Evaluating Term Extraction Methods for Domain Analysis

Chaitanya Nemallapudi

Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science

In

Computer Science

Dr. William B. Frakes, Chair
Dr. Gregory Kulczycki, Co-Chair
Dr. Ing-Ray Chen, Member

August 2, 2010
Falls Church, VA

Keywords: DARE, Domain engineering, Overlap, Vocabulary, Term extraction
Evaluating Term Extraction Methods for Domain Analysis

Chaitanya Nemallapudi

Abstract

This study compared the vocabularies created by various domain experts and the source documents selected by them to create the vocabulary. The results indicate that there is similarity among the vocabularies created and the source documents selected. Also, the relationship between the overlap scores of vocabularies created and overlap scores of source documents selected was tested and it was observed that no significant relation exists between them. In addition, the variability of the overlap scores of the vocabularies generated automatically to the variability of the overlap scores of those produced manually by domain experts was evaluated. The results suggested that these vocabularies are significantly different from each other.
# Table of Contents

1. Introduction: ........................................................................................................... 1  
   Experiment 1........................................................................................................ 3  
   Experiment 2........................................................................................................ 4  
   Experiment 3........................................................................................................ 4  

2. Problem Background ............................................................................................. 5  
   2.1. Software reuse and domain engineering ......................................................... 5  
   2.2. Terminology Extraction .................................................................................... 7  
   2.3. Conflation Algorithms ..................................................................................... 9  

3. Method .................................................................................................................... 11  
   3.1. Data Collection .................................................................................................. 11  
      3.1.1. Experiment 1 .............................................................................................. 11  
      3.1.2. Experiment 2 .............................................................................................. 15  
      3.1.3. Experiment 3 .............................................................................................. 16  
   3.2. Data Preparation ............................................................................................... 17  
      3.2.1. Experiment 1 .............................................................................................. 17  
      3.2.2. Experiment 2 .............................................................................................. 18  
   3.3. Experiments ...................................................................................................... 18  
      3.3.1. Experiment 1 .............................................................................................. 18  
      3.3.2. Experiment 2 .............................................................................................. 18  
      3.3.3. Experiment 3 .............................................................................................. 19  

4. Results .................................................................................................................... 21  
   4.1. Experiment 1 .................................................................................................... 21  
      4.1.1. Facet terms without Phrases ........................................................................ 21  
      4.1.2. Facet terms with phrases .......................................................................... 22  
      4.1.3. Facet groups without phrases .................................................................... 23  
      4.1.4. Facet groups with phrases ........................................................................ 24  
   4.2. Experiment 2 .................................................................................................... 25  
   4.3. Experiment 3 .................................................................................................... 30
5. Conclusions ........................................................................................................................................... 34

5.1. Future work ........................................................................................................................................ 34

Bibliography ............................................................................................................................................. 35

Appendix ..................................................................................................................................................... 37

Table of Figures

Figure 1: Equation for Overlap .................................................................................................................. 3
Figure 2: [Fair use] Metric Hierarchical Ordering [12] ............................................................................ 9
Figure 3: Frequency of Vocabulary sizes ................................................................................................ 12
Figure 4: Comparison of overlap score means of Manual-Manual ............................................................ 20
Figure 5: Comparison of facet terms without phrases ............................................................................. 21
Figure 6: Comparison of facet terms with phrases ................................................................................... 22
Figure 7: Comparison of facet groups without phrases ........................................................................... 23
Figure 8: Comparison of facet groups with phrases ................................................................................ 24
Figure 9: Information of source documents selected by domain experts .................................................. 26
Figure 10: Source Documents by each domain expert ............................................................................. 26
Figure 11: Information of Conflation algorithms chosen by the domain experts ..................................... 27
Figure 12: Frequency of Source Documents selected by the domain experts (Pareto Plot) ..................... 27
Figure 13: Comparison of Sources ........................................................................................................ 28
Figure 14: Scatter plot Matrix for Domain sources, Facet terms without phrases, Facet terms with phrases, Facet groups without phrases, Facet groups with phrases ......................................................... 29
Figure 15: Two-sample T-test to compare means of Manual-Manual and Automatic-Manual (without stemming and stop words) facet terms with phrases overlap scores ......................................................... 30
Figure 16: Two-sample T-test to compare means of Manual-Manual and Automatic-Manual (with stemming and stop words) facet terms with phrases overlap scores ......................................................... 31
Figure 17: Two-sample T-test to compare means of Manual-Manual and Automatic-Manual (with stemming and no stop words) facet terms with phrases overlap scores ......................................................... 32
Figure 18: Two-sample T-test to compare means of Manual-Manual and Automatic-Manual (using stop words and without stemming) facet terms with phrases overlap scores. .......................... 33
Tables

Table 1: Metric Summary and Abbreviations [12] ................................................................. 8
Table 2: Facet Table – An example ...................................................................................... 17
1. Introduction:

In software engineering, developers and researchers have been using tools and methods to reduce costs, time to market, and improve the quality of products. Software reuse and reusable components have been helping achieve these goals by reducing redundancy in processes. In this way, costs may be reduced and companies may be able to meet ever shortening product deadlines. Significant research and progress has been done in the field of software reuse over the past two decades [1], and despite the progress, many open research questions exist in this area such as such as software reliability and scalability i.e. whether reuse and domain engineering methods can be applied to large systems.[2]. This paper looks at the problem of term extraction methods for domain analysis.

Domain engineering is one of the ways to achieve software reuse. The process of domain engineering is divided into two phases: Domain Analysis and Domain Implementation [3]. In the domain analysis phase, a domain expert analyzes various sub-systems of a domain to find similarities and variabilities of the systems in the domain to understand what kind of reusable components can be created for future use. In the domain implementation phase, the information gathered in the domain analysis phase is used to create reusable components. Several approaches for creating reusable components have been found in the literature [2]. Some of these are Family Oriented Abstraction, Specification and Translation (FAST) [4], Feature-Oriented Domain Analysis (FODA) [5], Organization Domain Modeling(ODM) [6], Feature-Oriented Reuse Method (FORM) [7], Domain Specific Software Architectures (DSSA)[8].

DARE [3] is a method and tool that assists domain experts in performing domain analysis. One of the major goals of DARE research was to achieve automation of domain analysis activities. DARE helps the domain experts in capturing domain information from documents related to a particular domain. This domain information may be in the form of architectural diagrams, feature tables, facet tables and vocabulary. This information is recorded in a domain book, the final output of DARE.

Vocabulary is an important sub-section in the domain book. The vocabulary can be extracted from many sources like source code, subject matter experts and domain documents. A subset of documents that are related to the domain to be analyzed are chosen as the domain documents for analysis. These documents can be artifacts of the existing systems like design documents, requirement documents, user manuals etc. Vocabulary can be extracted either manually or automatically. In manual extraction, domain experts process the domain documents and select the terms that best represent the domain. Many information retrieval techniques can help extract the terms from domain corpuses.

Researchers are now focusing on developing and applying information retrieval techniques to automatically extract syntactic and semantic knowledge from domain corpuses [9]. Manually
extracting domain vocabulary is an expensive, time-consuming process. It requires domain experts to analyze large amount of domain information and extract domain-specific keywords from it. Automatic extractors, on the other hand, may not be the best methods to rely upon when quality is a concern. There has been insufficient empirical study to prove which method is more consistent. Hence, the need to address the problem of choosing a consistent method for domain-specific term extraction.

Automatic extraction methods are gaining more significance and there have been major advances in information extraction research over the past decade [10]. Studies show that manually extracted vocabulary differs from automated extraction because of human knowledge and experience when deciding whether a term is a good candidate to represent a particular domain. Even though, manually extracted vocabulary is more reliable, it is time-consuming. Automatically extracted vocabulary contains many redundant and meaningless terms it can be created at lower cost and time [11]. There has been much debate among researchers on replacing manual method with automatic extraction of index terms. This paper is an attempt to evaluate some aspects of manual and automatic extraction methods for domain analysis. This paper presents 3 experiments that were conducted on the vocabularies created by both manual and automatic extraction methods as a part of domain analysis using DARE.

The three experiments put in general terms:

Experiment 1: Domain experts working on the same domain (Conflation) have created vocabularies that best represent the domain. In this experiment, the vocabularies created by different domain experts were compared to determine if there is consistency i.e. (common terms) among the vocabularies. Our hypothesis, put informally, is that there will be similarities between the vocabularies created by the domain experts.

Experiment 2: Domain experts select a set of source documents to create the vocabulary. The source documents selected by the domain experts were compared to determine if there is consistency among the sources. Our hypothesis, put informally, is that there will be similarities between the sources selected by the domain experts to create the vocabulary.

Experiment 3: In Tilley’s study [12], the vocabularies created by the domain experts were compared with those generated by the automatic generators. Our study compared the vocabularies of the domain experts. These two data sets have been compared to determine if there is more consistency between the domain experts or between the domain experts and automatic generators.

“Symmetrical Overlap” [14] is the measure used to compare the vocabularies created by the domain experts and sources selected by them to create the vocabulary. In this study, “Symmetrical overlap” is referred to “Overlap.” The reason behind this choice is that overlap necessitates normalization. Since large vocabularies have relatively higher number of terms. This
increases the probability of number of matches between them and hence leads to higher similarity. Therefore, by normalizing the vocabulary sizes, a score between 0 and 1 is obtained which is scaled based on the size of the vocabularies. Overlap measure deals fairly with the vocabularies of all sizes.

Consider two sets A and B; “Overlap” is defined as “the number of terms in the intersection of the two representations (A & B) divided by the number retrieved by the union on those representations”[14].

The equation of the overlap is:

\[
O = \frac{|A \cap B|}{|A \cup B|}
\]

Figure 1: Equation for Overlap

As an example, assume that cardinalities of A and B are 15 and 20 respectively, and if 5 elements are common to both the sets, the intersection of the two sets would have a size of 5 elements and the cardinality of union would be 35, then the overlap of the sets would be 5/35 or 0.14.

The highest possible score is 1, which indicates that all the elements in the two sets match and the lowest possible score is 0 which indicates there is no match between the two sets.

We formalized a null hypothesis (H₀) and an alternate hypothesis (Hₐ) for our experiments as follows:

**Experiment 1**

**Alternate Hypothesis (Hₐ):** The overlap scores between vocabularies of different domain experts will be greater than 0.

**Null hypothesis (H₀):** The overlap scores between vocabularies of different domain experts will be equal to 0.
Experiment 2

Alternate Hypothesis (Hₐ): The overlap scores between the source documents used by the domain experts for analysis will be greater than 0.

Null Hypothesis (H₀): The overlap scores between the source documents of different domain experts will be equal to 0.

Experiment 3

Alternate Hypothesis (Hₐ): The overlap scores between different domain experts will be greater than that of overlap scores between domain experts and automatic extraction metrics. (There will be significant difference between the means of manual-manual and automatic-manual overlap scores.)

Null Hypothesis (H₀): The overlap scores between different domain experts will not be greater than that of overlap scores between domain experts and automatic extraction metrics. (There will be no significant difference between the means of manual-manual and automatic-manual overlap scores).
2. Problem Background

The topics of this paper come from three distinct fields. Section 2.1 briefly discusses the background of software reuse. Section 2.2 talks about automatic term extraction. Section 2.3 gives a brief introduction on the conflation domain.

2.1. Software reuse and domain engineering

As need for business applications grow rapidly, software developers are faced with a challenge of creating more complex and powerful software systems [15]. This puts pressure on the developers to create products that are reliable, inexpensive and have very short product deadline. These rapid changes in the software engineering field made researchers think of new ways to meet the ever-shortening product life cycles without increasing cost and labor. Increased productivity can be achieved in many ways such as object-oriented programming, component-based development and domain engineering and these are also some methods to achieve software reuse. [15].

Software reuse can be defined as the ability to use the existing programming code, software artifacts or knowledge in another application to improve quality and productivity[16]. The idea behind this is to avoid spending resources, time and money to build new software components or systems for which solutions already exist. Another significant benefit with software reuse is reduced risk, since it is better to reuse already existing software than to create something new with the risk of hidden problems that all new software has [17]. Domain engineering is the central concept in systematic software reuse. Most companies today, are ready to spend time and effort to identify and design reusable components for their future use rather than on building components from scratch. This helps them to achieve systematic reuse by reducing cost, time, and risk and to improve quality and productivity. Domain experts use methods and tools to understand the similarities and variabilities across the system and try to identify the reusable components for future use.

The idea of software reuse is not new[1]. It can be traced back to its origins with a proposal paper [18], suggesting software experts and developers to focus on creating components as standard catalogue of routines that can be applied to a larger class. In the late 1970’s a software reuse project named Draco introduced the ideas of components, domain languages, and domain analysis [19]. Since the publication of these papers, researchers have been focusing on various active areas like reuse libraries, reuse design, domain engineering methods and tools [2].
Software reuse has been practiced by developers either formally or informally since it offers many benefits like improvement in productivity and quality [20]. Studies have been reported that software reuse can reduce software product development time and costs [21]. In addition to these benefits, reliability is another objective of software reuse [2]. Many metrics and models exist to help software organizations that implement systematic reuse to measure their progress and identify the effective reuse strategies[16]. Software quality and productivity can be improved by shifting the focus of software engineering to a domain centered view [3]. Domain analysis facilitates reuse. Domain analysis includes identifying, capturing, organizing information that was used in developing old systems to define domain models for supporting software reuse within the new systems [22].

Domain Analysis and Reuse Environment [3] is a method and case tool that helps domain experts to record information in a domain. One of the significant findings of DARE research is that domain analysis process can be partially automated. Domain analysis in DARE uses domain documents, code and expert knowledge to extract information. This information from the domain analysis process is used to build domain models such as facet tables, feature table and generic architectures. The initial step in the domain analysis process is defining the scope of a domain. Then source documents are selected. The source documents are analyzed and information is extracted, organized and recorded into different sections of a domain book. Vocabulary analysis follows the domain scoping. The textual domain sources are analyzed using lexers and stop words to create the vocabulary of the domain. From this vocabulary, a domain keyword set is created using manual or automatic extraction methods. These keywords are then grouped into clusters based on similarities. Each cluster is given a facet name and is identified as a facet category in the facet table. Finally, a domain template describing the domain is created by using facet categories. The facet tables and domain templates are used to identify reusable components. In the architectural analysis, the system architecture diagrams of each exemplar system are merged into a generic system architecture that can express commonalities as well as variabilities of all systems in the domain [3].

DARE has been implemented in several prototypes. The first implementation of DARE was used to develop domain book metaphor, and to investigate machine-assisted graphical word and phrase extraction and clustering. In the second implementation, system architecture, facet table and several major subsystems were implemented. DARE-COTS is the third implementation which relies on commercial-off-the-shelf tools and freeware to create the domain book. DARE-COTS use various tools for text processing and code analysis. A fourth implementation of DARE is being created with a web user interface [3].
2.2. Terminology Extraction

Information extraction is a process of automatically extracting relevant terms from domain corpuses. The generation of domain specific vocabulary is a task of increased interest since it is the first important step in the process of domain engineering and information retrieval. Unfortunately, manually generating domain-specific terms is an expensive approach since it requires the intervention of human knowledge, i.e. subject knowledge experts. Another drawback with this approach is it is error-prone since it does not adapt to rapidly changing needs as new disciplines emerge quickly and others disappear at the same pace.[23] Thus, a definitive need of faster and cheaper methods for term extraction exists and the solution is automatic term extraction methods. Several methods exist to automatically extract terms from domain specific documents.

There has been a lot of research on evaluating different ways of selecting domain-specific terminology in regards to both statistical and linguistic properties of the text. Studies suggest high frequency in a corpus is a characteristic of both terminological as well as non-terminological expressions [24], and by adding linguistic knowledge to the representation rather than relying only on statistics such as term frequency yield better results in term extraction [25], [26]. Many works found in the literature has emphasized the importance of noun phrase selection. One study proposes a semi-automatic tool for technical term extraction that groups noun phrases by their head and the terms are ordered by decreasing frequency of the head word in the document. The highly frequent head words are used in technical term list and the less frequent head words are considered to be unlikely candidates for that particular domain. [27]. Tools are available to discover noun phrases from a given text. Lexter and NPTool are some such tools. A detailed description and working of the tools can be found in [28] and [29]. Some studies suggest that pure frequency as a measure of terminology extraction is a good metric [30] However, frequency of the term alone cannot be considered as a good indicator. More frequent terms may be good candidates from a grammatical point of view but they are totally irrelevant as terminological expressions [31].

In indexing, words that occur frequently within a document are likely to be good descriptors of that document. Similarly, domain-specific terms can be identified by comparing a term’s relative frequency in a given domain corpus to its relative frequency in a large, well distributed corpus. Hence terminology extraction and automatic indexing are relatively similar tasks and same set of techniques and methods can be applied. However, in terminology extraction, the less frequent terms that are not representative of the domain but statistically significant can be considered as good candidates; in indexing, a certain frequency threshold may be needed in order extract the terms that best describe the text. [26]. Many text mining tools are available to construct ontologies and also identify terms that are good descriptors for the concepts in the ontology. A study on “Comparison of statistical methods for automatic term extraction has evaluated various information retrieval metrics to evaluate their effectiveness in
domain-specific term extraction.” This study concluded that term frequency is one of the most important factors in term extraction [12].

Table 1 [12] provides a short description of metrics and the weighting rationale of each term. Figure 2 [12] provides a taxonomical view of the metrics in the experiment.

Table 1: Metric Summary and Abbreviations [12]

<table>
<thead>
<tr>
<th>Metric title</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corpus Term Frequency</td>
<td>Rewards high term count, large documents have advantage</td>
</tr>
<tr>
<td>Document Term Frequency</td>
<td>Rewards words that appear a lot in one document</td>
</tr>
<tr>
<td>Evenly Distributed</td>
<td>All documents contribute same number of terms</td>
</tr>
<tr>
<td>Favor Big Documents</td>
<td>Reward for large documents</td>
</tr>
<tr>
<td>Normalized Term Frequency</td>
<td>Rewards high term count, but negates large document skewing</td>
</tr>
<tr>
<td>Document Relativized</td>
<td>Less reward for large documents, penalizes verbose documents</td>
</tr>
<tr>
<td>corpus relativized</td>
<td>Less reward for large documents</td>
</tr>
<tr>
<td>Document Relativized - Document Average Frequency</td>
<td>Less reward for large documents</td>
</tr>
<tr>
<td>Corpus Relativized - Document Average Frequency</td>
<td>Less reward for large documents</td>
</tr>
<tr>
<td>Term Frequency and Inverse Document Frequency</td>
<td>Reward terms that are in few documents, but that appear frequently</td>
</tr>
<tr>
<td>Term Frequency and Logged Inverse Document Frequency</td>
<td>Flattens distribution of document frequency, making outliers less powerful</td>
</tr>
<tr>
<td>Distribution consensus</td>
<td>Rewards terms that occur in same frequency in multiple documents</td>
</tr>
<tr>
<td>Binary Consensus</td>
<td>Rewards consensus, rewards minimum frequency of one</td>
</tr>
</tbody>
</table>
Figure 2: [Fair use] Metric Hierarchical Ordering [12]

2.3. Conflation Algorithms

As defined in the literature [32], conflation is a process of grouping non-identical words which refer to the same principal concept. In the context of information retrieval, conflation is useful in efficient indexing, query expansion and faster retrieval. The conflation process can be done either manually or automatically. The automatic process of conflation is called stemming. Stemming algorithms can be divided into 4 types. They are affix removal, n-gram, successor variety and table lookup [33]. Affix removal algorithms remove suffixes or prefixes from terms and reduce them to a stem. The first affix removal stemming algorithm was developed by Lovins [34]. Successor variety uses the frequencies of letter sequences in a word corpus to find stems of terms. Hafer and Weiss [35] proposed a word segmentation algorithm in successor variety that uses statistical properties of a corpus to automatically segment words into their stems and affixes. The n-gram method conflates terms based on the ratio of common letter sequences.
called n-grams [36]. In Table lookup method, the terms and their associated stems are indexed in a table.

Several stemming algorithms have been described in the literature Lovin[34] and Porter [37]. The Lovins stemmer removes the suffix of the longest-match, where as Porter algorithm removes from suffixes from a pre-defined set [38]. The Porter stemmer has several drawbacks. 1) It is difficult to understand, modify and it produces stems that are not words [39]. In Information retrieval systems, the performance of stemming methods is evaluated in terms of their effect on retrieval performance and the evaluations of stemming methods have produced mixed results. Some studies proved that using general purpose stemming algorithms did not result in improvements in retrieval performance [40]; it is also evident from the studies that consistent but rather small improvements can be seen in retrieval effectiveness [38]. This study has also proved that retrieval performance can be improved by indexing documents by synonym of the words rather than the word itself.

A detailed evaluation on the Performance of five different stemming algorithms (S-stemmer, Lovins stemmer, Porter stemmer, Xerox inflectional stemmer, and Xerox derivational stemmer) was conducted by Hull [41] and the conclusions of this work are: 1) stemming always improve retrieval performance 2) Rule based suffix removal may not always be an ideal approach for stemming. 3) prefix removal may be an unfavorable feature in stemming algorithms [41]. A study on using domain analysis for the conflation algorithms compared the performance of automatically generated stemmers with those created manually. The results are compared in terms of identical stem generation, development times, LOC and total time spent to generate stems. The study concluded that the automatically generated stemmers produced more than 99.9% identical stems with the manually developed stemmers. The stemmers produced by application generators have bigger executables than the stemmers developed by humans. No statistical difference found between the generated and developed stemmers in terms of LOC and the total time spent to stem all terms [36].
3. Method

3.1. Data Collection

To test the above hypotheses, data from domain engineering course projects at Virginia Tech were used. DARE methodology was used for domain analysis and domain books were created by each student as a result of these exercises. Each student collected documents that are related to a domain. A minimum of three documents were selected with no restriction on the maximum number of documents. These documents selected included research papers, system documentation, books, web pages or code. The task was to process domain documents and select the terms that best represent the domain. The students were allowed to use tools that automatically provide index terms for the domain corpus and then based on their domain knowledge, manually selected the terms that best represent the domain. The selected terms or vocabulary was then used to create other artifacts of the domain book like facet table, feature table, system architecture. The vocabularies created by the domain experts and the source documents used for analysis were used as our test data. The vocabularies created manually by the students were considered as our domain expert created vocabularies. For this study, we have chosen to use the projects from the “Conflation” domain.

3.1.1. Experiment 1

In the DARE methodology, the vocabulary terms are created as a result of an important step known as vocabulary analysis. In this process, domain experts use domain documents and expert knowledge to create an initial word set. From this initial word list, a domain keyword list is determined by manually analyzing the initial word set. These keywords are then grouped into clusters according to commonalities among them. Using these word clusters, a facet table is created for the domain. Each column in the facet table will become a facet group and is identified by a facet name. In the first experiment, we identified the facet names as facet groups and the words that fall into the category of a facet group as facet terms. The facet groups and facet terms created by the domain experts during vocabulary analysis are the source of input for this experiment. The facet term vocabulary sets and facet group vocabulary sets created by twenty nine domain experts are used in this experiment.
Figure 3: Frequency of Vocabulary sizes.

The histogram in figure 4 shows the frequency of the vocabularies created by the domain experts with size of vocabularies and count plotted on X-axis and Y-axis respectively. The mean vocabulary size was 39.17, and the median was 38.00. In terms of variability, the highest vocabulary size was 95 and the lowest vocabulary size was 15 yielding a range of 80 and a standard deviation of 20.25.
Figure 4: Scatter plot matrix for domain sources selected and their corresponding vocabulary sizes.

Correlation coefficient was computed to assess the relationship between the count of source documents selected for the domain analysis and the size of the vocabulary created. A scatter plot shows a positive correlation between the number of sources documents selected for domain analysis and their corresponding domain vocabularies $r=0.56$, $p=0.05$. 

The histogram in figure 5 shows the frequency of the facet groups created by the domain experts with number of facet groups and count plotted on X-axis and Y-axis respectively. The mean number of facet group was 6.17, and the median was 6.00. In terms of variability, the highest number of facet groups created by the domain experts was 10 and the lowest number of facet groups was 3 yielding a range of 7 and a standard deviation of 1.81.
The above histogram shows the frequency of the facet terms created by the domain experts with number of facet terms and count plotted on X-axis and Y-axis respectively. The mean number of facet terms was 34.41, and the median was 31.00. In terms of variability, the highest number of facet terms created by the domain experts was 89 and the lowest number of facet terms was 11 yielding a range of 78 and a standard deviation of 19.81.

3.1.2. Experiment 2

The Initial step in the DARE methodology is to collect the source information. These are the documents related to the domain that has to be analyzed. The source documents such as source code, system descriptions, system architectures, system feature tables are the source of input for this experiment. The source documents from twenty nine different domain experts were used for this experiment. Figure 7 displays the frequency of corpus sizes or source documents used to select the vocabulary.
The histogram shows the frequency of sources selected by the domain experts with number of source documents selected and count plotted on X-axis and Y-axis respectively. The mean number of sources selected was 12.31, and the median was 13.00. In terms of variability, the highest number of sources selected was 46 and the lowest number of sources selected was 5 yielding a range of 41 and a standard deviation of 9.24.

3.1.3. Experiment 3

The data for this experiment was obtained from the results of Tilley’s study [12] in which “various information retrieval and filtering metrics were tested and evaluated to determine their effectiveness in identifying domain vocabulary”. The vocabulary extracted using these metrics were compared with the expert or manually selected vocabulary and overlap scores were computed. In our study, these overlap scores are referred to Automated-manual overlap scores and use them to compare with the overlap scores measured between various domain experts in the first experiment.
3.2. Data Preparation

3.2.1. Experiment 1

A Java program was written that used vocabulary sets of various domain experts to calculate the overlap scores. The vocabularies are obtained from the facet tables created by the domain experts. As mentioned earlier, we identify the facet names as facet groups and the words that fall into the category of a facet group as facet terms. We placed the facet group names and facet term names in two distinct text files. Since the vocabularies are manually created by the domain experts, we had to take additional steps for data extraction. The vocabulary sets contained non-alpha numeric characters such as _, -, ( ). These characters have to be removed from the vocabularies since they might create noise. For e.g. the words “n gram” and “n-gram” might be considered as two different words although they represent a single conflation algorithm. This clean-up makes vocabulary sets more comparable. The vocabulary was then passed through a stemmer to create standardized vocabulary. We also found that some vocabulary sets contain phrases rather than single terms. For e.g. in some vocabulary sets, the phrase “domain engineer” was considered as a single term rather than two different words. For this reason, four variations of the vocabulary sets were identified for all the domain experts. They are:

- Facet terms without phrases
- Facet terms with phrases.
- Facet group without phrases
- Facet groups with phrases

Overlap scores were calculated for all the four variations of the vocabulary sets of all domain experts.

For e.g. consider the following facet table for the conflation domain.

**Table 2: Facet Table – An example**

<table>
<thead>
<tr>
<th>Conflation Methods</th>
<th>Performance</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>Frequency</td>
<td>Table lookup</td>
</tr>
<tr>
<td>Manual</td>
<td>Precision</td>
<td>Successor variety</td>
</tr>
<tr>
<td></td>
<td>Effectiveness</td>
<td></td>
</tr>
</tbody>
</table>

*Facet terms without phrases* will be the set: [Automatic, Manual, Compression, Precision, Effectiveness, Frequency, Table, Lookup, Successor, Variety] with 10 members.

*Facet terms with phrases* will be the set: [Automatic, Manual, Compression, Precision, Effectiveness, Frequency, Table lookup, Successor variety] with 8 members.
Facet groups without phrases will be the set: [Conflation, Methods, Performance, Algorithm] with 4 members.
Facet groups with phrases will be the set: [Conflation Methods, Performance, Algorithm] with 3 members.

3.2.2. Experiment 2

A java program was written to compare the titles of the source documents and compute overlap score between source documents selected by the domain experts. The titles of all the source documents were manually processed in order to maintain consistency. For references with no titles, a unique name was chosen and used consistently throughout the experiment.

3.3. Experiments

3.3.1. Experiment 1

Due to the unique backgrounds and experiences of different domain experts, the vocabularies created by different subject matter experts for the same domain may vary. The overlap scores of the vocabularies created by each domain expert were compared to all the other expert’s vocabularies. By vocabularies, both facet groups and facet terms separately. As mentioned before, the overlap score is the cardinality of the intersection of the pair wise vocabularies created by domain experts over the union. It can be measured using the equation:

\[
\text{Overlap (X)} = \frac{|V_{dei} \cap V_{dej}|}{|V_{dei} \cup V_{dej}|} \quad \text{......................... (1)}
\]

\(V_{dei}\) - Vocabulary created by Domain expert \(i\).
\(V_{dej}\) - Vocabulary created by Domain expert \(j\).

3.3.2. Experiment 2

The domain experts have selected an arbitrary number of source documents to create domain vocabularies. The set of source documents that are selected by a domain expert will differ from other expert’s source documents. The overlap score is measured as the cardinality of intersection of source documents that are selected by the experts over the union of the same. It can be measured using the equation:

\[
\text{Overlap(X)} = \frac{|Docs_{dei} \cap Docs_{dej}|}{|Docs_{dei} \cup Docs_{dej}|} \quad \text{......................... (2)}
\]
3.3.3. Experiment 3

In this experiment, the overlap scores between vocabularies generated by the automatic generators and vocabularies created by domain experts have been compared with the overlap score between vocabularies created by the different domain experts. The main purpose this experiment is to compare the Automatic-Manual overlap score with Manual-Manual overlap scores to evaluate the significance of automatic generators in vocabulary generation for domain analysis. Manually created vocabularies of 29 domain experts in this study have been compared with 7 vocabularies created by automatic extractors from Tilley’s study [12].

Information retrieval metrics used in the Tilley’s study [12]

Overlap between vocabularies produced by domain experts and automatic generators.

Grand mean of the overlap score between vocabularies of domain experts and automatic generators.

Figure 8: Comparison of overlap score means of Automatic-Manual
<table>
<thead>
<tr>
<th></th>
<th>$\text{de}_1$</th>
<th>$\text{de}_2$</th>
<th>$\text{de}_{3..}$</th>
<th>$\text{de}_{29}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{de}_1$</td>
<td>$X_{(\text{de}_1, \text{de}_1)}$</td>
<td>$X_{(\text{de}_1, \text{de}_2)}$</td>
<td>$\cdots$</td>
<td>$X_{(\text{de}<em>1, \text{de}</em>{29})}$</td>
</tr>
<tr>
<td>$\text{de}_2$</td>
<td>$X_{(\text{de}_2, \text{de}_1)}$</td>
<td>$X_{(\text{de}_2, \text{de}_2)}$</td>
<td>$\cdots$</td>
<td>$X_{(\text{de}<em>2, \text{de}</em>{29})}$</td>
</tr>
<tr>
<td>$\text{de}_{29}$</td>
<td>$X_{(\text{de}_{29}, \text{de}_1)}$</td>
<td>$X_{(\text{de}_{29}, \text{de}_2)}$</td>
<td>$\cdots$</td>
<td>$X_{(\text{de}<em>{29}, \text{de}</em>{29})}$</td>
</tr>
</tbody>
</table>

$\bar{X}_{(\text{Manual-Manual})}$

- $\text{de}_1, \ldots, \text{de}_{29}$ → Domain experts.
- $X_{(\text{de}_1, \text{de}_1)} \ldots X_{(\text{de}_{29}, \text{de}_{29})}$ → Overlap between vocabularies produced by domain experts.
- $\bar{X}_{(\text{Manual-Manual})}$ → Grand mean of the overlap score between vocabularies of domain experts.

**Figure 4:** Comparison of overlap score means of Manual-Manual
4. Results

In this section, we discuss the resulting data and how it supports or contradicts our hypotheses. Section 4.1 discusses the tests performed on the results of finding overlap between domain experts’ vocabulary (Experiment 1), section 4.2 discusses the tests performed on the results of finding overlap between sources used by domain experts in domain analysis (Experiment 2). Section 4.3 discusses the tests performed on the results comparing automatic term extraction overlap and overlap scores between domain experts (Experiment 3).

4.1. Experiment 1

4.1.1. Facet terms without Phrases

The facet terms (without phrases) of twenty nine different domain expert’s vocabularies were compared, the sample size was 406 data points. The mean score was 0.14, and the median was 0.13. In terms of variability, the highest overlap score was 0.54 and the lowest overlap score was 0 yielding a range of 0.54 and a standard deviation of 0.08. Number of overlap scores that were zero was 13.

**Box plots:**

![Box plots](image)

Two pairs of extreme outliers have been observed in the above box plot domain experts $X_{(15, 22)} = 0.54$ and $X_{(16, 23)}=0.55$. By observing the vocabulary facet terms of these two domain experts, it was found that, both of them used the names of the different conflation algorithms as the vocabulary terms and it was believed that this could be the reason for their high overlap scores. The overlap scores of domain expert 8, is significantly different from others. Looking into the domain book, it was observed that, this domain expert unlike others, worked on conflation of storage systems, databases and directories that have similar data rather than
conflation of words and images and hence the vocabulary or facet terms selected by this domain expert did not match with other’s facet terms. Similarly, the overlap scores of the domain expert 14 are significantly different from others. The possible reason could be that most the of the sources selected by this domain expert are architectural diagrams in the form of flowcharts and more over this domain expert has used some kind of automatic indexing tool to extract the domain vocabulary. The reason behind this belief is that the most common words that are used in the flow charts like Yes, No etc, which does not represent conflation domain were selected as facet terms. Another pair of extreme outliers whose overlap scores are close to 0 is X (26, 29) = 0.03. It was observed that the facet terms selected by domain expert 26 represents the conflation algorithms domain in general where as the facet terms selected by the domain expert 29 represents suffix stripping algorithms, a part of the conflation algorithm domain.

**T-test:**
A one-sample t-test using an alpha level of .05 compared the sample mean of facet terms without phrases created by various domain experts with the mean overlap = 0. The sample mean of 0.14 (SD = 0.09) was found to be statistically different from this value, t (405) = 33.30, “p< .0005”, one-tailed suggesting that the mean overlap score of various domain experts was greater than 0. The 95% confidence interval for the mean was -0.006 to 0.007. The effect size of the sample data was 0.85 and Cohen’s d value was 3.30.

**4.1.2. Facet terms with phrases**
The facet terms with phrases of vocabularies created by twenty nine different domain experts were compared; the sample size is 406 data points. The mean score was 0.07, and the median was 0.07. In terms of variability, the highest overlap score was 0.29 and the lowest overlap score was 0 yielding a range of 0.29 and a standard deviation of 0.05. Number of overlap scores that were zero was 41.

**Box plot:**

![Box plot](Figure 6: Comparison of facet terms with phrases)
In general, it is believed that, facet terms without phrases will yield higher overlap scores when compared with facet terms with phrases. For e.g., The median overlap score for domain expert 14 is between 0 and 0.1 (Figure 6), when phrases are split and are considered as distinct vocabulary terms whereas the median overlap score fell down to 0 when phrases are considered as single vocabulary terms. The vocabularies of domain expert 24 and 25; \( X_{(24, 25)} = 0.22 \), have a good overlap. While observing the domain books of these domain experts, a close match in the source documents of these domain experts was found. The overlap scores that are outliers in the above box plot \( X_{(15, 22)} = 0.51, X_{(16, 23)} = 0.41 \) and \( X_{(13, 18)} = 0.44 \) are consistent with the outliers of facet terms without phrases but resulted in a lower overlap.

Further looking into the Facet terms, it can be said that even though the overlap score is lower for facet terms with phrases, they are more meaningful when looked as phrases rather than as individual terms.

**T-test:**

A one-sample t-test using an alpha level of .05 compared the sample mean of facet terms with phrases created by various domain experts with the mean overlap = 0. The sample mean of 0.07 (SD = 0.05) was found to be statistically different from this value, \( t (405) = 28.11 \), “p< .0005”, one-tailed suggesting that the mean overlap score of various domain experts is greater than 0. The 95% confidence interval for the mean was -0.003 to 0.004. The effect size of the sample data was 0.81 and Cohen’s d value was 2.79.

**4.1.3. Facet groups without phrases**

The facet groups without phrases of vocabularies created by twenty nine different domain experts were compared; the sample size was 406 data points. The mean score was 0.06, and the median was 0.05. In terms of variability, the highest overlap score was 0.67 and the lowest overlap score was 0 yielding a range of 0.67 and a standard deviation of 0.08. Number of overlap scores that were zero was 141.

**Box plots:**

![Box plots: Comparison of facet groups without phrases](image)
The outlier pairs in the above box plot $X_{(15,22)} = 0.33$ and $X_{(16,23)} = 0.22$, are consistent with the outliers of facet terms. Domain Expert 5 in the above box plot has an outlier whose value is 0. While observing the overlap scores, it was found that this domain expert’s choice of facet groups did not match with the few other domain expert’s facet groups. One possible reason could be that since the choice of facet groups should be made from a smaller set of words unlike the choice of facet terms which can be made from wider set of terms, the probability that the domain expert’s choosing a similar facet group name is low. Also, overlap score of domain experts 24 and 25; $X_{(24,25)} = 0.22$ was significant when comparing the Facet terms, but in case of groups, none of them matched, even after the phrases were split into individual terms. This shows that the choice of grouping differs significantly among domain experts. To add credibility to the above statement, comparing the source documents between 24 and 25 (Figure 13: Comparison of Sources), provides an overlap score of 0.59, which means to a large extent, the sources are similar.

**T-test:**

A one-sample t-test using an alpha level of .05 compared the sample mean of vocabulary terms without phrases created by various domain experts with the mean overlap = 0. The sample mean of 0.06 (SD = 0.08) was found to be statistically different from this value, $t (405) = 14.61$, “p < .0005”, one-tailed suggesting that the mean overlap score of various domain experts is greater than 0. The 95% confidence interval for the mean was -0.004 to 0.004. The effect size of the sample data was 0.58 and Cohen’s d value was 1.45.

**4.1.4. Facet groups with phrases**

The facet groups with phrases of vocabularies created by twenty nine different domain experts were compared; the sample size was 406 data points. The mean score was 0.02, and the median was 0. In terms of variability, the highest overlap score was 0.30 and the lowest overlap score was 0 yielding a range of 0.30 and a standard deviation of 0.05. Number of overlap scores that were zero was 329.

**Box plots:**

![Box plots](image_url)
In the above box plot, it was observed that, most of the domain experts have zero overlap scores when compared with others. The same reason that was discussed in the case of facet groups with phrases will be applicable. Moreover, it was believed that the selection of facet groups with phrases will yield low overlap scores when compared to facet groups with phrases. One possible reason could be the choice of words i.e. for example one domain expert may choose the name “Conflation techniques” to represent various conflation algorithms like n-gram, successor variety etc, but some other domain expert may name it as “Conflation Methods.” It might yield in higher overlap scores for facet terms but will result in low overlap scores for facet groups with phrases. Many outliers can be seen in the above box plot. It was believed that since there is very low probability that the facet groups with phrases of two domain experts will match, the median overlap score tends to zero and any partial match between the facet groups with phrases are considered as outliers. It was observed that the median overlap for all the domain experts is 0.

**T-test:**

A one-sample t-test using an alpha level of .05 compared the sample mean of vocabulary terms with phrases created by various domain experts with the mean overlap = 0. The sample mean of 0.02, (SD = 0.05) was found to be statistically different from this value, t (405) = 8.40, “p< .0005”, one-tailed suggesting that the mean overlap score of various domain experts is greater than 0. The 95% confidence interval for the mean was -0.002 to 0.002. The effect size of the sample data was 0.58 and Cohen’s d value was 1.45.

**4.2. Experiment 2**

357 source documents selected by twenty six different domain experts have been compared and checked, for consistency between the source documents selected by the domain experts, by measuring the overlap. Correlation co-efficient was computed to determine if there exists any correlation between the sources selected and the vocabularies created.

The following figure represents more detailed information about the source documents selected by the domain experts.
Various source documents on conflation domain consisted of source code (109), online references (98), journals (87), text files (23), course notes (13), books (9) and manuals (9).

The conflation algorithms that were chosen are Porter (22), n-gram (11), Soundex(10), Lovins (11), Lancaster (Paice/Husk) (9), Successor variety (1) and Dawson (1). Some domain
experts have used more than 1 implementation of the algorithms in different languages. The algorithms chosen were in Java, C, C++, and Perl.

The following figure represents the frequency of some common algorithms chosen.

![Figure 11: Information of Conflation algorithms chosen by the domain experts.](image)

It was observed that Porter stemming algorithm was selected by most of the domain experts while Successor variety (1) and Dawson (1) were used by very few domain experts.

![Figure 12: Frequency of Source Documents selected by the domain experts (Pareto Plot).](image)
The pareto chart in the figure 16 shows the frequency of the source documents selected by the domain experts for the domain analysis. It was observed that the source code, “Porter in C” was used by 16 domain experts. The next frequently used source was the text book, “Stemming Algorithms in Information Retrieval” by W.Frakes and R. Baeza-Yates. It was referred by 11 domain experts. The Journals “An algorithm for suffix stripping” and “Another Stemmer” were used by 10 domain experts. The online reference, “Lancaster stemming algorithm” and the source codes, “Lovins in C” and “Paice in C” were commonly used by 8 domain experts. It was also observed that there were 96 unique sources.

**Box plots:**

![Box plots](image)

**Figure 13: Comparison of Sources**

The source information for some domain experts was not available. The most extreme outlier pair in the above box plot is domain expert X_{(24, 25)}. The source documents selected by these two domain experts are similar. One possible reason for the higher overlap could be the selection of source code. Both the domain experts have selected similar conflation algorithms (Lovins, Lancaster, Porter) and moreover these algorithms were implemented in same language (ANSI C). It was observed that the source documents selected by the different domain experts were not significantly different from each other. It was also observed that few sources were common among all the domain experts for e.g. Stemming Algorithms in Information Retrieval by W.Frakes and R. Baeza-Yates[33]. The extreme outlier pair X_{(24, 25)} has been discussed previously. The reason for most of the other outliers X_{(26, 28)}, X_{(7, 9)} in the above box plot is due to the close match of source documents

**Correlation:**
Correlation coefficient was computed to assess the relationship between the overlap scores of the source documents selected for the domain analysis and the overlap scores of the vocabularies created. It showed a positive correlation between overlap scores of facet terms with phrases and facet terms without phrases, $r = 0.83$, $p = 0.05$ and facet groups with phrases and facet groups without phrases, $r = 0.54$, $p = 0.05$. The matrix shows a low correlation between the overlap scores of documents and the vocabulary sets. A scatter plot summarizes the results (Figure 17). Overall, a strong, positive correlation between the overlap scores of facet group and facet term categories exists and a weak correlation between the overlap scores of sources documents selected for domain analysis and their corresponding domain vocabularies.

**T-test:**

A one-sample t-test using an alpha level of 0.05 compared the sample mean of source documents domain experts used to select domain vocabulary with the null hypothesis, mean overlap = 0. The sample mean of 0.09, (SD = 0.09) was found to be statistically different from
this value, \( t(559) = 21.99 \), “\( p < .0005 \)”, one-tailed suggesting that the mean overlap score of source documents selected by different domain experts is greater than 0. The 95% confidence interval for the mean was -0.006 to 0.007. The effect size of the sample data was 0.68 and Cohen’s d value was 1.86.

4.3. Experiment 3

Two sample T-tests were performed to evaluate the significance of the hypothesis, “the mean of the overlap scores created by domain experts is significantly different than mean of the vocabularies created by the automatic extractors”.

Figure 15: Two-sample T-test to compare means of Manual-Manual and Automatic-Manual (without stemming and stop words) facet terms with phrases overlap scores.

A two-sample t-test using an alpha level of .05 compared the sample mean of facet terms with phrases created manually by domain experts with those facet terms with phrases automatically generated by the automatic generators and without using stemming and stop words to test the null hypothesis that the two groups of vocabularies are not significantly different from each other. The means of two overlap scores has the estimated difference of 0.021, (Std Err = 0.005) was found to be statistically different from this value, \( t(388) = 3.70 \), “\( p < .0005 \)”, two-tailed suggesting that the mean overlap score of the facet terms with phrases created manually by domain experts are significantly different from the facet terms with phrases created by automated generators without using stemming and stop words. The effect size of the sample data was 0.18 and Cohen’s d value was 0.37.
Figure 16: Two-sample T-test to compare means of Manual-Manual and Automatic-Manual (with stemming and stop words) facet terms with phrases overlap scores.

A two-sample t-test using an alpha level of .05 compared the sample mean of facet terms with phrases created manually by domain experts with those vocabularies facet terms with phrases automatically generated by the information retrieval metrics and using stemming and stop words to test the null hypothesis that the two groups of vocabularies are not significantly different from each other. The means of two overlap scores has the estimated difference of -0.02, (Std Err = 0.007) was found to be statistically different from this value, t (281) = -2.79, “p< .0005”, two-tailed suggesting that the mean overlap score of the vocabularies created manually by domain experts are significantly different from the vocabularies created by automated generators using stemming and stop words. The effect size of the sample data was 0.16 and Cohen’s d value was -0.33.
Figure 17: Two-sample T-test to compare means of Manual-Manual and Automatic-Manual (with stemming and no stop words) facet terms with phrases overlap scores.

A two-sample t-test using an alpha level of .05 compared the sample mean of facet terms with phrases created manually by domain experts with those facet terms with phrases automatically generated by the automatic generators and using stemming and no stop words to test the null hypothesis that the two groups of vocabularies are not significantly different from each other. The means of two overlap scores has the estimated difference of -0.02, (Std Err = 0.007) was found to be statistically different from this value, $t(281) = -2.85$, “$p < .0005$”, two-tailed suggesting that the mean overlap score of the facet terms with phrases created manually by domain experts are significantly different from the facet terms with phrases created by automated generators using stemming and no stop words. The effect size of the sample data was 0.16 and Cohen’s d value was -0.34.
A two-sample t-test using an alpha level of .05 compared the sample mean of facet terms with phrases terms created manually by domain experts with those facet terms with phrases automatically generated by the automatic generators and using stop words and without stemming to test the null hypothesis that the two groups of vocabularies are not significantly different from each other. The means of two overlap scores has the estimated difference of 0.021, (Std Err = 0.005) was found to be statistically different from this value, t (340) = 3.80, “p< .0005”, two-tailed suggesting that the mean overlap score of the vocabularies created manually by domain experts are significantly different from the vocabularies created by automated generators using stop words and without stemming. The effect size of the sample data was 0.20 and Cohen’s d value was 0.41.
5. Conclusions

The overlap scores of vocabulary sets were computed and compared in terms of facet terms without phrases, facet terms with phrases, facet groups without phrases, and facet groups with phrases of different domain experts. It was observed that the facet terms and groups when selected without phrases yielded higher overlap scores than facet terms and groups with phrases. The results also show that the mean value of the overlap scores of the vocabularies and the source documents was significantly greater than 0. However, the overlap scores of the vocabularies and the source documents were not significantly different between the domain experts. Correlation co-efficient was computed to determine whether there exists any relationship between the vocabulary terms selected and source documents used to create them. There was a strong, positive correlation between the facet group and facet term categories and a weak correlation between the sources documents selected for domain analysis and their corresponding domain vocabularies. Also, the variability of the vocabularies generated automatically to the variability of those produced manually by domain experts was evaluated. The results show that the vocabularies are significantly different from each other.

5.1. Future work

It was observed that, in some cases, though the terms or groups selected by different domain experts were similar in meaning, these words were treated as distinct terms. This significantly reduced the overlap scores. For e.g. methods and techniques were terms that are contextually similar, these terms were treated as distinct. Experiments can be conducted by taking contextually similar words into consideration that would yield higher overlap scores and more importantly improving the accuracy in comparing vocabularies created by domain experts.
Bibliography

11. S. Seljan and A. Gašpar, “First Steps in Term and Collocation Extraction from English-Croatian Corpus.”
34. J.B. Lovins, Development of a Stemming Algorithm, 1968.
Appendix

Experiment 1:
Input - directories containing vocabularies (groups and terms) of domain experts

Process - Exp1Main.java, main program, takes vocabulary of two domain experts at a time and calculates the overlap

Scanfile.java is called by the main program and takes each domain expert’s vocabulary and performs, data cleansing, splitting the data into terms. Perform stemming and output best possible terms for matching. With various switches, the code could be reused between different approaches (terms with phrases, terms without phrases etc.)

Overlap.java is also called by the main program. It takes cleaned and stemmed vocabulary of any two domain experts as input, and calculates the overlap.

Exp1Main.java

package org.vinnu;
import java.io.File;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Arrays;
public class Exp1Main {
    /**
     * @param args
     * @throws FileNotFoundException
     */
    public static void main(String[] args) throws FileNotFoundException {
        ArrayList<String> vocab1 = new ArrayList<String>();
        ArrayList<String> vocab2 = new ArrayList<String>();
        String top_dir = "D:\Documents\Documents\MIS\Fall09\CS5974\Vocabulary_Exp1\"";
        File root = new File(top_dir);
        String[] dirs = root.list();
        Arrays.sort(dirs);
        for (int i = 0; i < dirs.length; i++)
        {
            //System.out.println(dir1.getPath());
            ScanFile sf1 = new ScanFile();
            //To split the word - give the boolean value -False else True
            vocab1 = sf1.ReadDir(top_dir + dirs[i], "terms", true);
            for (int j = 0; j < dirs.length; j++)
            {
                //For Experiment 1
                //System.out.println("Elements in Test1:" + vocab1);
                ScanFile sf2 = new ScanFile();
                vocab2 = sf2.ReadDir(top_dir + dirs[j], "terms", true);
                //System.out.println("Elements in Test2:" + vocab2);
                Overlap op = new Overlap(vocab1, vocab2);
                System.out.println(dirs[i] + ":" + dirs[j] + ":" + op.getOverlap());
            }
        }
    }
}
public class ScanFile {

    /**
     * protected ArrayList<String> wordAL = new ArrayList<String>();
     */
    * @param dir_path
    * @param ListType
    * @param split
    */
    public ArrayList<String> ReadDir(String dir_path, String ListType, boolean split) {
        ArrayList<String> words = new ArrayList<String>();
        try {
            File dir = new File(dir_path);
            String[] files = null;
            if (dir.isDirectory()) {
                files = dir.list(new FilenameFilter() {
                    public boolean accept(File dir, String name) {
                        return !name.endsWith("~");
                    }
                });
            }
            Arrays.sort(files);
            for(String fname : files) {
                //check if "terms" or "groups"
                if(fname.contains(ListType)) {
                    //Call ReadFile method and pass the information to it.
                    words = ReadFile(dir_path + "/" + fname,split);
                }
            }
        } catch (FileNotFoundException ex) {
            Logger.getLogger(ScanFile.class.getName()).log(Level.SEVERE, null, ex);
        }
        return words;
    }

    /**
     * @param Filename
     * @param useDelimiter
     * @return
     * @throws FileNotFoundException
     */
    public ArrayList<String> ReadFile(String Filename, boolean useDelimiter) {
        return null;
    }
}
*\npublic ArrayList<String> ReadFile(String Filename, boolean boolDelim) throws 
FileNotFoundException 
{
    FileReader fr = new FileReader(Filename);
    Scanner sr = new Scanner(fr);
    //if false, then split words in each line,
    if(boolDelim)
        sr.useDelimiter("\n");
    String word=null;
    Stemmer stem = new Stemmer();
    StopWords stp = new StopWords();
    while(sr.hasNext())
    {
        word = sr.next().trim();
        if(!word.isEmpty()) //To remove empty lines
        {
            //Words separated by special characters or space in a line
            Pattern p = Pattern.compile("[\s()-&/\\//\ ]");
            Matcher m = p.matcher(word.trim());
            //if(word.contains(" ") || word.contains("-")||word.contains("&"))
            if(m.find())
            {
                String outWord = ";",
                word = word.replaceAll("[\s()-&/\\//\ ]"," ").replace("   ", " ").replace(" ", " ");
            // System.out.println(word);
            Scanner ssr = new Scanner(word);
            ssr.useDelimiter("\s");
            ArrayList<String> outWords = new ArrayList<String>();
            while(ssr.hasNext())
            {
                word = ssr.next();
                //word = stem.stem(word.trim());
                StopWords stop = new StopWords();
                List<String> stopWords = stop.stopList;
                for(String stpWord : stopWords)
                {
                }
                //For Unsplit
                if(boolDelim)
                    outWord += word + " ";
                //else
                // System.out.println("In if " + word);
                outWords.add(word.trim());
                //outWord = word;
            }
            if(boolDelim) {
                if(!wordAL.contains(outWord.trim().toLowerCase()))
                    //System.out.println("In if " + outWord);
                    wordAL.add(outWord.trim().toLowerCase());
            }
            else
            {
                for (String inWord:outWords)
                {
                    //System.out.println("In if " + inWord);
                    if(!wordAL.contains(inWord.trim().toLowerCase()))
                        wordAL.add(inWord.trim().toLowerCase());
                }
            }
        }
    }
    word.trim().toLowerCase();
}
Stemmer.java

Refer to Porter Stemmer Implementation in Java [42]

Experiment 2:

Input - files containing title of sources used by domain experts

Process - Exp1Main.java, main program, takes titles of 2 domain experts at a time and calculates the overlap

RefMatch.java is called by the main program and takes each domain expert’s titles and performs data cleansing.

Overlap.java is also called by the main program. It takes cleaned source document titles of any 2 domain experts as input, and calculates the overlap.

Exp2Main.java

package org.vinnu;

import java.io.*;
import java.util.*;
public class Exp2Main {
    /**
     * @param args
     * @throws FileNotFoundException
     */
    public static void main(String[] args) throws FileNotFoundException {
        ArrayList<String> title1 = new ArrayList<String>();
        ArrayList<String> title2 = new ArrayList<String>();
        String top_dir = “/media/sdb2/Documents/Documents/MIS/Fall09/CS5974/Exp2/”;
        File root = new File(top_dir);
        String[] files = root.list();
        // System.out.println(files.toString());
        Arrays.sort(files);

        //word = stem.stem(word);
        //System.out.println(word);
        if(!wordAL.contains(word.trim().toLowerCase()))
            wordAL.add(word.trim().toLowerCase());
        //System.out.println("In Else " + word);
    }
    return wordAL;
}
for(String file1:files)
{
    for(String file2:files)
    {
        //For Experiment 2
        RefMatch rm1 = new RefMatch();
        title1 = rm1.ReadFile(top_dir+file1);
        RefMatch rm2 = new RefMatch();
        title2 = rm2.ReadFile(top_dir+file2);
        Overlap op = new Overlap(title1,title2);
        System.out.println( file1 + ":" + file2 + ":" + op.getOverlap());
        title1.clear();
        title2.clear();
    }
}

RefMatch.java

package org.vinnu;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.Scanner;
public class RefMatch {
    /**
     * @param Filename
     * @return
     * @throws FileNotFoundException
     */
    protected ArrayList<String> titleAL = new ArrayList<String>();
    public ArrayList<String> ReadFile(String Filename) throws FileNotFoundException
    {
        FileReader fr = new FileReader(Filename);
        Scanner sr = new Scanner(fr);
        sr.useDelimiter("[
        
        String title=null;
        while(sr.hasNext())
        {
            title = sr.next().trim();
            title = title.replace(" ", ");
            if(!title.isEmpty()) //To remove empty lines
            {
                // System.out.println(title.toLowerCase());
                if(!titleAL.contains(title.toLowerCase()))
                titleAL.add(title.toLowerCase());
            }
        }
        return titleAL;
    }
}

Overlap.java

package org.vinnu;
import java.util.ArrayList;
public class Overlap {
    ArrayList<String> _vocab1 = new ArrayList<String>();
    ArrayList<String> _vocab2 = new ArrayList<String>();

    /**
     * @param vocab1
     * @param vocab2
     */
    public Overlap(ArrayList<String> vocab1, ArrayList<String> vocab2) {
        this._vocab1 = vocab1;
        this._vocab2 = vocab2;
    }

    /**
     * @return
     */
    protected int UnionList() {
        ArrayList<String> Union = new ArrayList<String>(){
            Union.addAll(this._vocab1);
            for (String word : this._vocab2) {
                if (!Union.contains(word))
                    Union.add(word);
            }
        }
        //System.out.println("The Union of two files is:" + Union.size());
        return Union.size();
    }

    /**
     * @return
     */
    protected int IntersectList() {
        ArrayList<String> Intersect = new ArrayList<String>(){
            for (String word : this._vocab1) {
                if (this._vocab2.contains(word))
                    Intersect.add(word);
            }
        }
        //System.out.println("The intersection of two files is: " + Intersect.size());
        //System.out.println(Intersect);
        return Intersect.size();
    }

    /**
     * @return
     */
    public double getOverlap() {
        double overlap = (double)IntersectList() / (double)UnionList();
        return overlap;
    }
}