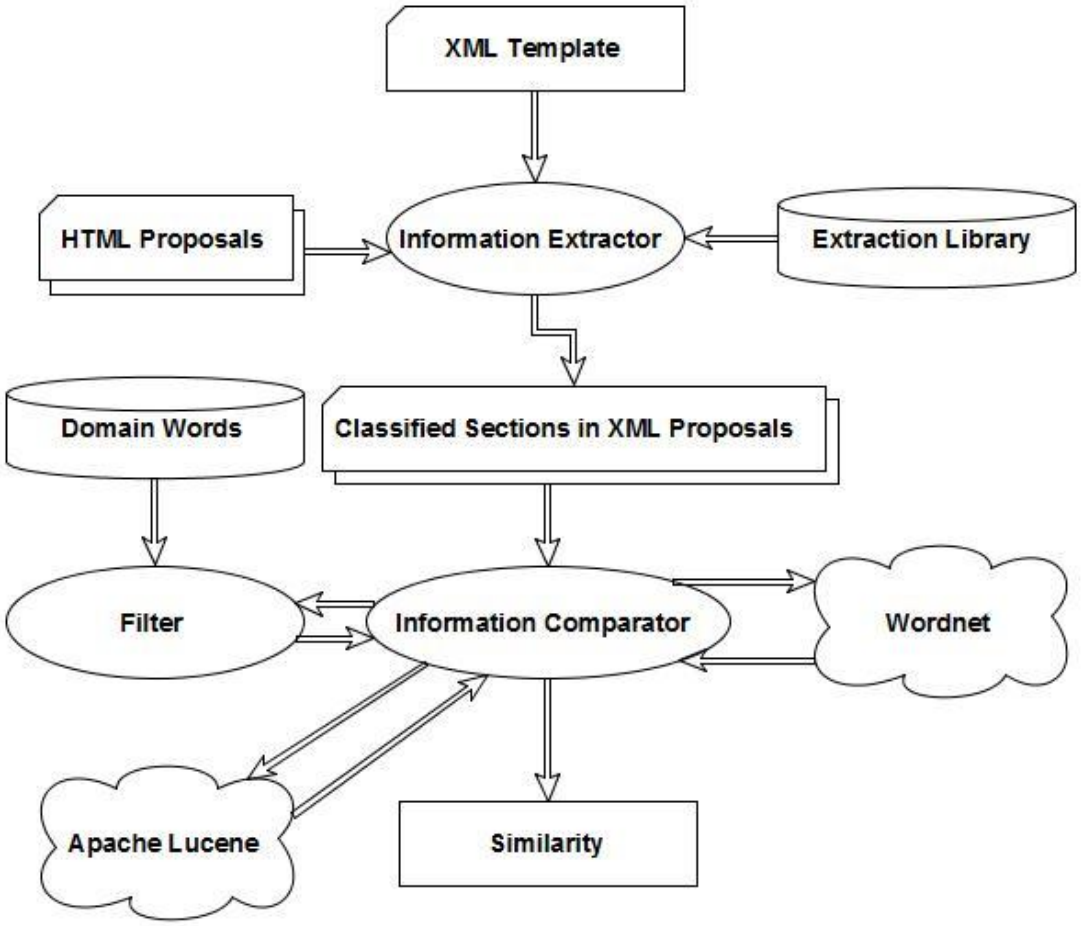

Figure 1: Architecture Overview

The system mainly consists of two parts: Information Extractor and information comparator. In the first part of the system which is Information Extraction (IE), its function is to extract information from the report and categorize the content according to its domain like the content relevant to industries, content related to academic background or content related to sponsors.

4.1 Conversion of HTML to XML

The first phase is similar to the Course Outline Data Extractor (CODE) application [8]. The application retrieves text from semi-structured HTML documents and stores in machine readable XML format. The HTML data extracted are then put into the predefined XML tags according to their domain. The input to this part of the system is the research proposal (HTML file), predefined template (XML file) and several libraries of pattern and key terms.

The first phase of the system is to convert the HTML to XML. The system parses the HTML file and creates a DOM structure, and then uses various Information Extraction algorithms. The Information Extraction algorithms extract the text in specified tag in XML template file and then put the text into another XML file. The XML template has all the information regarding the tag from which the text is to be extracted according to its pattern. It has the specific tag between which the text is to be retrieved. The tag names in the template specify the text to be extracted form research proposal. The sections from research proposal, which are of interest, are stored in the defined tags in the XML file. The list of key terms are used to uniquely identify sections of interest from the sections in the proposal. The libraries of key terms are maintained manually, so the sections to be extracted are manageable.

The final output of the system is the machine-readable XML document with the sections of interest from a research proposal, which are stored in defined tags. The XML documents make it simpler to structure XML documents, process the extracted part and assign them to specific domain.

The HTML to XML conversion part consist of three components, they are as follows:

1. Pre-processor;
2. HTML parsing and DOM building;
3. Information Extraction.

The pre-processor component deletes unnecessary tags, styling information and white spaces, which are very common in HTML documents. The objective of this component is to only keep the tags and text important for further processing.

The next component is HTML parser is DOM building. This component parses the HTML documents, and then the DOM is created form the nodes, where each node is labelled as either HTML tag or the document content.

The last component is information extraction identifies the content of each node in the DOM using libraries of keywords and relate them to appropriate XML tags of the XML template. The component uses keywords and pattern recognition algorithm to retrieve text from the HTML document and put the appropriate parts of text in specific tags in the XML file, which is related to a domain.

4.2 Classification of XML documents

The output of the component mentioned in Section 3.1 is machine readable XML file from the HTML file of a research proposal. The XML file has all the useful information in defined tags within the file.

The next phase is to process the text in defined tags in the XML file. This phase uses the list of words in the file separated with domains and sub-domains. The number of words

belonging to different domains is calculated in combination with its frequency. The final ranking of the domain for data is determined by maximum number of words in a specific domain with their frequency. This data will be useful for creating "Research Map" in future work.

The input to this component is an XML file, which has content extracted form HTML file, list of words according to its domain. This list of keywords is used to select the keywords from XML presented in the list. The selected keywords are then analyzed, and the number of keywords belonging to a specific domain and the number of occurrence are calculated.

This component can be further decomposed into three components as follows:

1. Keyword matching;
2. Calculation of keyword frequency;
3. Determining the similarity.

The keyword matching algorithm is implemented using Java. The list of words are stored in a text file according to their domains. The Files class is used to read the file and then it is stored in List object. After we have the List object, the keywords from XML file is compared to it. The comparison is done based on ontology. The ontology used to conceptualize the keywords is classification in WordNet. In WordNet a group of words that are semantically equivalent are called "synset". Each "synset" is a concept. The "synset" can be sub-concept of other "synset" in Wordnet. In the present work, "hypernym" is used as a super-concept of the words and then it is compared with the list.

The calculation of number of keywords matching the “hypernym” of the keyword in the list to “hypernym” of the words in the XML file, and the frequency of positively matched cases is implemented in the second component. The concept of “hypernym” is explained in more details in Section 5.3. This data is then sent to next domain to calculate the similarity between the XML file compared using “tf-idf”.

Chapter 5: System Implementation

5.1 HTML to XML conversion

The general idea for implementation of the conversion component is described in Figure 2.

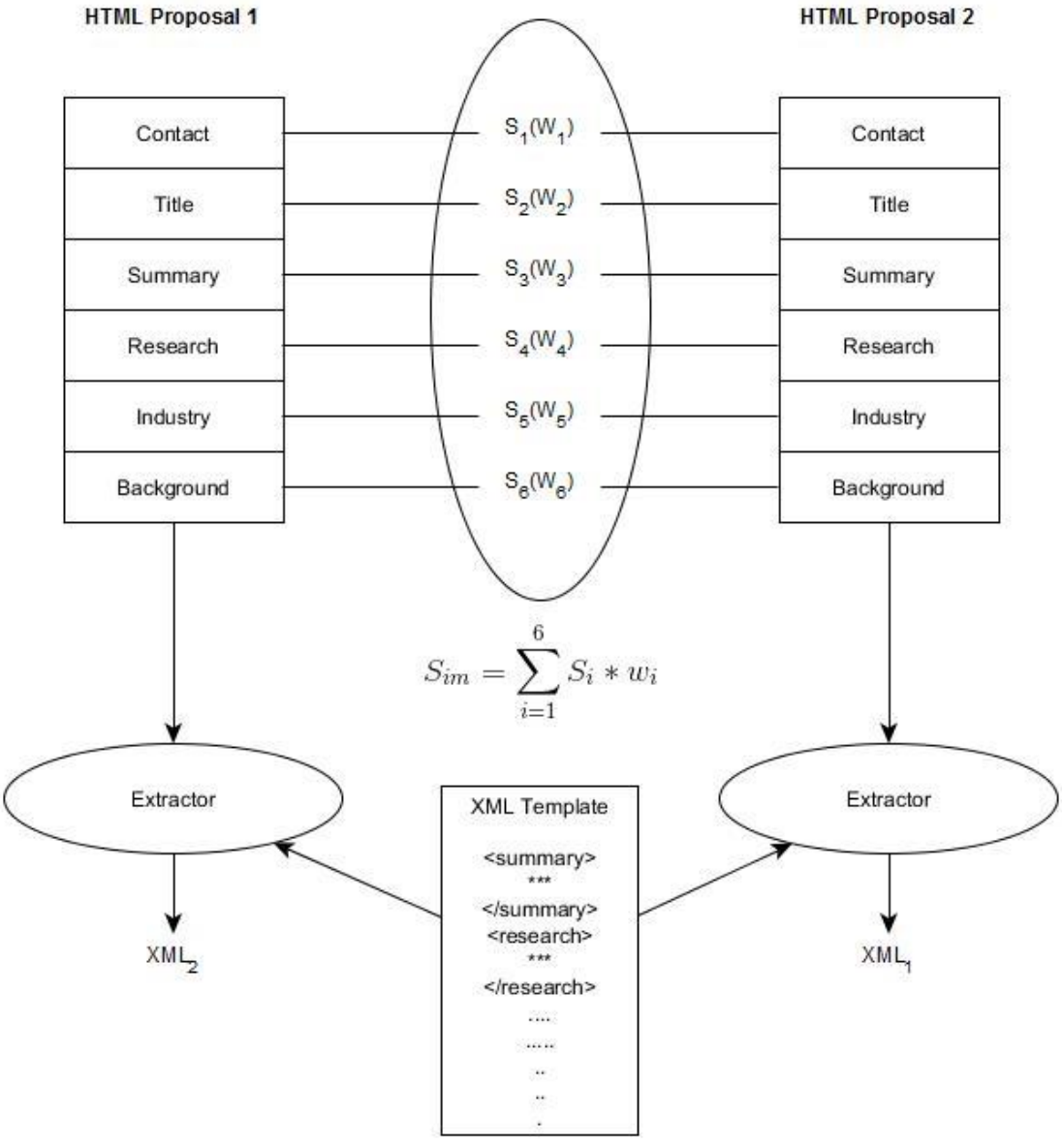

Figure 2: Information Extractor

Initially the HTML file is parsed and then styling information and white spaces are removed. The result of this part is stored in a file, which can be read easily and DOM structure can be created. The HTML parser is used to parse text documents into nodes, and then recursive algorithms are used to visit each node and determine if it is a HTML tag or if it is content. The DOM is constructed on its basis. There are several Information Extraction algorithms, which are applied to the DOM created to extract relevant section of the documents. The Information Extraction algorithm does the major task of differentiating the important section like core section of research proposal and general information section, such as name of researcher, title, abstract, etc.

The extraction task is performed by parsing the DOM and searching keywords or patterns specific for a section. The extracted text is then stored in the XML file within defined tag for that specific section. For analysing each node of DOM, libraries of keywords and patterns are used to distinguish between abstract, introduction, core technical content and other relevant technical sections with in the research proposal. The libraries of keywords are stored in text files, and patterns are stored in the form of XML files, and they both are given as inputs to GUI. The keywords and patterns are retrieved from the files and are then given to Information Extraction to perform its task. The files define the keywords and patterns used to extract the interested section from research proposal. This functionality allows having flexibility to feed input, which can be different format, it makes the software tool independent of the format of the research proposal. The text files with keywords can be easily modified by a non-programmer to add new subject domains.

An example of an XML template is as follows (the XML file of the research proposal given in Appendix A):

```
<document>
      <proposal>
            <type>1</type>
            <title_tag>2.1 – Project Title</title_tag>
            <researcher_tag>1.1</researcher_tag>
            <email_tag>1.7 E-mail</email_tag>
            <summary_tag>2.4 Plain-language summary of
project</summary_tag>
            <description_tag>2.5 Detailed technical
description</description_tag>
            <innovation_tag>3.1 – Describe how your Project will
generate new intellectual property</innovation_tag>
            <innovation_tag2>3.2 </innovation_tag2>
            <innovation_tag3>3.3 </innovation_tag3>
            <expertise_tag>3.4 – Demonstrate the academic and
research expertise</expertise_tag>
            <patternStart>19.85pt'></patternStart>
            <patternEnd1>&#60;</patternEnd1>
            <patternEnd2>/td></patternEnd2>
      </proposal>
</document>
```

The example output of the HTML to XML conversion is as follows:

```
<document>
      <proposal>
            <title> Integration of Web-based Business Rules</title>
            <researcher> Yevgen Biletskiy </researcher>
            <email>biletski@unb.ca</email>
            <summary> The wide proliferation of semantic information
technologies enables computer software to access and automatically
interpret information from a...</summary>
            <description> At the present time, information is stored
in a vast amount of documents created by many people with various
business interests, preferences and backgrounds. One important type
of such a document is one which store business rules. Classically,
business rules are logic...</description>
            <innovation> The project will ultimately generate new
intellectual property in the form of new knowledge-based software
tool prototypes to perform interoperation among business rules
created in various formats. This...</innovation>
            <expertise> Dr. Yevgen Biletskiy has been leading
research in the areas of semantic information technology and
knowledge engineering for the past 18 years, which includes his PhD
work, in particular...</expertise>
      </proposal>
</document>
```

5.2 Keyword selection

The present system is implemented in Java using Apache Lucene and WordNet as shown in Figure 3. The initial step is to retrieve text from the XML file generated from the Information Extractor. The text to be processed by the Information Extractor has most of the words, which are for grammar, so contribute no meaning to the sentences. Such words are called Stop Words. Apache Lucene is used to filter out Stop Words. Apache Lucene has default list of Stop Words to filter, but it is not a complete list. In the present work, list of Stop Words are supplied externally. The list of stop words used is from https://code.google.com/p/stop-words/ website, which is extensive and covers most of the Stop Words used in most sentences.

The output after filtering out Stop Words is the List of meaningful words. The list of words fetched after filtering stop words may have the words, which may not be domain specific, for example the words like student, research, property, and so on. These words cannot be used to compare the research proposal as most of the research proposal might have these words. If these words are included while comparing, the similarity value won't give appropriate idea about similarity between two research proposals. So, only the words that are specific to the domain should be compared, but the other words should not be considered or should be given less importance during the comparison process. In the present work, the less important words are completely filtered out and only domain specific words are used for comparison. The domain specific words and "hypernym" of those words, which is retrieved using WordNet, are considered.

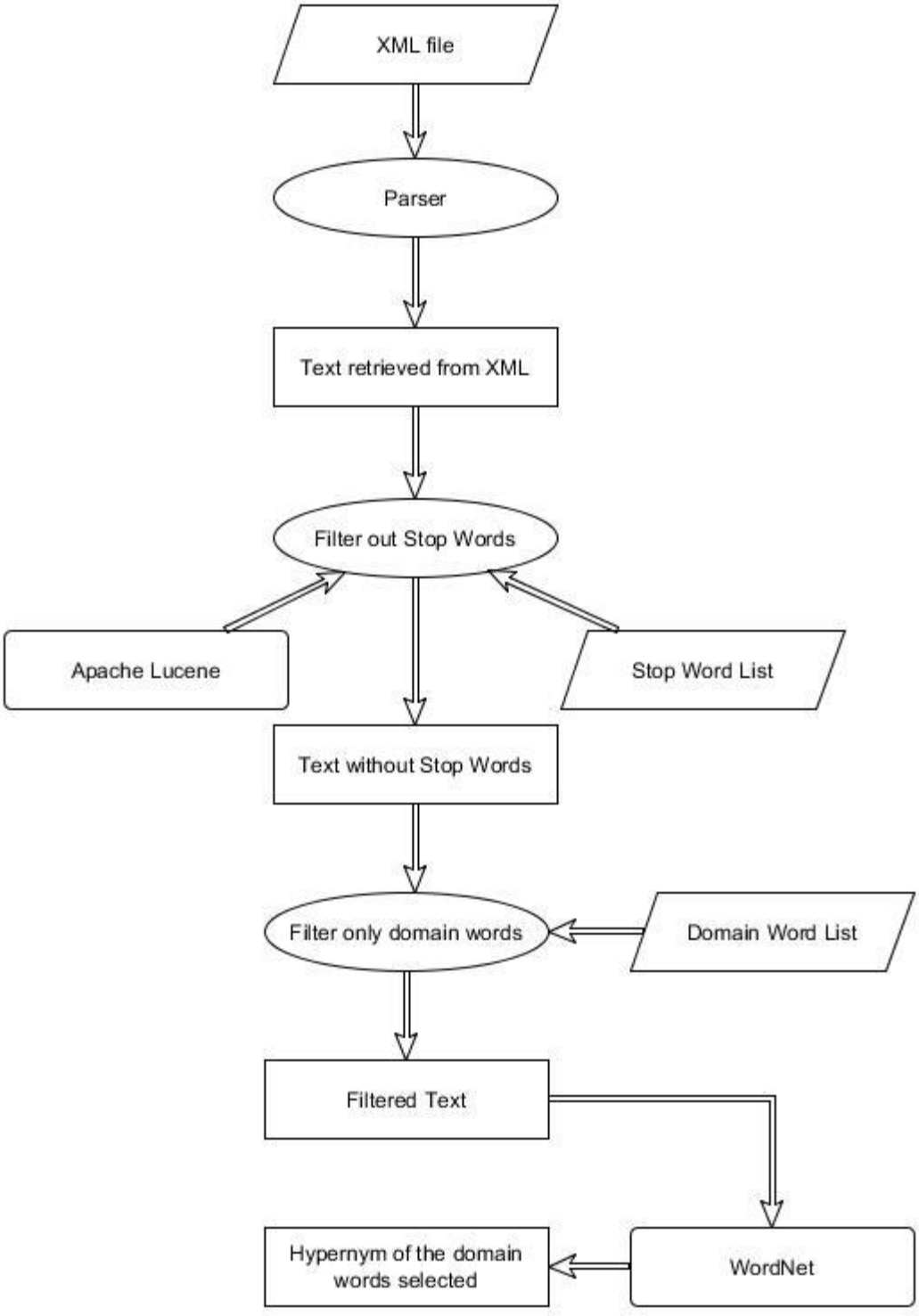

Figure 3: Keywords selection

5.3 Comparator

The Comparator part (Figure 4) is responsible for comparing of two XML files as shown in figure below. The output of the Comparator is the percentage of similarity of the documents. The Comparator also includes the keyword selection for the comparison process.

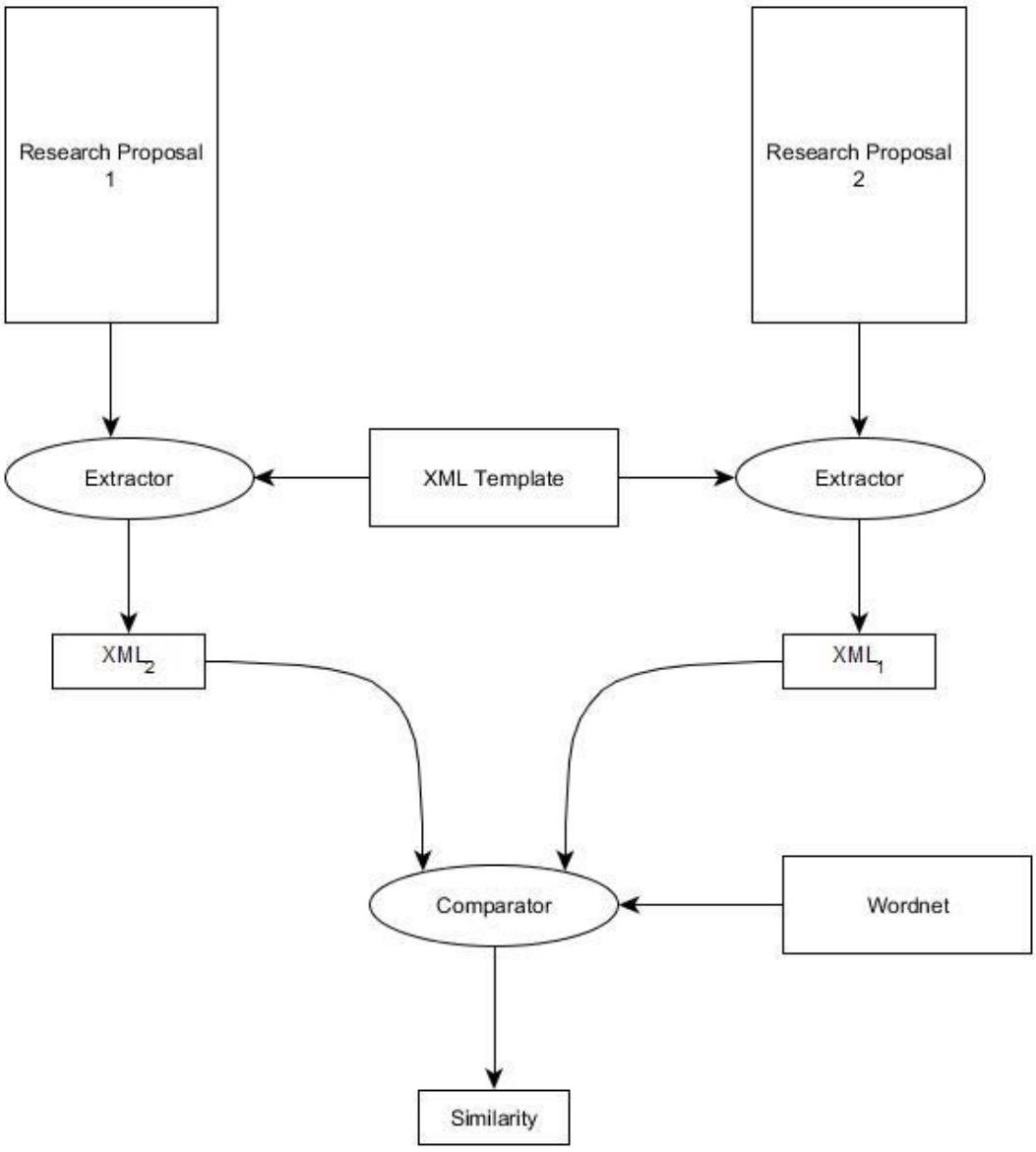

Figure 4: Comparator

The comparator after selecting keywords, it gives List as an output (Figure 5). This list of keywords are then given to WordNet to retrieve its higher level semantics called "hypernym". The API library used to communicate with WordNet in Java programming language is the Java WordNet Interface (JWI). The JWI is developed at MIT. JWI library 3.0 version is used in the present work. JWI has less memory footprint, has high-performance, it is open source, and it supports both file based and in-memory implementation [9].

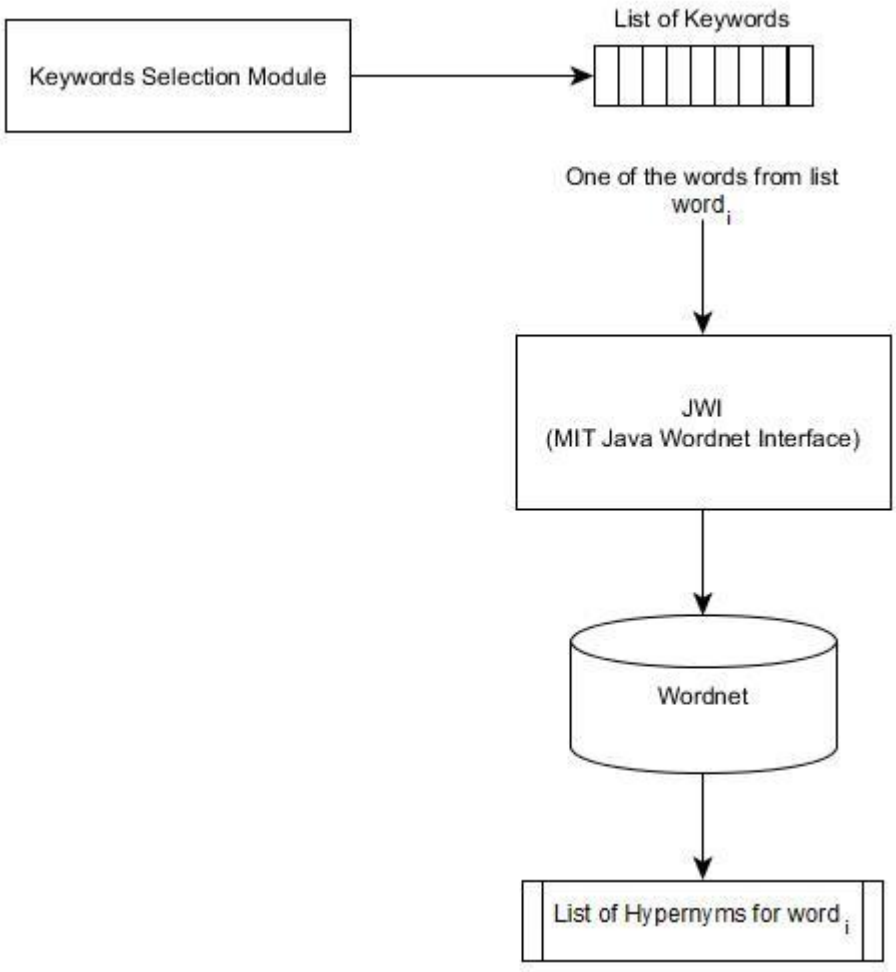

Figure 5: JWI and WordNet

The comparator main objective is to compare the keywords present in both the research proposal. If more keywords are found matching between both the proposals then it can be considered as the proposals are very similar to each other. It is one of the way to check for similarity. In the present work, instead of checking for similar keywords, the "hypernym" of the keywords are matched, which are retrieved using WordNet as explained in Figure 6. .

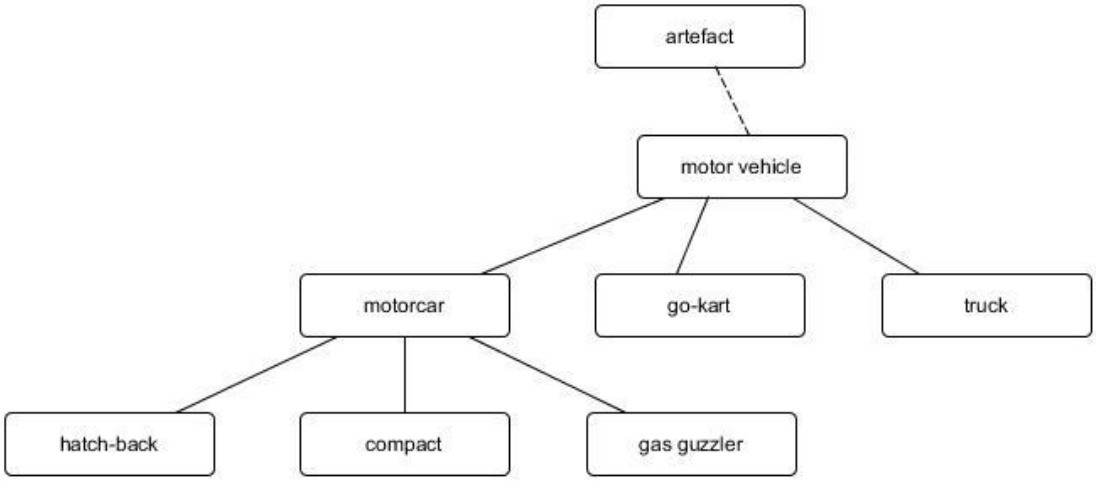

Figure 6: Hypernym

In present work, for example, one of the research proposal has keyword 'hatch-back' while in other research paper the keyword present is 'compact'. Both the research paper discusses about motorcar by giving these two keywords as reference. If only keywords are matched then these two keywords won't be considered as similar even though the research paper are meaning same with just different keywords. So, to avoid this problem "hypernym" are used. As per the present work the "hypernym" of 'hatch-back' is motorcar and same is the "hypernym" for 'compact' and so they are matched and similarity between these two keywords are considered.

After retrieving the keywords that are matching using “hypernym” and frequency of its occurrence, the similarity between two research proposals of the same section is calculated using “tf-idf” and cosine similarity. The “tf-idf” and cosine similarity can be explained with the example as follows.

Consider two research proposal d_1 and d_2 has the keywords are given below:

d_1 = retrieval | research | tfidf | cosine | wordnet | research

d_2 = comparator | tfidf | tfidf | cosine | jwi | cosine | comparator | swings

The term frequencies are presented in Table 1 and Table 2.

Table 1: Term frequency for document 1.

d_1	retrival	Research	tfidf	cosine	wordnet
Term Frequency	1	2	1	1	1

Table 2: Term frequency for document 2

d_2	comparator	Frequency	tfidf	cosine	jwi	swings
Term Frequency	2	1	2	2	1	1

The term frequencies mentioned above are the words and the number occurrences in the document. The term frequencies mentioned are absolute numbers. The next step is to normalize the term frequency (Table 3 and Table 4).

Table 3: Normalized term frequency for document 1

d_1	retrival	research	tfidf	cosine	wordnet
Normalized Term Frequency	0.167	0.333	0.167	0.167	0.167

Table 4: Normalized term frequency for document 2

d_2	comparator	frequency	tfidf	cosine	jwi	swings
Normalized Term Frequency	0.222	0.111	0.222	0.222	0.111	0.111

The next step is to calculate Inverse Document Frequency (IDF) and is done as follows:

$$IDF(w) = \log(\frac{|D|}{DF(w)})$$

Where, |D| = total number of documents,

DF(W) = document frequency of keywords.

Now, IDF for each word is calculated and the output is as presented in Table 5. The value of IDF is added by 1 so that a non-zero value of term is present in both the documents every time.

Table 5: IDF of all terms

Terms	IDF
retrieval	1.693
research	1.693
tfidf	1
cosine	1
wordnet	1.693
comparator	1.693
frequency	1.693
jwi	1.693
swings	1.693

Now "tf-idf" of matching words of the documents are calculated by taking product of TF and IDF. The output is as given in Table 6.

Table 6: TF-IDF of matching elements

Terms	d_1	d_2
Tfidf	0.167	0.222
Cosin	0.167	0.222
Retrival	0.283	0
Research	0.565	0
Wordnet	0.283	0
comparator	0	0.376
frequency	0	0.188
Jwi	0	0.188
Swings	0	0.188

Now, the cosine similarity is used to find similarity between two vectors given in Table 6 is as follows:

$$similarity = \cos(\vec{d_\iota}, \vec{q}) = \frac{(\vec{d_\iota} * \vec{q})}{(||\vec{d_\iota}|| * ||\vec{q}||)} = 0.173$$

5.4 Graphic User Interface

The GUI (Figure 7) is implemented using Java Swing. The GUI triggers the information Extractor, and after the result of comparison is passed to the GUI component, the result is displayed to user.

There are two GUI layouts in the present work. The first GUI layout is for feeding the input, while second layout displays the output. The first layout takes four file inputs and weights of the sections for the proposals to be matched. Two input files are two research proposals of .html or .htm file format, one template file which is of XML file format and one text file with list of domain words in it. The first layout is as shown in Figure 8. The weight input fields for each section of the report must sum to one to be normalized. The similarity value obtained is then displayed in the form of percentage for each section. The total similarity is calculated by average of similarities and the weights entered, and it is normalized.

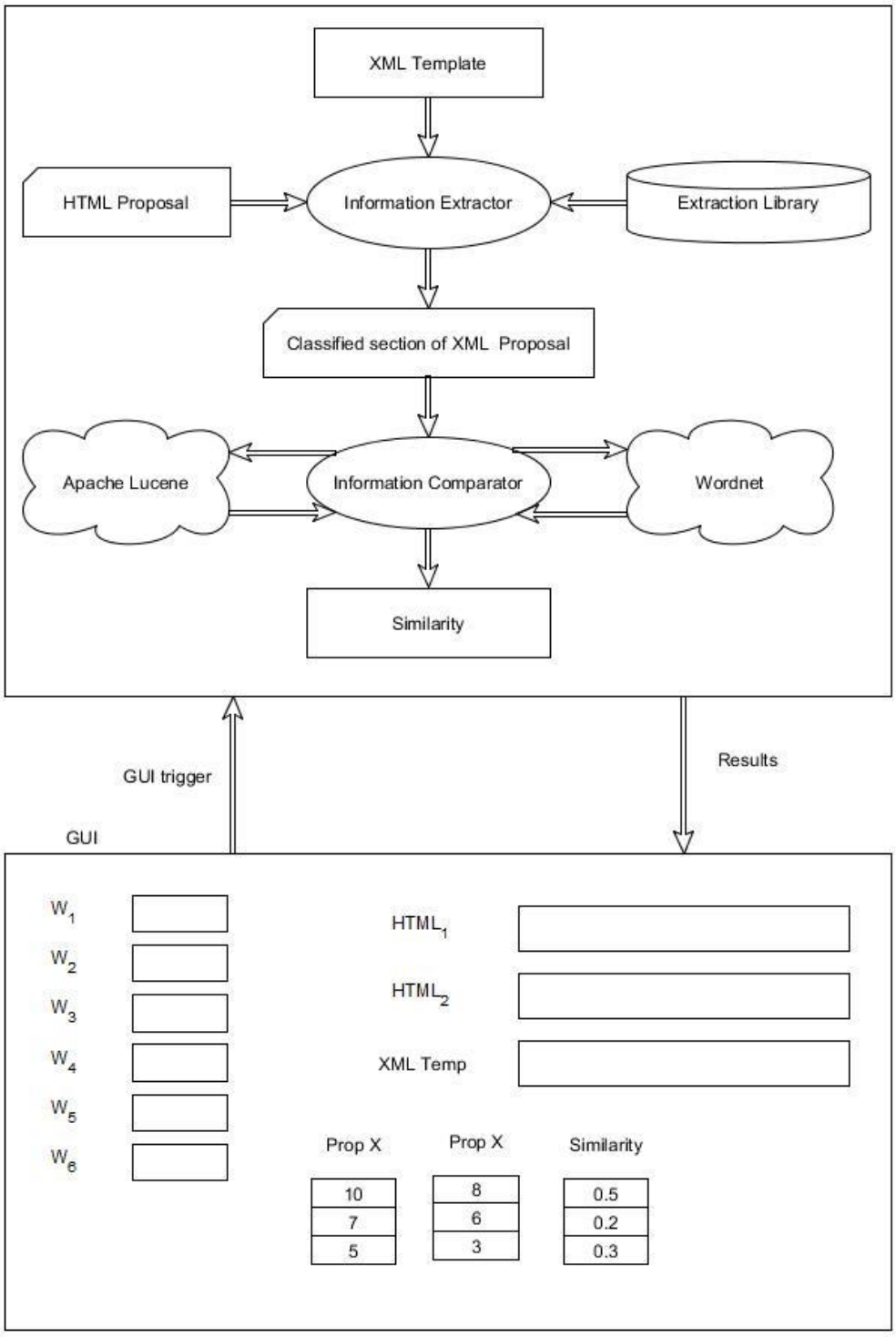

Figure 7: GUI Trigger

The second GUI is for displaying output. The first part gives the details of the first research proposal like research proposal filename, title and researcher. The third column gives details about the second research proposal. The middle column gives details about the similarity between two proposals according to the section and then at last the total similarity. The layout is as shown in Figure 9. The layout also facilitates to export the result in a file and save it for later analysis.

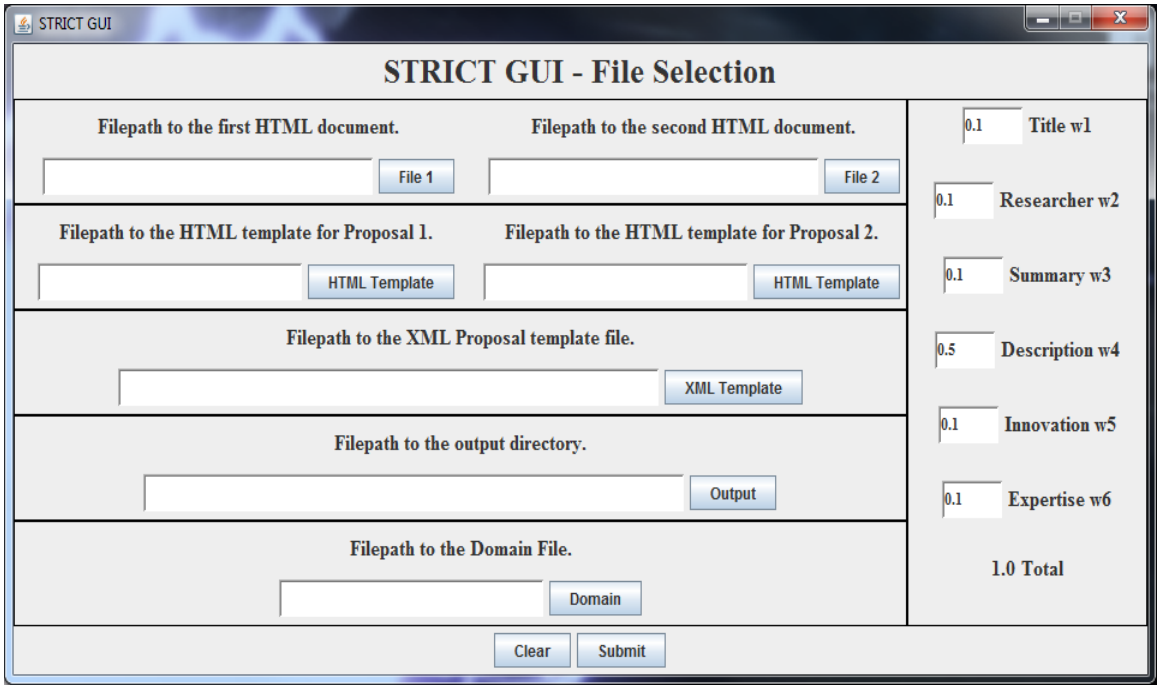

Figure 8: Input Layout

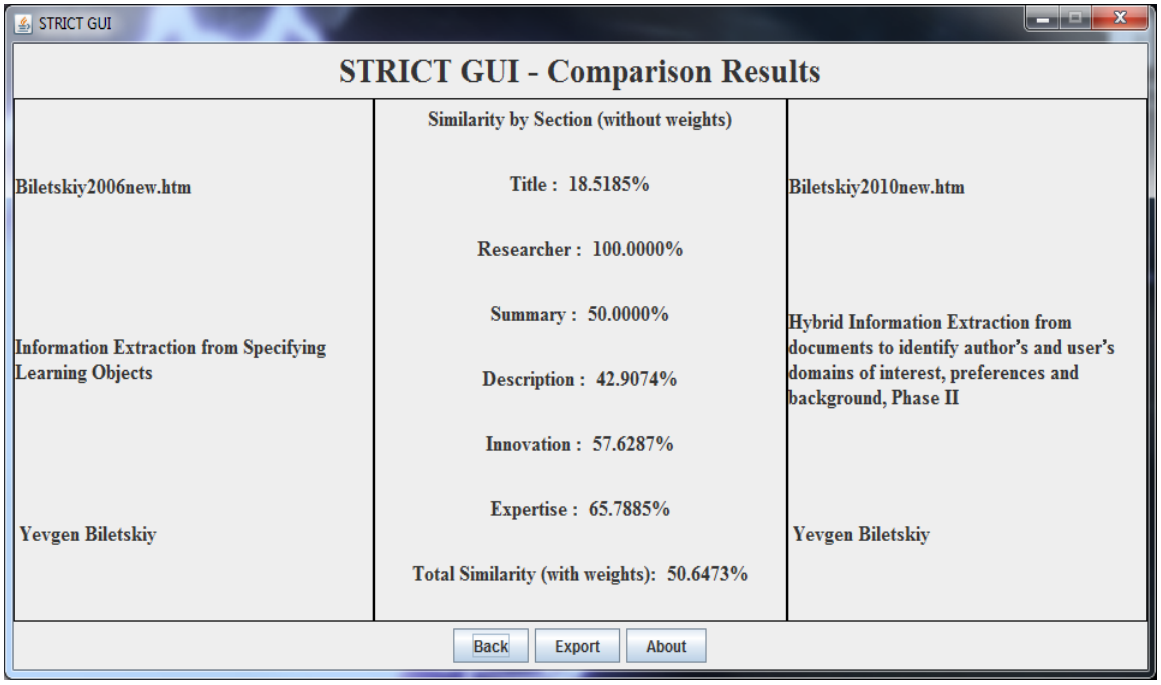

Figure 9: Output layout

Chapter 6: Quality Assurance

The system is tested on Dr. Yevgen Biletskiy's 6 research proposals. These proposals have all been converted to the appropriate HTML format. They are represented as:

Biletskiy2006new.html
Biletskiy2007new.html
Biletskiy2009new.html
Biletskiy2010new.html
Biletskiy2014new.html
Biletskiy2015new.html

These files all conform to the XML template created 'HTML_template_NEW' format. Once the data has been extracted from an HTML file, it is stored in XML files of the same name. The XML file contains the strings that are to be used by the comparator. The output XML files are represented as:

Biletskiy2006new.xml
Biletskiy2007new.xml
Biletskiy2009new.xml
Biletskiy2010new.xml
Biletskiy2014new.xml
Biletskiy2015new.xml

The GUI is designed to compare 2 research proposals at a time. The Biletskiy2010new.html has been chosen as the standard against which all the other proposals will be compared. Each of the 6 test documents will be compared against Biletskiy2010new and their comparison results recorded.

The weight distribution for the total similarity of all comparisons will be as follows:

Title: 0.1
Researcher: 0.1
Summary: 0.1
Description: 0.5
Innovation: 0.1
Expertise: 0.1

The complete extraction of all relevant data from a section of a given HTML file to the corresponding section of the of its XML counterpart. The ability to pass strings to the comparator and receive/display results. The ability of the GUI to take weights, correctly apply them to the results, and sum them for the total similarity. The test result are as shown below.

Comparison between Biletskiy2010new.html and Biletskiy2006new.html

Similarity by Section (without weights):
Title: 18.5185
Researcher: 100.0000
Summary: 50.0000
Description: 42.9074
Innovation: 57.6287
Expertise: 65.7885

Total Similarity (with weights):
Total: 50.6473

Comparison between Biletskiy2010new.html and Biletskiy2007new.html

Similarity by Section (without weights):
Title: 3.9683
Researcher: 100.0000
Summary: 47.3684
Description: 46.8052
Innovation: 81.7370
Expertise: 70.4279

Total Similarity (with weights):
Total: 53.7528

Comparison between Biletskiy2010new.html and Biletskiy2009new.html

Similarity by Section (without weights):
Title: 90.9091
Researcher: 100.0000
Summary: 92.4816
Description: 92.9052
Innovation: 100.0000
Expertise: 99.7948

Total Similarity (with weights):
Total: 94.7711

Comparison between Biletskiy2010new.html and Biletskiy2010new.html

Similarity by Section (without weights):
Title: 100.0000
Researcher: 100.0000
Summary: 100.0000
Description: 100.0000
Innovation: 100.0000
Expertise: 100.0000

Total Similarity (with weights):
Total: 100.0000

Comparison between Biletskiy2010new.html and Biletskiy2014new.html

Similarity by Section (without weights):
Title: 1.4388
Researcher: 100.0000
Summary: 41.8777
Description: 28.5714
Innovation: 80.4530
Expertise: 77.0672

Total Similarity (with weights):
Total: 44.3694

Comparison between Biletskiy2010new.html and Biletskiy2015new.html

Similarity by Section (without weights):
Title: 13.0952
Researcher: 100.0000
Summary: 36.3688
Description: 53.8993
Innovation: 71.3303
Expertise: 77.5000

Total Similarity (with weights):
Total: 56.7791

Chapter 7: Conclusion and Future Work

The main objective of the current work is to find the similarity between two HTML documents, in particular research proposals. The part of the project is the adoption of the concepts developed in the previous project CODE (Course Outline Data Extractor). The CODE generates XML files from research proposals in HTML file format. The XML has the extracted text between defined tags according to the domain of interest, which is further analyzed. The prototype software tool in the present work reads the text from the XML file, selects the keywords according to its domain and calculates its frequency of occurrence. After getting keywords and its frequency of occurrence, WordNet is used to compute higher level of semantics called "hypernym", and then it is compared to "hypernym" of keywords of another XML file, and then similarity is calculated using "tf-idf".

The ultimate goal of the project is to determine the similarity between two documents with respect to its domain of interest. This information of similarity is used to create a vector for all the domains and sub-domains the documents (research proposals) belong to. Based on the vector generated, the network of researchers can be created from industrial as well as academic background according to their extent of similarity between the research papers. In the future work, we can progress towards the ultimate applied goal of the project to create a "Research Map" of New Brunswick using the network of researchers created using the similarity between research proposals.

The present work selects the keywords based on the domains to which it belongs to. The domains are divided into sub-domains, as the present work focuses on calculation with

respect to domain, though it can be extended to the sub-domain level. The list that is generated needs to consider the domain and also subdomain in it. It would also require some modifications in text files generated with list of keywords. The file created must be able to differentiate between domain, sub-domain and keywords. Hence, the current project can also be extended to sub-domain classification level.

The present work also creates the GUI to input two research proposals, the XML template, and the file with the list of keywords according to the domains. The facility to enter weight is also provided to manually give priority to sections of research proposals for finding similarity between them. The GUI then triggers the Information Extractor and then the Information Comparator. The final output is shown on GUI with parts of the research proposal and the similarity value between these parts.

Finally the current objective is achieved by implementing a software tool that analyzes the data of interest form research proposal, compares them and computes the similarity between them. The software tool developed in the present work is an initial work needed to develop a "Research Map" of New Brunswick. Creation of this map requires population of the domains, evaluation and adjustment when more than two proposals are compared.

References

[01] A. Laender, B. Ribeiro-Neto, A. da Silva, and J. Teixeira, *A brief survey of web data extraction tools*, SIGMOD Rec. 31 2002, no. 2, 84--93.

[02] C-H. Chang, M. Kayed, M. Ramzy Girgis, K. F. Shaalan, *A Survey of Web Information Extraction Systems.* IEEE Transactions on Knowledge and Data Engineering, 18/10, 1411-1428, 2006.

[03] A. Hliaoutakis, G. Varelas, E. Voutsakis, E. G. M. Petrakis, and E. Milios, *Information retrieval by semantic similarity*, Intern. Journal on Semantic Web and Information Systems (IJSWIS), vol. 3(3), pp.55--73, July/Sept. 2006. Special Issue of Multimedia Semantics, 2006.

[04] Y. Biletskiy and T. Scribner, *Conversion of learning objects to meaningful xml*, In Proceedings of the 8-th IASTED International Conference on Communication, Internet and Information Technology 2005, pp. 125--130.

[05] *RDF Schema.* W3 digital library. Retrieved July 3, 2012, from http://www.w3.org/TR/rdf-schema/.

[06] H. Boley, V. Bhavsar, D. Hirtle, A. Singh, Z. Sun, and L. Yang, *A match-making system for learners and learning objects*, International Journal of Interactive Technology and Smart Education, vol. 2(3) ,(2005), pp. 171—178

[07] B. Lui, K. Chang, Editorial: Special Issue on Web Content Mining. SIGKDD Explorations, 6/2, pp. 1-4 2004.

[08] Y. Biletskiy, J. Brown, G. Ranganathan, *Information Extraction from Syllabi for e-Advising, Expert Systems with Applications*, Elsevier, 36(3), Part 1, 4508-4516, 2009.

[09] M. Finlayson, *Java Libraries for Accessing the Princeton Wordnet: Comparison and Evaluation.* Proceedings of the 7th Global Wordnet Conference. Tartu, Estonia 2014.

[10] C. Fellbaum, *WordNet: An Electronic Lexical Database*, Cambridge, MA: MIT Press, 1998.

Appendix A: XML File of a Proposal

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<document>
<proposal>
            <title>Hybrid Information Extraction from documents to
identify author's and user's domains of interest, preferences and
background</title>
            <researcher> Yevgen Biletskiy </researcher>
            <email>biletski@unb.ca</email>
<summary>
The wide proliferation of semantic information technologies enables
computer software to access and automatically interpret information
from a vast amount of electronic documents created within various
scientific and educational domains and backgrounds, cultural and social
preferences, and particular interests. Due to the structural and
semantic diversity of such documents, extracting, processing, and
interpreting this information is often inhibited. Although there are
many software systems (products) on the market that promise high
precision of information extraction, in fact they are often very domain
and scenario specific and tailored for particular document structures.
So, it is often impossible to apply existing software as "universal"
tools to extract information from various documents, interpret and
deliver this information to users. Therefore, each concrete task
requires a specific sophisticated approach to extract information and
deliver it to users. The proposed research project will target two
specific sub-goals. The first sub-goal is to create a "Research Map" of
New Brunswick that includes meaningful information about researchers in
academia and industry, as well as industrial and public needs in
research outcomes in New Brunswick. The purpose of this "Research Map"
is to identify opportunities for research collaborations and
innovations in New Brunswick. The second sub-goal is to create an
"Education Map" of New Brunswick that includes meaningful information
about courses and programs offered in New Brunswick educational
institutions as well as how this information is associated with
disciplines listed in the National Occupation Classification (NOC)
list. The purpose of this initiative is to facilitate the Preliminary
Learning Assessment and Recognition (PLAR) and enable learners and help
academic advisors to predict and select the best pathway to achieve the
desired learner's degree level based on the learner's educational
background and competencies. Both of these sub-goals will require
development of specific information extraction and classification
methods and algorithms applicable to source documents containing
information for the Maps . The development of these methods and
algorithms is the main objective of the current proposal.
</summary>

<description>
```

At the present time, information is stored in a vast amount of documents created by many people with various interests, preferences and backgrounds. So, the information in these documents is structurally and semantically heterogeneous that often inhibits extracting, processing and interpreting the information by humans and computer systems, which try to understand its meaning. Although there are many software systems (products) on the market that promise high precision of information extraction, in fact they are often very domain and scenario specific and tailored for particular document structures. So, it is often impossible to apply existing software as "universal" tools to extract information from various documents, interpret and deliver this information to users. Therefore, each concrete task requires a specific sophisticated approach to extract information and deliver it to users (say information consumer). This is how people read and interpret information: we interpret information based on the documents' structure and our background. So, why should a computer (say robot) work better? I would like to describe two real-world scenarios that support the statements above. At the conceptual level of research, the issues described above fall into one category: development of mediation software tools, which enable the extraction of information from documents of various types and deliver this information to users in desirable forms. The mediation means that there is a software agent between information sources (e.g. electronic documents) and information consumers (e.g. individual information consumers or databases that integrate the information provided from sources). This software agent is responsible to extract information from documents, interpret it, store and deliver it to information consumers/users in the most understandable way. Both scenarios described above seem well fit in the research I am working on. Information extraction from documents of various formats is one of the most important components of my research. I propose to develop a hybrid method for information extraction from documents in specific domains and specific structure. The proposed hybrid method blends ontology-based (knowledge-based) and position-based (geographic-based) methods of information extraction. This combination currently works for information extraction from HTML documents generated from databases where are strong patterns (i.e. documents on e-Bay, Amazon, etc.) and even HTML documents generated by humans where there are some structural regions (e.g. syllabi). The system for information extraction from HTML syllabi has been developed under my supervision [1]. The NBIF proposal documents have a well-defined structure, but each section (region) requires the application of a specific information extraction method; moreover, the weighted-tree algorithm developed at UNB might better help to classify NBIF proposals. So, my proposition would work for this scenario. In addition, there are two algorithms, which can be useful to the proposed work: combination of keyword and concept based search [2] and embedding non-destructive annotations [3]. – For references, see section 3.4. The initial idea to handle the problem of information extraction from documents lacking domain-specific jargon (e.g. research proposals) is

to filter not only stop-words (such as conjunctions, prepositions, adverbs, etc.), but also general words used in research proposals. The existing methods of information extraction would actually help to identify such common words as top ranked – this will allow truncating the common words bringing domain-specific keyword to the top.
</description>

<innovation>
The wide proliferation of semantic information technologies enables computer software to access and automatically interpret information from a large amount of electronic documents created within various scientific and educational domains and backgrounds, cultural and social preferences, and particular interests. There are two main outcomes from the proposed research, which can be commercialized: 1. A software tool prototype for information extraction from research proposals with the goal to create a "Research Map" of New Brunswick that includes meaningful information about researchers in academia and industry, as well as industrial and public needs in research outcomes in New Brunswick. The purpose of this "Research Map" is to identify opportunities for research collaborations and innovations in New Brunswick. 2. A software tool prototype for information extraction from university and college calendars, curricula, course descriptions, and description of learner's competencies to create an "Education Map" of New Brunswick that includes meaningful information about courses and programs offered in New Brunswick educational institutions as well as how this information is associated with disciplines listed in the National Occupation Classification (NOC) list. The purpose of this initiative is to facilitate the Preliminary Learning Assessment and Recognition (PLAR). Although the proposed information extraction tool prototypes are specific to documents used in the province of New Brunswick, the methods and algorithms can be adapted to documents circulated in other provinces and other fields (Government, Industry, etc.). For instance, PLAR can be used not only for assessment of learner's credentials by educational institutions, but also by HR departments in industry to properly select best candidates for available positions. PLAR also can be used by learners themselves to select the best pathway for continuing education. So, the tool prototypes themselves can be improved to the level of commercial products, as well as can be used to provide services to interested organizations and/or individuals. In the Innovation Agenda of New Brunswick, information technology, and specifically TeleEducation infrastructure, online education and training, management, delivery tools and technologies, and online university environments, are identified as strategic areas in which New Brunswick has a competitive advantage. The training and research experience these students receive will enable them to contribute to this important component of the New Brunswick economy. The research work will also collimate in an academic thesis for the graduate student. In the long term, when the proposed research and tools are applied: the "Research Map" will be used to

identify opportunities for research collaborations between academia and industries, and innovations in New Brunswick; the created "Education Map" will facilitate the Preliminary Learning Assessment and Recognition (PLAR) and enable learners and help academic advisors to predict and select the best pathway to achieve the desired learner's degree level based on the learner's educational background and competencies.
</innovation>

<expertise>
Dr. Yevgen Biletskiy has been leading research in the area of knowledge engineering for the past twelve years, which includes his PhD work, in particular investigating and developing methods for extracting and representing data semantics in heterogeneous data storages, and implementing these methods to achieve interoperability between semantically heterogeneous electronic documents and information consumers (users). He has published 50 journal and conference papers and one book, For the past five years, much of Dr. Biletskiy's work has been devoted to building and integrating domain ontologies of various electronic documents with the goal of delivering them to the user's context according to the user's preferences. The recent (2008) publications related to this work are the following: 1. Yevgen Biletskiy, J Anthony Brown, Girish R Ranganathan, Information Extraction from Syllabi for e-Advising, Expert Systems with Applications, Elsevier, 36(3), Part 1, pp. 4508-4516, 2009. 2. Yevgen Biletskiy, Hamidreza Baghi, Igor Keleberda, Michael Fleming. Adjustable Personalization of Search and Delivery of Learning Objects to Learners, Expert Systems with Applications, Elsevier (in press), 2008. 3. Hamidreza Baghi, Yevgen Biletskiy, Howard Li. A Blended Approach for Search of Learning Objects, IEEE Canadian Conference on Electrical and Computer Engineering CCECE08, Niagara Falls, Canada, pp. 1805-1809, 2008. 4. Girish R Ranganathan, Yevgen Biletskiy, Alexey Kaltchenko. Semantic Annotation for Semi-structured Documents, IEEE Canadian Conference on Electrical and Computer Engineering CCECE08, Niagara Falls, Canada, 919-922, 2008. 5. Yevgen Biletskiy, Girish R Ranganathan, Olga Vorochek. Identification and Resolution of Conflicts During Ontological Integration Using Rules, Expert Systems: The Journal of Knowledge Engineering, Blackwell, (in press), 2008. 6. Yevgen Biletskiy, Girish R Ranganathan. An invertebrate semantic/software application development framework for knowledge-based systems, Knowledge-Based Systems, Elsevier, Volume 21, Issue 5, pp. 371-376, 2008. 7. Yevgen Biletskiy, Harold Boley, Girish R Ranganathan. RuleML-based Learning Object Interoperability on the Semantic Web, Interactive Technologies and SMART Education, Emerald, volume 5, issue 1, pp. 39-58, 2008. 8. Yevgen Biletskiy, Girish R Ranganathan. A system for automatic conversion of academic transcripts, British Journal of Educational Technology, Blackwell, Volume 39, No 4, pp. 734-737, 2008. 9. Yevgen Biletskiy, Girish R Ranganathan, J Anthony Brown, Representing User-Friendly Business Rules in a Semantic Web-Based

Format, ISAST Transactions on Computers and Software Engineering No. 1, Vol. 2, pp. 8-12, 2008, on-line journal, available: www.isastorganization.org/CS2ready.pdf">http://www.isastorganization.org/CS2ready.pdf 10. Yevgen Biletskiy, J Anthony Brown, Girish R Ranganathan. Information Pre-Processing for Building Domain Ontology, 3-rd International Conference on Cognitive Science, Moscow, Russia, 2008.
</expertise>
</proposal>
</document>

Appendix B: Software Source Code

ExtractionLogic.java

```
      public String extract(String loc, String template, String
outdir){

            loc=loc.replaceAll("%20", " ");
            System.out.println("Converting to String...");
            String fileAsString = fileToString(loc);

            fileAsString = fileAsString.replace("<![if
!supportLists]>", " ");
            fileAsString = fileAsString.replace("<![endif]>", " ");
            fileAsString = fileAsString.replace("<a   href=\"http://",
" ");
            fileAsString = fileAsString.replace("</a>", " ");

            //Apply preprocessing to long String.
            System.out.println("Applying preprocessor...");
            Preprocessor pre = new Preprocessor();
            fileAsString = pre.process(fileAsString);
            //Save long String as a temporary text file in
C:/workspace/temp.txt
            System.out.println("Saving to temporary file...");

            /*
             * Find project path
             */
            String projectPath = System.getProperty("user.dir");

            try{
                  FileWriter outStream = new FileWriter(projectPath +
"\\resources\\DOM.txt");
                  outStream.write(fileAsString);
                  outStream.close();
            }
            catch(IOException e) {
                  System.out.println("Output Error");
            }

            //Parse temporary file and build DOM.
            System.out.println("Building Document Object Model...");


            DomBuilder builder = new DomBuilder();
```

```
//          DOMDataStructure dom =
builder.build("C:/Users/Tristan/Desktop/ExtractionProject/temp.txt");

            DOMDataStructure dom = builder.build(projectPath +
"\\resources\\DOM.txt");

            //Apply extraction methods
            System.out.println("Applying Extraction Methods...");
//          String synLib =
"C:/Users/Tristan/Desktop/ExtractionProject/Libraries/sysnonyms2.xml";
//          String exactSynLib =
"C:/Users/Tristan/Desktop/ExtractionProject/Libraries/exactMatches2.xml
";
//          String keywordLearningFile =
"C:/Users/Tristan/Desktop/ExtractionProject/Libraries/proposal.xml";
//          String synLib = "C:/code/Libraries/sysnonyms2.xml";
//          String exactSynLib = "C:/code/Libraries/exactMatches2.xml";
//          String keywordLearningFile =
"C:/code/Libraries/proposal.xml";
            String synLib = projectPath + "\\resources\\synonyms.xml";
            String exactSynLib = projectPath +
"\\resources\\exactMatches.xml";
//          String keywordLearningFile = "/resources/proposal.xml";
            String outputFile = outdir + "\\" + getFileName(loc)+
".xml";

            ExtractionMethods extractor = new
ExtractionMethods(templateObject);
            Vector matchList = new Vector();

            System.out.println("Applying Synonym Extraction ...");
            Vector synMatches = extractor.synonymExtraction(dom,
synLib);
            matchList.add(synMatches);



public void printToXML(Vector matchList, String outputFile, String
template){
            try {
                  FileOutputStream outStream = new
FileOutputStream(outputFile);
                  PrintWriter pWriter = new PrintWriter(outStream);
                  BufferedReader templateStream = new
BufferedReader(new FileReader(template));
                  String line = templateStream.readLine();
                  pWriter.println("<?xml version=\"1.0\"
encoding=\"ISO-8859-1\"?>");
                  pWriter.flush();
                  while(line!=null){
                        if(line.contains("><")){
```

```
                              String tag =
line.substring(line.indexOf("<")+1, line.indexOf(">"));

                              for(int i =0; i<matchList.size(); i++){
                                    Vector currentList =
(Vector)matchList.elementAt(i);
                                    for(int j=0; j<currentList.size();
j++){
                                          Match currentMatch =
(Match)currentList.elementAt(j);


      if(currentMatch.getName().equals(tag)){
                                                Vector data =
currentMatch.getData();
                                                for(int k = 0;
k<data.size(); k++){
                                                      String
currentData = (String)data.elementAt(k);


      currentData=currentData.replace("$", "");
                                                      String toPrint =
line.replaceAll("><", ">"+currentData+"<");

      pWriter.println(toPrint);
                                                      pWriter.flush();

                                                }
                                          }
                                    }
                              }
                        }
                        else{
                              pWriter.println(line);
                              pWriter.flush();
                        }
                        line=templateStream.readLine();
                  }
            }
            catch(FileNotFoundException e) {
                  e.printStackTrace();
                  System.exit(0);
            }
            catch(IOException e) {
                  e.printStackTrace();
                  System.exit(0);
            }
      }
```

ExtractionMethods.java

```
public Vector summaryExtraction(DOMDataStructure dom){

            Vector matches = new Vector();

            Match match = new Match ("summary");
```

```
            HTMLTagNode root = dom.getRoot();
            HTMLIteratorInterface iter = new
HTMLPreorderIterator(root);

            iter.first();
            while (!iter.isDone()) {
                  AbstractHTMLNode tmpNode = iter.currentItem();
                  String tmpText = tmpNode.getContent();
                  //System.out.println("tmpText: " + tmpText);

                if(tmpText.contains(summary_tag)){

                  // An simple algorithm for getting summary content
for a specific pattern
                  String summaryOriginal =
iter.currentItem().getContent();
                  //System.out.println("[summaryOriginal] = " +
summaryOriginal);


                  if(summaryOriginal.contains(patternStart)){

                        String summaryContent =
extractBetween(summaryOriginal, patternStart, patternEnd);
                        //System.out.println("[summaryContent] = " +
summaryContent);

                        //tmpNode = iter.currentItem();
                        //System.out.println("tmpNode.getContent(): " +
tmpNode.getContent());

                        match.addData(summaryContent);
                        matches.add(match);
                        return matches;
                  }
                  else{
                        iter.next();
                              iter.next();
                              iter.next();
                              if
(iter.currentItem().getContent().contains("td width=")) iter.next();
                              tmpNode = iter.currentItem();
                              match.addData(tmpNode.getContent());
                              matches.add(match);
                              return matches;
                  }

                  }
                  iter.next();
            }
            //System.out.println("Summary Match: " +
match.getData().toString());
            return matches;
      }
```

STRICT_logic.java

```
      public XMLstrings runExtraction(String URLIN, String templateIn,
String outdirIn, String htmlTemplateIn) {

            String URL = URLIN;
            String template = templateIn;
            String outdir = outdirIn;
            String htmlTemplate = htmlTemplateIn;


            HTML_Template template1 = new HTML_Template(0, null, null,
null, null, null, null, null, null, null, null, null);
            System.out.print("curent values");
            System.out.print(template1);
            try {
                  //StaxReader is created, passed a html template and
the results are stored in a list
                  StaxReader reader1 = new StaxReader(htmlTemplate);
                  List<HTML_Template> book = reader1.readTemp();
                  //the template information from the first item in the
list is recored in template1
                  template1=book.get(0);

            // exceptions
            } catch (FileNotFoundException ex) {
                        System.out.print("FNF");
            } catch (XMLStreamException e) {
                        System.out.print("STREAM");
            }catch (NullPointerException ej) {
                        System.out.print("NULL");
            }

            //Extraction logic is created, (passing the template), and
xml document is created
            ExtractionLogic extr = new ExtractionLogic(template1);
            String XMLDoc = extr.extract(URL, template, outdir);

            XMLstrings proposal1= new
XMLstrings(null,null,null,null,null,null,null);
            try {
                  //StaxReader2 is created, passed a html template and
the results are stored in a list
                  StaxReader2 reader2 = new StaxReader2(XMLDoc);
                  List<XMLstrings> book = reader2.readTemp();
                  //the template information from the first item in the
list is recored in proposal1
                  proposal1=book.get(0);

            // exceptions
            } catch (FileNotFoundException ex) {
                        System.out.print("FNF");
            } catch (XMLStreamException e) {
                        System.out.print("STREAM");
            }catch (NullPointerException ej) {
                        System.out.print("NULL");
            }
```

```
            return proposal1;
      }

      public double runComparisonSummary(XMLstrings proposal1In,
XMLstrings proposal2In, String domainPathIn){
            XMLstrings prop1=proposal1In;
            XMLstrings prop2=proposal2In;
            String domainPath = domainPathIn;

            String prop1Summary = prop1.getSummary();
            String prop2Summary = prop2.getSummary();

            InformationComparator compare = new
InformationComparator(domainPath);
            double summarySim = compare.getSimilarity(prop1Summary,
prop2Summary);

            return summarySim;
      }
```

InformationComparator.java

```
      public double getSimilarity(String text1, String text2) {

            double similarity = 0.0;

            try {
                  Map<String, Integer> wordList1 = this
                              .displayTokenUsingStopAnalyzer(text1);

                  Map<String, Integer> wordList2 = this
                              .displayTokenUsingStopAnalyzer(text2);

                  wordList1 = filter(wordList1);
                  wordList2 = filter(wordList2);

                  IDictionary dictionary = this.conWNet();

                  Map<String, Integer> hypernymInfo1 =
this.getHypernymList(
                              dictionary, wordList1);

                  Map<String, Integer> hypernymInfo2 =
this.getHypernymList(
                              dictionary, wordList2);

                  similarity = this
                              .getSimilarityFromMap(hypernymInfo1,
hypernymInfo2);

            } catch (IOException e) {
                  e.printStackTrace();
            }

            return similarity;

      }
```

Curriculum Vitae

Omkar Gurav

766 Windsor Street • Fredericton, NB E3B 4G5 • omgurav@gmail.com • +1 506-261-4822

Skills

Java Technologies	Web Technologies	Front-end technologies	Database Technologies	Others
JSP, Servlet	PHP	AngularJS	MySQL	Apache Cordova
Spring Framework	CodeIgniter	JQuery	Oracle	R
Hibernate	Ruby on Rails	Bootstrap Framework	PL/SQL	Matlab
JAX-RS		Ionic Framework	MongoDB	
BIRT		HTML, CSS		

Education

- Masters in Computer Science - University of New Brunswick, Fredericton - April 2015
- Bachelor of Engineering in Computer Engineering - University of Mumbai, Mumbai - May 2011

Experience

Graduate Research Assistant. (MCS Report) May' 2014 - Present
University of New Brunswick, Fredericton, Canada.
The research project is about creating a Proposal matching tool. The project is implemented using Java programming language and packages used are WordNet, JWI, and Apache Lucene. The technique used to compare proposal is 'tf-idf'.

Technical and Web Assistant Feb' 2015 – Apr' 2015
University of New Brunswick, Fredericton, Canada.
Solely responsible for the complete project. The project is creating a website which contains the course related material. Service orientated architecture is implemented with REST using Ruby on Rails. The project will be deployed on OpenShift public cloud. Frontend is developed using AngularJS. The project also involves developing hybrid app using Apache Cordova/PhoneGap.

Software Developer Co-Op May' 2014 - Dec' 2014
Siemens Canada, Fredericton, Canada.
Worked with Smart Grid project using agile development technique with Spring Framework, Hibernate and other Java EE technology. Contributed in REST service implementation, creation of report using Business Intelligence Reporting Tool (BIRT), unit testing using JUnit Framework. Collaborated with US team for bug solving and functionality enhancement of system.

Web Designer/Developer Feb' 2014 - April' 2014

University of New Brunswick, Fredericton, Canada.

Solely handled the project from requirement gathering, designing and implementation for a research lab at University of New Brunswick. The project involved content management system for easy web site editing. Extensive learning and implementation using Ruby on Rails which in turn lead to rapid web development.

Teaching Assistant Sept' 2013 - Dec' 2013

University of New Brunswick, Fredericton, Canada.

Organize laboratories for undergraduate students, assisted professors in lab delivery, students doubt clarification, markings and student database management

Software Developer Sept' 2012 - Sept' 2013

FirstEconomy Pvt. Ltd., Mumbai, India.

Social media management web project was independently handled by me from research to implementation. Multiple social media APIs like Facebook, Twitter and Google Analytics were used. Frontend technology used are JavaScript and JQuery and PHP in CodeIgniter Framework at server side. Designed and developed the database schema in MySQL and I was also responsible for writing SQL queries for it. Data analysis and visualized the data using Google Graph API.

Indiaordes.com, a ecommerce project in which I worked in the team for functionality enhancements and troubleshooting using PHP in CodeIgniter Framework, front end development and writing SQL queries. Implemented Payment and SMS Gateway and used PHPMailer to send emails. Designed and developed control panel for shops and admin for product management, delivery management and account management.

Certification

- Completed Oracle Certified JAVA Professional, JAVA SE 6 Programmer course.
- Completed Oracle Certified Database SQL Expert, Oracle 10g Programmer course.
- Participated in workshop on Ethical Hacking conducted by IEEE.
- Participated in Academic Developer Conference conducted by IEEE & Microsoft Student Partner.

Co-curricular and Extra Curricular Activities

- Performed system integration for LAN connectivity and Software installation for successfully organizing a gaming competition.
- Planning and organizing industrial visit to the Software Paradigm InfoTech Pvt. Ltd.

Volunteering Experience

- Volunteered in several extracurricular events in Computer Engineering student association at S. S. Jondhale Engineering College.
- Volunteered and arranged blood donation campaigns at university.
- Volunteered at several Indian student association and Indo-Canadian association events.
- Fund raising for HelpAge India and Indian Cancer Society.