8-2007

A COMPARATIVE STUDY ON ONTOLOGY GENERATION AND TEXT CLUSTERING USING VSM, LSI, AND DOCUMENT ONTOLOGY MODELS

William Taylor
Clemson University, wptaylo@clemson.edu

A COMPARATIVE STUDY ON ONTOLOGY GENERATION AND TEXT CLUSTERING USING VSM, LSI, AND DOCUMENT ONTOLOGY MODELS

A Thesis
Presented to
The Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Computer Science

---

by
William P. Taylor II
August 2007

Accepted by:
Dr. James Wang, Committee Chair
Dr. Feng Lou,
Dr. Mark Smotherman
Dr. Srimani

ABSTRACT

Although using ontologies to assist information retrieval and text document processing has recently attracted more and more attention, existing ontology-based approaches have not shown advantages over the traditional keywords-based Latent Semantic Indexing (LSI) method. This paper proposes an algorithm to extract a concept forest (CF) from a document with the assistance of a natural language ontology, the WordNet lexical database. Using concept forests to represent the semantics of text documents, the semantic similarities of these documents are then measured as the commonalities of their concept forests. Performance studies of text document clustering based on different document similarity measurement methods show that the CF-based similarity measurement is an effective alternative to the existing keywords-based methods. Especially, this CF-based approach has obvious advantages over the existing keywords-based methods, including LSI, in dealing with text abstract databases, such as MEDLINE, or in P2P environments where it is impractical to collect the entire document corpus for analysis.

ACKNOWLEDGEMENTS

I would like to thank the faculty of the department for advice and help throughout this challenging endeavor. I am grateful to Dr. James Wang for providing the initial inspiration for this work, and for his patience, advice, and suggestions throughout the life of this project. I also want to thank Dr. Smotherman for his guidance on many matters during my study at Clemson.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

CHAPTER ONE

Introduction

While the growth of World Wide Web has invigorated research on information retrieval and text mining, the success of GOOGLE [1] further validates the importance of information retrieval and text mining in the Internet era. Currently, keywords-based techniques are commonly used in various information retrieval and text mining applications. Among these keywords-based methods, Vector Space Model (VSM) [2] and Latent Semantic Indexing (LSI) [3] are the most widely adopted. Using VSM, a text document is represented by a vector of the frequencies of terms appearing in this document. The similarity between two text documents is measured as the cosine coefficient between their term frequency vectors. However, a major drawback of the keywords-based VSM approach is its inability of handling the polysemy and synonymy phenomenon of the natural language. As meanings of words and understanding of concepts differ in different communities, different users might use the same word for different concepts (polysemy) or use different words for the same concept (synonymy). Thus, matching only keywords may not accurately reveal the semantic similarity among text documents or between search criteria and text documents due to the heterogeneity and independency of data sources and data repositories. For example, the keyword "java" can represent three different concepts: coffee, an island, or a programming language (polysemy), while keywords "dog" and "canine" may represent the same concept (synonymy) in different documents.

LSI tries to overcome the limitation of VSM by using statistically derived conceptual indices to represent text documents and queries. LSI assumes that there is an underlying latent structure in word usage that is partially obscured by variability of word choice and tries

to address the polysemy and synonymy problems through modeling the co-occurrence of keywords in documents. Though earlier studies contend that LSI may implicitly reveal concepts through the co-occurrence of keywords, we found that the co-occurrence of keywords in documents may not necessarily mean the contextuality, especially in multi-disciplinary research papers such as biomedical research papers or in Web documents discussing multiple topics. This is exactly why using LSI-based tools to extract terms from commercial web documents, which may contain ads, headlines, and new feeds, is a questionable practice. Therefore, we must instead look at the co-occurrence of conceptually similar terms. In addition, some text document archives, such as MEDLINE [4] database and web blogging entries, contain primarily short abstracts or articles instead of long papers. These short documents may not provide enough co-occurrence information for LSI-based semantic similarity measurement. Furthermore, in dynamic environments, such as live news feeds or P2P systems, it is impractical to collect the entire document corpus for analysis.

Thesis Organization

This thesis consists of eight chapters. Chapter one, as already presented, includes the introduction, and overview of our research work. In Chapter two some of the related works are briefly discussed and the motivation of the project is presented. In Chapter three we give a brief description of the WordNet Lexical Database, how preprocessing and ontology generation is done, and in Chapter four we give a brief description of the Vector Space Model and Latent Semantic Indexing, while also reviewing the ontology comparison tool; towards the end of Chapter four we explain the importance of clustering. In Chapters five and six we present the performance results of our research, definitions and concepts, clustering analysis, and runtime analysis, along with the limits and disadvantages that were

uncovered when performing the tests. We discuss the conclusion and future work in Chapter eight.

CHAPTER TWO

Related Work

Recently, many studies tried to use ontologies to assist information retrieval and text document processing. These ontology-based approaches can be divided into two categories. One category of ontology-based methods [5], [6], [7], [8], [9], [10], [11], and [12] apply machine learning methods, such as clustering analysis and fuzzy logic, to construct ontologies from text documents and, then, use these ontologies to assist information retrieval and text document processing [13], [14], [15], [16], and [17]. However, these methods require analyzing the entire document corpus to construct a good ontology, and the performance of information retrieval and text document processing depends on how good the constructed ontologies are. During the corpus analysis, terms rarely appearing in the document corpus are often ignored because of their low frequencies of occurrence. However, high information content of these rare terms is valuable for information retrieval according to information theory. Ignoring these terms in the constructed ontologies may affect the performance of information retrieval and text document processing. Nonetheless, these ontology-based methods have not been fully evaluated against the keywords-based LSI method, arguably the best keywords-based method.

The second category of ontology-based methods utilizes an existing ontology, such as WordNet [18] to assist information retrieval and text document processing. These methods use three different approaches to take advantage of the existing ontological knowledge. The first approach [19], [20], [21], and [22] involves using WordNet to find synonyms or hypernyms of terms to improve the performance of information retrieval and text document processing. However, this approach may introduce "noise" by adding

semantic content that is not present in the document corpus. For instance, given a document about "beef" and a document about "pork", a hypernym-based method may use "meat" to replace "beef" and "pork" because two terms have a common hypernym "meat". This approach over-simplifies or over-generalizes the problem, making it impossible to distinguish documents containing "beef" from documents containing "pork". Another problem with this approach is that it does not perform word sense disambiguation. Instead, all synonyms or hypernyms related to a keyword are used to replace the keyword. These weaknesses often lead to disappointing information retrieval and text document processing performance [23] and [24]. The second approach focuses on word sense disambiguation [25], [26], [27], [28] to address the synonymy and polysemy problem in natural language processing. However, this approach tries to determine an exact sense for a term, often resulting in misclassification of terms. This approach also ignores the impact of the semantic similarities and relationships among different terms in the same text document on the performance of information retrieval and text document processing.

To address the problems in the first two approaches, the third approach applies various techniques [29], [30], [31], and [32] to discover the semantic similarities and relationships of terms and uses them to enhance the keywords-based information retrieval and text document processing methods, such as VSM. However, the techniques used to discover the term relationships and similarities have their weaknesses. Sedding [29] used a naive, syntax-based disambiguation approach by assigning each word a part-of-speech (POS) tag and by enriching the "bag-of-words" data representation, which were often used for document clustering by extracting synonyms and hypernyms from WordNet. Unfortunately, this study found that including synonyms and hypernyms, disambiguated only by PoS tags,

does not improve the effectiveness of text document clustering. The authors attributed this underperformance to the noise introduced by incorrect senses retrieved from WordNet and concluded that disambiguation by PoS alone is insufficient to reveal the full potential of including background knowledge in information retrieval and text document processing. To further investigate this issue, Simone [30] proposed a document search technique that uses other methods, in addition to POS tagging, to cluster search results into meaningful categories according to the words that modify the original search term in the text document. This work focuses on determining if the antonymy relation, instead of synonyms and hypernyms, could be used on the modifiers found in documents to decompose a set of search results into a hierarchy of sub-clusters. Unfortunately, their experimental studies again suggest that this approach cannot improve the performance of information retrieval.

While these two studies suggest exploiting term relationships or similarities using WordNet may not improve the performance of information retrieval and text document processing, other studies using different methods imply that it is possible to use term relationships or similarities to improve the performance of the keywords-based VSM. Huang [31] used a guided self-organization map (SOM), a result of merging statistical methods, competitive neural models, and semantic relationships obtained from WordNet, to improve the performance of the traditional VSM. However, certain human involvement is required to build the guided SOM. Jing [32] calculates a mutual information matrix for all terms in the documents based on information obtained from WordNet and uses the mutual information to enhance the keywords-based VSM method. However automatically computing term mutual information (TMI) is sometimes problematic and may lead to wrong conclusions about the quality of the learned mutual similarity [33]. Even though using SOM and TMI

can improve the performance of the keywords-based VSM, their performance in comparison with LSI, the best keywords-based method, has not been investigated. Furthermore, these methods require analyzing the entire document corpus as VSM and LSI do.

To address the problems in existing ontology-based methods, we propose a new ontology-assisted method to measure the semantic similarity of text documents. This new method constructs a concept forest (CF) from a text document, based on the co-occurrence of terms and their semantic relationships found in WordNet. Using the CF to represent the semantics text documents, we propose a simple method to measure the semantic similarity of two text documents. One unique feature of our proposed CF-based method is that we derive the concept forest based only on analyzing the co-occurrences and relationships of terms within a single document. Conversely, existing approaches all require analyzing the entire text document corpus to determine the semantic similarity of two text documents. Therefore, our CF-based method is a practical alternative to the existing information retrieval and text document processing methods in dynamic environments such as P2P systems and live news feeds, where it is impractical to collect the entire document corpus for analysis.

CHAPTER THREE

Approach

Our approach will be presented in the following series: First we will begin with an overview of the lexical database WordNet that is used to aid in constructing the Concept Forest (CF) of a document, this will be followed by an overview of how our overall system functions with the addition of using WordNet, and lastly we provide a detailed description of each component that is comprised of our system namely: document preprocessing, CF generation, similarity measurements implemented (SS, VSM, LSI), and the clustering algorithms used; below we proceed with describing WordNet v.2.1.

WordNet Lexical Database

WordNet® is a large lexical public database of English words. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The purpose is to produce a combination of a dictionary and a thesaurus that is more intuitively usable, and to support automatic text analysis. The database contains approximately 150,000 words organized in over 115,000 synsets. Every synset contains a group of synonymous words or collocations; different senses of a word are in different synsets. Most synsets are connected to other synsets through many semantic relations these relations differ based on the type of word and include:

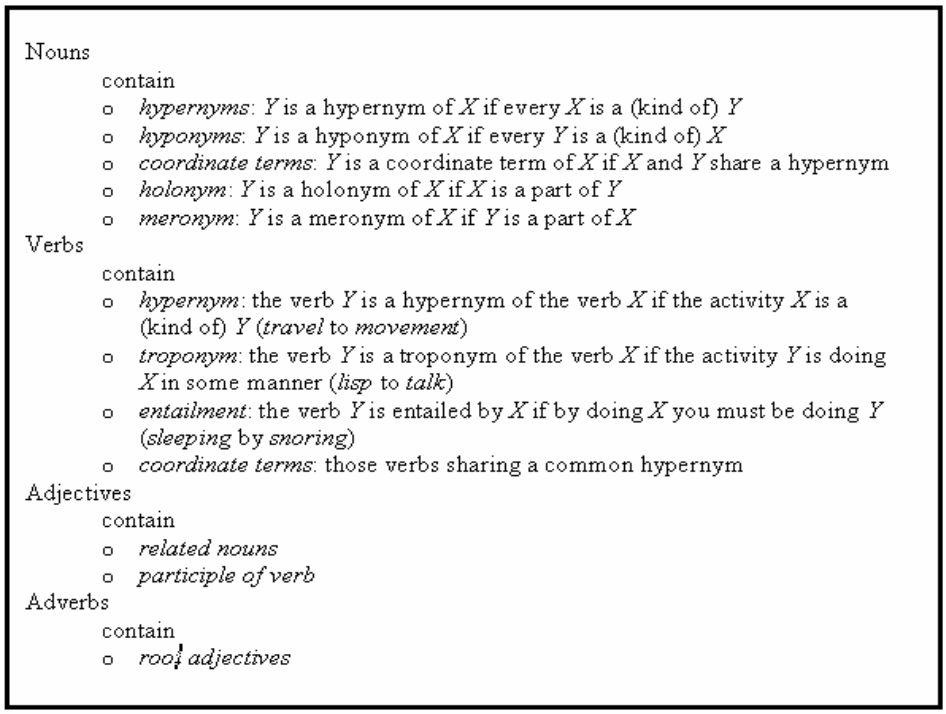

Figure 1: WordNet Hierarchy & Synset Attributes

WordNet Synset Structure

Both nouns and verbs are organized into hierarchies, defined by hypernym or IS A relationships. For instance, looking at Figure 2, the first sense of the word dog would have the following hypernym hierarchy; the words on the same level are synonyms of each other: some sense of dog is synonymous with some other senses of domestic dog and Canis familiaris, and so on. Each set of synonyms (synset – Synonym Set), has a unique index (called SynsetIDx) and shares its properties, such as a gloss (or dictionary) definition.

```
dog, domestic dog, Canis familiaris
   => canine, canid
      => carnivore
        => placental, placental mammal, eutherian, eutherian mammal
          => mammal
            => vertebrate, craniate
              => chordate
                => animal, animate being, beast, brute, creature,fauna
                  => ...
```

Figure 2: Hierarchy of Relationships of Synsets

At the top level, these hierarchies are organized into base types, 25 primitive groups for nouns, and 15 for verbs. These primitive groups are connected to an abstract root node that has been assumed by various applications that use WordNet. In the case of adjectives, the organization is different. Two opposite 'head' senses work as binary poles, while 'satellite' synonyms connect to each of the heads via synonymy relations. Thus, the hierarchies, and the concept involved with lexicographic files, do not apply here the same way they do for nouns and verbs. The network of nouns is far deeper than that of the other parts of speech. Verbs have a far *bushier* structure, and adjectives are organized into many distinct clusters. Adverbs are defined in terms of the adjectives they are derived from, and thus inherit their structure from that of the adjectives. For a complete description of WordNet see [18].

Approach Overview

Our approach consists of taking as input a document corpus *C*. For every document *d* in *C* that is imported into our software, we extract the keywords from every document and create a wordlist structure (see Figure 3) that maintains information about each keyword. Using WordNet, the software then retrieves the synsets for each keyword in the WordList.

Then, utilizing the hierarchy structure of the synsets the software finds relationships among keywords, and constructs the CF using the present relationships; for each document its corresponding CF is output to a file. This approach is different from existing approaches because the ontology that is created is comprised of the keywords and the SynsetIDs that ONLY share relationships with other terms in the document. After each document is made into its corresponding CF, we apply a similarity measurement algorithm (VSM, LSI, and SS) that outputs a similarity matrix, which is representative of each documents similarity to every other document in the corpus. This similarity matrix is then imported into the Java LING PIPE clustering software. LING PIPE is then applied to group the documents into *N* categories.

Preprocessing

When a document is initially loaded into the C program, the software takes advantage of the string manipulation capabilities of PERL and calls a PERL script that creates a temporary file that contains every keyword and the frequency of the appearance of that keyword. The C program reads data from the temporary file that was created and compares each word to the words in a 'stop-list'; the 'stop-list' is a character array of 556 words consisting of pronouns, common verbs, common nouns, adjectives, and frilly words. The words chosen to be included in the stop-list add little or no semantic value in determining the semantic content of the document. These words were adopted through research of how the LSI and VSM models best represent data [47].

For each word that is not found in the stop-list we creates a data 'wordList' data structure (see Figure 3) – that stores information about the term, i.e. word frequency, number of relationships it has with other words (described later), Noun Synset List, Verb

Synset List, etc. By integrating the WordNet database into the software we gain the capability of utilizing methods provided through WordNet; a method that is used is morpstr. Morpstr is a method that provides a solution to the problem of stemming. Stemming is the process of mapping inflected (or sometimes derived) words to their stem, base or root form — generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root i.e. mapping 'cared', 'cares', and 'caring' to the root word 'care'.

```
struct wordList
{
  char* kword;
  int w_count;                  //Total Words found in WordNet
  int depthV;int depthV;        //Depth of the Verb Hypernyms
  int depthN;                   //Depth of the Noun Hypernyms
  long sharedRel[250];          //Each word that has a relationship
                                //with the SynsetIDx added to SharedRelationship
  int sharedPos;
  int listNodeCnter;
  struct wordList** children;
  struct wordList* nxtWord;     //Next Word (extracted from Document)
  struct wordList* listNodes[250]; //Ptrs to sharedPos nodes
  SynsetPtr SSLink_V;           //Verb Hypernyms
  SynsetPtr SSLink_N;           //Nouns Hypernyms
} *wordRoot;
```

Figure 3: WordList Structure used to track statistics and relationships among terms

After each term has been reduced to its base/root word form the software cycles through each term in the wordList and begins to retrieve the synsets (synonym sets) for the noun and verb parts of speech. This concludes the preprocessing phase; Concept Forest construction follows.

Concept-Forest (CF) Construction

Our CF-based method includes three steps: concept forest construction, semantic content purification, and similarity measurement calculations. Once the synsets have been retrieved and stored for each term in the document, to produce the ontology for the document two items need to occur: first, establishing relationships between document terms and secondly, creating an n-ary tree to represent topics/subjects in the document from the relationships that were formed. First we will describe how we establish relationships between terms and then we will focus on tree construction for CF generation.

Mapping Relationships

In the event that two terms in a document share a similar meaning it is our duty to capture the concept that both terms are referring to; we capture this concept through the terms SynsetID. The SynsetID is a numerical value that is associated with a group of synonyms that identifies the synonym set (synset). For example, if we were to look at the term 'Java' we are able to identify the following:

Java:
Sense 1: Coffee, cafe (SynsetID: 67893543)
Sense 2: Programming, Programming Language (SynsetID: 71154118)
Sense 3: Island (SynsetID: 82735541)

'Java' has three senses, and a corresponding SynsetID for each sense. The SynsetID represents all of the words/synonyms of that sense. SynsetID 71154118 corresponds to the concept of 'Programming' and 'Programming Language'; therefore if a document is about 'Java the Programming language', sense one and sense three will be disqualified and we would use the SynsetID of sense two in constructing the CF. Now that we have established

what and how a SynsetID is used we now explain the process of mapping relationships between terms and in disqualifying or narrowing a terms senses to select the correct sense the document is referring to.

To map relationships amongst terms the program cycles through each term and each term's synset list comparing the synset list of one term to another term in the document (see Figure 4).

```
Produces relationships
For each term (t1) in a Document (D1)
  For each synset (s) for (t1)
     Search for term (t2) in Document (D1) in the synset (s) of (t1)
        If t2 equals t1
                break;
        End If
        If Found
                defineRelationship(t1, t2)
                addSynsetIDx_toRelationshipList(t2, t1)
        End IF
        Else
                t2 = nextTerm(Document (D1))
  End For
                t1 = nextTerm(Document (D1))
End For
```

Figure 4: Ontology Generation Pseudo-code

Given two terms – $term_1$ and $term_2$ - if $term_2$ is found in $term_1$ synset list then a relationship is made and the SynsetID of $term_1$ synset is added to $term_2$ shared relationship list, if not the program then compares $term_3$ with $term_1$ synset list, and so on, until all terms have been evaluated. This approach gives us a method that allows us to determine if two terms are semantically related (see Figure 5) by comparing the synsets in a terms hypernym hierarchy; next we discuss tree construction.

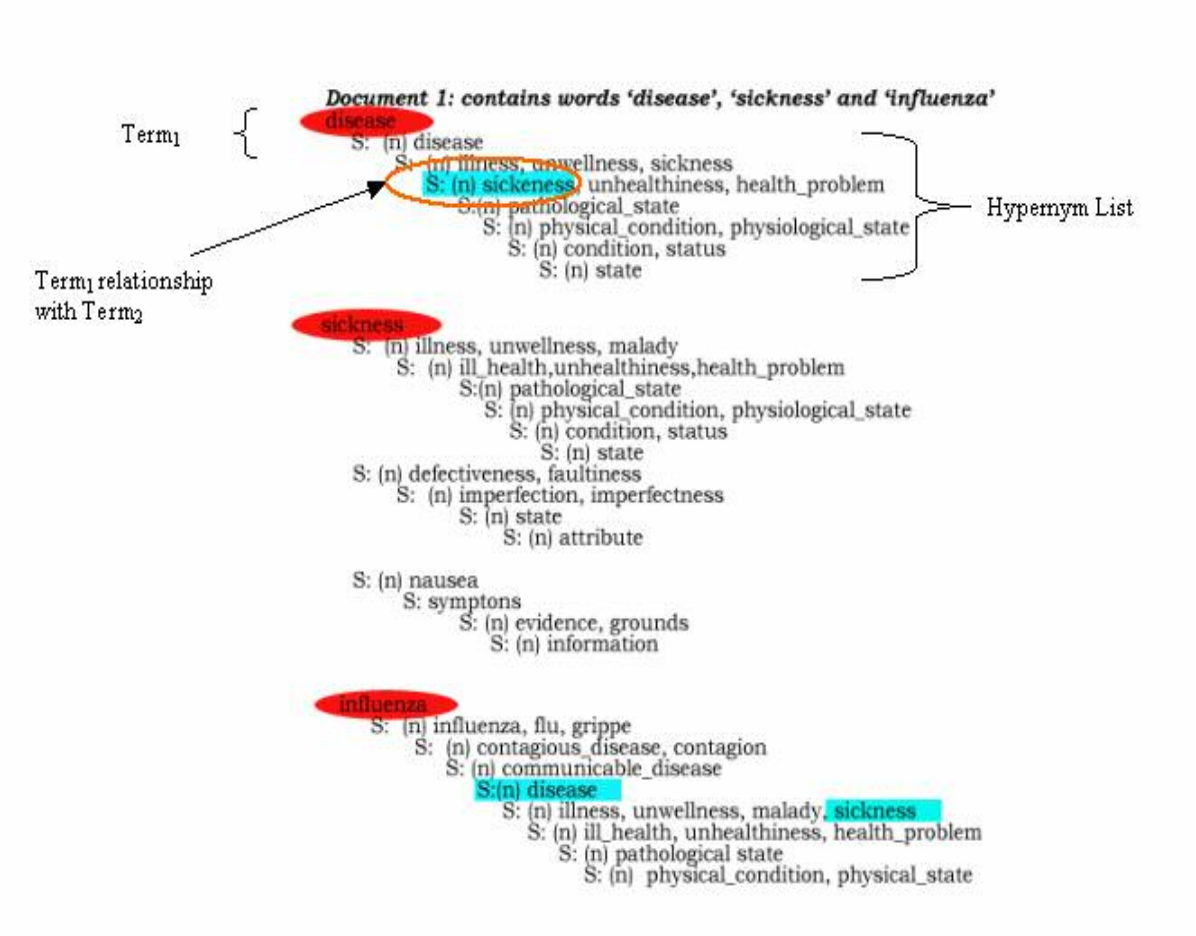

Figure 5: Finding Relationships among Terms

*Searching through the terms hypernym list we are able to find IS-A relationships and map terms together using the relationships.

After all terms have been evaluated the program attempts to construct a tree that is representative of the subjects within the document. To begin, we assign the first term in the list as a $root_1$ node. After assigning the $root_1$ node we begin to account for each of the relationships the $root_1$ term has with other terms; these other terms are treated as children. We then process the children in the same fashion as the $root_1$ term; for each child of the root, we account for it's relationships amongst other terms in the document and those terms

become the children's children, and so on – to avoid cycles/loops within the tree structure we store all processed terms inside of an array – if a term is unprocessed, meaning it is not found in the tree, the term is added and counted as processed. After all of the relationships of the terms in a tree have been accounted for, the tree is finished. If there exists term(s) in the list/document that have not been processed by the completion of a tree this routine restarts; the unprocessed term becomes $root_2$, we account for it's children, and so on (see Figure 6).

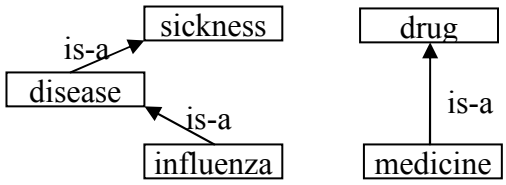

Figure 6: Creating Trees from Relationships of Terms

Semantic Content Purification

We utilize the terms word frequency to derive a value for the total weight of the tree. Each term has an associated word frequency value; this value corresponds to the number of occurrences this term was found in the document. We perform a depth first search on each tree adding the word frequency for each term (or more simply as the tree is being constructed we add the terms word frequency). Then we take the resulting sum and divide this number by the total words in the word list (stop words excluded); the quotient multiplied by one hundred is the Semantic Concept Purification (SCP) value (see Figure 7).

$$SCR_i = w_i \Big/ \sum_{j=1}^{n} w_j$$

Figure 7: Calculating the Semantic Concept Purification (SCP) value of the CF

The SCP value directly corresponds to the semantic organization of the document. Ideally a clearly and concisely written document should have an SCP value above 85%, meaning that most of the terms in the document are related in some fashion to each other, meaning that the document contained little noise or references to concepts unrelated to the topic of the document. Because it is possible for the contents of a document to express *N* different ideas, this method would generate *N* different trees (unless the ideas were inter-related) – each having associated with it an SCP value (see Figure 8 for pseudo-code). The higher the SCP value the better the chances of constructing a Concept Forest that is representative of the document; this solution is implemented to reduce noise in documents.

```
Build Tree Structure of Relationships:
For each term (t1) in WordList
        If RelationshipList(t1) < 1
                Term (t1) is not related to any other term in the document
                Output t1
                treePercent [root] = treePercent[root] + t1 -> word_count;
        End If
        Else
                If !processed(t1->synsetIDx)
                   treePercent [root] += t1 -> word_count
                   while i < RelationshipList(t1)
                         If !(processed(getRelationshipList(t1,i)
                             stack.push(getRelationshipList(t1,i)
                         End If
                     End while
                End If

        /* Establish Children */
        while stack.NotEmpty()
                t1 = stack.top();
                If !processed(t1)
                   treePercent [root] += t1 -> word_count
                   processed(t1->synsetIDx)
                   while i < RelationshipList(t1)
                         If !(processed(getRelationshipList(t1,i)
                             stack.push(getRelationshipList(t1,i))
                         End If
                     End while
                End If
        End While

    treePercent [root] = treePercent[root] / totalWordsInDoc;
    root++
End For
```

Figure 8: Pseudo-code to construct Tree Based on Term Relationship

To generate the CF for the document we utilize a SCP threshold value. The SCP value of each tree is used to help determine which trees/subjects the most emphasis should be placed upon. By default the SCP threshold value is set to 5%. For every tree whose SCP value is above 5% the program writes to a file the term, and each of its synonyms a total of "word frequency" times – where "word frequency" is the number of occurrences the term was found in the document. If the SCP value is 5% or lower, for each term in the tree, only the term is included in the CF. In this way greater emphasis will be placed on trees with higher SCP values and VSM and LSI will naturally reflect this weighting system because of the inherit nature of the respective algorithms. The program can also be modified to entirely disregard trees whose SCP value is below a given threshold or to disregard SCP values entirely; we have stayed away from testing these cases due to purposely wanting to incorporate unique words that may not be related to other terms and also to incorporate new concepts that WordNet may not contain information on. Next we use a similarity measurement algorithm (SS, VSM, or LSI) that determines the similarities between two CFs or two documents. We look into how the similarity measurements compare CFs or documents next.

Similarity Measurements

Semantic Similarity Measurement

Using a concept forest to represent the semantic content of a text document, the semantic similarity of two text documents can be determined by comparing their concept forests. To facilitate the text document similarity measurement, we give a formal definition of the concept forest here. A concept forest is defined as a Directed Acyclic Graph (DAG): CF = [T, E, R], where T = $\{t_1, t_2, \ldots, t_n\}$ is a set of stemmed words or synsetIDs, and E =

$\{e_1, e_2, \ldots, e_m\}$ is a set of edges connecting synsetIDs with relationships defined in $R = \{r_1, r_2, \ldots, r_k\}$. Specifically, an edge e_i is defined as a triplet $[t_{i1}, t_{i2}, r_j]$ where $t_{i1}, t_{i2} \in T$ and $r_j \in R$. In addition, two terms can be linked by only one relationship, that is,

$\forall l \neq k, [t_i, t_j, r_k] \in E \Rightarrow [t_i, t_j, r_l] \notin E$.

Given two concept forests $CF_1 = [T_1, E_1, R_1]$ and $CF_2 = [T_2, E_2, R_2]$, determining their similarity needs to consider the similarities of their associated term sets, edge sets, and relationship sets. However, in this study, we use only hypernym ("is-a") relationship to construct the concept forest. Therefore, we calculate the semantic similarity of two text documents by simply comparing the similarity of two term sets (T_1 and T_2), hoping this simple measurement is sufficient for information retrieval and text document processing.

Given two documents D_1 and D_2, and their concept forests $CF_1 = [T_1, E_1, R_1]$ and $CF_2 = [T_2, E_2, R_2]$ respectively, the semantic similarity between these two documents is measured as:

$$SS(CF_1, CF_2) = \frac{(CF_1 \bigcap CF_2)}{(CF_1 \bigcup CF_2)}$$

Equation 1: Semantic Similarity

Another approach is switching the input of the Vector Space Model and Latent Semantic Indexing algorithms from keyword based to that of an CF-based input scheme. The details of the implementation of the VSM and LSI models remain the same but after constructing the CF we then use as input the CF in place of using the original document – the results of which can be found to in the results section of this paper; now that we have given an example of how to compute the Semantic Similarity of an CF (the same can be

applied to the extracted keywords of a document) we proceed to describing the second implemented similarity measure that was used in this system, the vector space model.

Vector Space Model

We implemented the Vector Space Model (VSM) was implemented using PERL v5.6.2 and VSM can be divided into three stages. The first stage is the document indexing where content bearing terms are extracted from the document text. The second stage is the weighting of the indexed terms to enhance retrieval of document relevant to the user. The last stage ranks the document with respect to the query according to a similarity measure.

By using automatic document indexing non-significant words such as, common pronouns, common verbs, participles, etc. are removed from the document vector, so the document will only be represented by content bearing words. This indexing can be based on term frequency, where terms that have both high and low frequency within a document are considered to be function words.

A common weighting scheme for terms within a document is to use the frequency of occurrence as stated by Luhn [36]. The term frequency is somewhat content descriptive for the documents and is generally used as the basis of a weighted document vector. It is also possible to use binary document vector, but the results have not been as good compared to term frequency when using the vector space model.

The similarity in vector space models is determined by using associative coefficients based on the inner product of the document vector and query vector, where word overlap indicates similarity. The inner product is usually normalized. The most popular similarity measure is the cosine coefficient, which measures the angle between a document vector and

the query vector (see Equation 2). Other measures are e.g., Jaccard and Dice coefficients [39].

$$\cos\theta = \frac{V_1 * V_2}{\| V_1 \| \| V_2 \|}$$

Equation 2 Vector Space Model Similarity Measure

Latent Semantic Indexing

In LSI, first a term-document matrix *A* and a query matrix *q* are constructed. Matrix *A* is constructed by gathering all content terms from all available documents. For each content term in our list, we go across the appropriate row and put the number of occurrences of that word in the column for any document where that word appears. If the word does not appear, we leave that column assigned to a numerical value of zero. The resulting grid displays everything that we know about our document collection. We can list all the content words in any given document by looking for X-es in the appropriate column, or we can find all the documents containing a certain content word by looking across the appropriate row. When this process ends matrix *A* is an MxN two-dimensional matrix where *M* represents terms across all documents and *N* represents the total number of available documents. The next step in LSI is decomposing this matrix using a technique called singular value decomposition which we have used GNU – GSL scientific library to calculate. We demonstrate this entire process with an example below:

Our "collection" consists of the following "documents"

D_1: Shipment of gold damaged in fire
D_2: Delivery of silver arrived in a silver truck
D_3: Shipment of gold arrived in a truck

For this example, and as in our research, we use the Term Count Model to score term weights and query weights, so local weights are defined as word occurrences. The following document indexing rules were also used:

- Stop words were not ignored
- Text was tokenized and lowercased
- No stemming was used
- Terms were CFted alphabetically

We will use LSI to rank these documents for the query "*gold silver truck*"

Step 1: Score Term weights and constructs the term-document matrix A

Terms		D_1	D_2	D_3		q
a		1	1	1		0
arrived		0	1	1		0
damaged		1	0	0		0
delivery		0	1	0		0
fire	A =	1	0	0	Q =	0
gold		1	0	1		1
in		1	1	1		0
of		1	1	1		0
shipment		1	0	1		0
silver		0	2	0		1
truck		0	1	1		1

Step 2: Using the Singular Value Decomposition algorithm we decompose Matrix *A* and find the U, S, and V matrices where $A = USV^T$; Using GNU – Gnu Scientific Library (GSL) we get:

$$U = \begin{bmatrix} -.4201 & .0748 & -.0460 \\ -.2995 & -.2001 & .4078 \\ -.1206 & .2749 & -.4538 \\ -.1576 & -.3046 & -.2006 \\ -.1206 & .2749 & -.4538 \\ -.2626 & .3794 & .1547 \\ -.4201 & .0748 & -.0460 \\ -.4201 & .0748 & -.0460 \\ -.2626 & .3794 & .1547 \\ -.3151 & -.3093 & -.4013 \\ -.2995 & -.2001 & .4078 \end{bmatrix}$$

$$S = \begin{bmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{bmatrix}$$

$$V = \begin{bmatrix} -.4945 & .6492 & -.5780 \\ -.6458 & -.7194 & .2556 \\ -.5817 & .2469 & .7750 \end{bmatrix}$$

$$V_2^T = \begin{bmatrix} -.4945 & .6458 & -.5817 \\ -.6492 & -.7194 & .2469 \\ -.5780 & -.2556 & .7750 \end{bmatrix}$$

Step 3: Implement a Rank 2 approximation by keeping the first columns of U and V and the first columns of rows of S. Because of the small document size we chose a Rank of 2; choosing an optimal Rank is an open question within the research community.

K = 2

$U_2 =$

-.4201	.0748
-.2995	-.2001
-.1206	.2749
-.1576	-.3046
-.1206	.2749
-.2626	.3794
-.4201	.0748
-.4201	.0748
-.2626	.3794
-.3151	-.3093
-.2995	-.2001

$S_2 =$

4.0989	0.0000
0.0000	2.3616

$V_2 =$

-.4945	.6492
-.6458	-.7194
-.5817	.2469

$V_2^T =$

-.4945	.6458	-.5817
-.6492	-.7194	.2469

Step 4: Find the new document vector coordinates in this reduced 2-dimensional space. Rows of V hold eigenvector values. These are the coordinates of individual document vectors, so,

D_1(-0.4945, 0.6492)
D_2(-0.6458,-0.7194)
D_3(-0.5817, 0.2469)

Step 5: Find the new query vector coordinates in the reduced 2-dimensional space.
$Q = Q^T * U_2 S_2^{-1}$

Note: These are the new coordinates of the query vector in two dimensions. Note how this matrix is now different from the original query matrix **Q** given in Step 1.

Q^T = 0 0 0 0 0 1 0 0 0 1 1

U_2

-.4201	.0748
-.2995	-.2001
-.1206	.2749
-.1576	-.3046
-.1206	.2749
-.2626	.3794
-.4201	.0748
-.4201	.0748
-.2626	.3794
-.3151	-.3093
-.2995	-.2001

S_2^{-1}

1/(4.0989)	0.0000
0.0000	1/(2.3616)

New Q:

Q = [-.2140, -0.1821]

Step 6: Rank documents in decreasing order of query-document cosine similarities.
General Formula:

$$Sim(Query, Doc) = \frac{Query * Doc}{|Query| * |Doc|}$$

Equation 3 Latent Semantic Indexing Similarity Measure

$$Sim(Query, Doc_1) = \frac{\overset{D_1}{(-.02190)}\overset{D_1}{(-.04945)} + (-.1821)(.6492)}{\sqrt{(-.2140)^2 * (-.01821)^2} * \sqrt{(-.4945)^2 * (.6492)^2}} \quad \text{<= eigenvalues}$$

=> **-0.0541**

$$Sim(Query, Doc_2) = \frac{\overset{D_2}{(-.02190)}\overset{D_2}{(-.6458)} + (-.1821)(.7194)}{\sqrt{(-.2140)^2 * (-.01821)^2} * \sqrt{(-.6458)^2 * (.7194)^2}} \quad \text{<= eigenvalues}$$

=> **.9910**

$$Sim(Query, Doc_3) = \frac{\overset{D_3}{(-.02190)}\overset{D_3}{(-.5817)} + (-.1821)(.2469)}{\sqrt{(-.2140)^2 * (-.01821)^2} * \sqrt{(-.5817)^2 * (.2469)^2}} \quad \text{<= eigenvalues}$$

=> .4478

Ranking Documents in Descending Order:

$$D_2 > D_3 > D_1$$

Latent Semantic Indexing (LSI) model was implemented in C with the aid of GNU (GNU) - GNU Scientific Library (GSL) v1.9 to aid in the computation of the singular value decompositions of a matrix.

After we supplied a similarity measurement to apply to the document collection the software generates a similarity matrix. This similarity matrix is then used for document

clustering/ text categorization. Below we have given a brief overview of clustering as well as the detailed means of how we approached clustering the documents.

Clustering

Clustering pertains to finding a *structure* in a collection of unlabeled data. A loose definition of clustering could be "the process of organizing objects into groups whose members are similar in some way". A *cluster* is therefore a collection of objects that are "similar" between them and are "dissimilar" to the objects belonging to other clusters. The similarity criterion is *distance*: two or more objects belong to the same cluster if they are "close" according to a given distance; this is called *distance-based clustering*. The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. But how to decide what constitutes a good clustering? It can be shown that there is no absolute "best" criterion, which would be independent of the final aim of the clustering. Consequently, it is the user, which must supply this criterion, in such a way that the result of the clustering will suit their needs. For our research we have chosen to use LINGPIPE clustering software – a Natural Langue Processing Suite that implements the two most common agglomerative clustering algorithms single-link clustering and complete-link clustering; in this thesis because of its ease of use we chose to utilize the hierarchical agglomerative single-link clustering algorithm; below is a further discussion of how clustering is approached within our research.

Hierarchical-Clustering

The most used clustering algorithms consist of K-Means, Fuzzy C-Means, Hierarchical, and Gaussian. Within our software system we have implemented a Hierarchical-clustering algorithm. In hierarchical clustering the data are not partitioned into

a particular cluster in a single step. Instead, a series of partitions takes place, which may run from a single cluster containing all objects to n clusters each containing a single object. Hierarchical Clustering is subdivided into agglomerative methods, which proceed by series of fusions of the n objects into groups, and divisive methods, which separate n objects successively into finer groupings. Agglomerative techniques are more commonly used. Hierarchical clustering may be represented by a two-dimensional diagram known as dendrogram, which illustrates the fusions or divisions made at each successive stage of analysis. An example of such a dendrogram is given below:

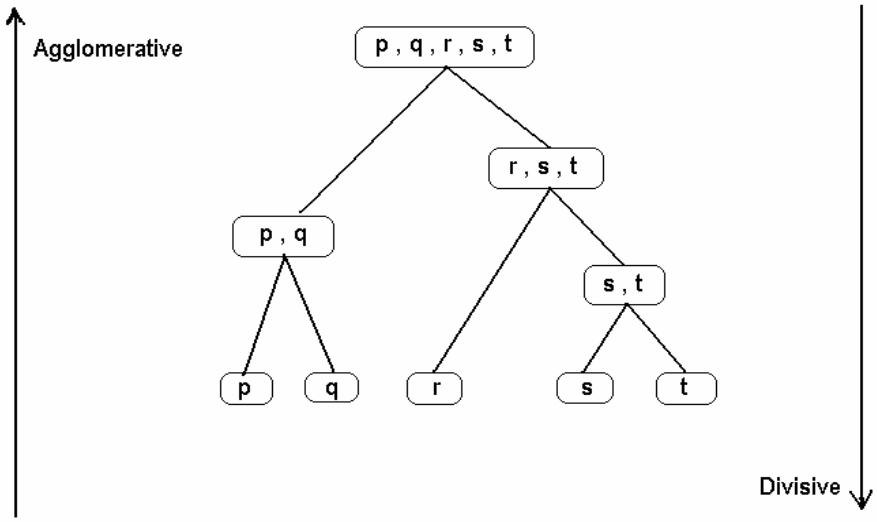

Figure 8: Agglomerative Bottom-Up Clustering Approach

Agglomerative Hierarchical Clustering

We have chosen the bottom-up approach of hierarchical-clustering, agglomerative clustering. An agglomerative hierarchical clustering procedure produces a series of partitions of the data, P_n, P_{n-1}, ……. P_1. The first P_n consists of n single object 'clusters', the last $P_{1,}$

consists of single group containing all n cases. At each particular stage the method joins together the two clusters, which are closest together (most similar). At the first stage, of course, this amounts to joining together the two objects that are closest together, since at the initial stage each cluster has one object. Differences between methods arise because of the different ways of defining distance (or similarity) between clusters such as Single-Link, Complete-Link, Average-Link, Ward's Method, and others; within our system we have adopted the Single-Link cluster distance measure.

Single-Link Distance Measure

Single link is one of the simplest hierarchical agglomerative clustering distance measure and is also known as the nearest neighbor technique. The defining feature of the method is that distance between groups is defined as the distance between the closest pair of objects, where only pairs consisting of one object from each group are considered.

In the single linkage method, ***D(r,s)*** is computed as

$$\boldsymbol{D(r,s)} = \text{Min } \{\, d(i,j) : \text{Where object } i \text{ is in cluster } \boldsymbol{r} \text{ and object } j \text{ is cluster } \boldsymbol{s} \,\}$$

Here the distance between every possible object pair *(i,j)* is computed, where object *i* is in cluster ***r*** and object *j* is in cluster ***s***. The minimum value of these distances is said to be the distance between clusters ***r*** and ***s.*** In other words, the distance between two clusters is given by the value of the shortest link between the clusters. At each stage of hierarchical clustering, the clusters ***r*** and ***s***, for which ***D(r,s)*** is minimum, are merged.

CHAPTER FOUR

Validation

The purpose of this chapter serves to describe the nature of the data that was tested and to provide a detailed description of our testing methods. For our experiments we have used a dataset provided by the University of California, Irvine - UCI Knowledge Discovery in Database (KDD) Archive. A dataset from this archive was chosen because it serves as a permanent repository of publicly accessible data sets for research in KDD and data mining. By using a centralized data repository this will help others to obtain the same data sets and replicate our experiments.

UCI Knowledge Discovery in Database (KDD) Dataset

The UCI KDD Archive provides several data sources with the purpose of aiding researchers in developing solutions for problems in Information Retrieval. For our experiments we have chosen to use the UCI KDD Text Characterization data source. The Text Characterization data source is comprised of four datasets: 21newsgroups, Reuters-21578, abstract1, and abstracts2. Due to the number and size of the articles in the Reuters-21578 dataset, this is the dataset we chose to use for testing.

Reuters-21578 Text Categorization Collection

The Reuters-21578 dataset is a collection of documents that appeared on Reuters newswire in 1987. The documents were assembled and indexed with categories this dataset consists of approximately 21,500 files covering 132 (possibly overlapping) categories; the file size per article can range from 12 – 900 words. Within this dataset the categories that were used in this experiment can be viewed in the below table.

Category	Files in Category
Heat	313
Gas	176
Interest	425
GNP	151
Nat-Gas	126
ACQ	2169
Wheat	288
Grain	570
Corn	223
Money-supply	135
Sugar	170
Coffee	142
Ship	279
Trade	510
Crude	546
Earn	3638
Money-fx	667

Table 1: Categories Used of Reuters-21578 Dataset

It is important to note that a single file may belong to multiple categories and that the categories that were not selected to be included in our experiments on average contained from 1 – 15 files.

Experiment Methods

To prove our claim that states using CFs to represent a document for Text-Categorization is more accurate then using a keyword based approach we have specifically designed two experiments. The first experiment consists of testing five corpuses varying in file size from fifty to one thousand documents; the articles contained within each corpus range from 12 – 900 words. We believe that the first experiment caters to keyword-based approaches simply due to natural language statistics that loosely claim the more words in a document the more likely that document contains repetitive word usage and higher the

likelihood of the document expressing one concept multiple ways. The second experiment consists of testing five corpuses varying in file size from fifty to one thousand documents; the articles contained in these documents vary in file size from 12 – 400 words. It is our belief that the classification results for the second experiments will favor using CFs because of the lower word content and greater need to disambiguate and map relationships among terms to classify documents. The below table (Table 2) will describe the testing data for each experiment.

Experiment 1	12-900 Words Per Article		
	Corpus 1		
		Size	50 Documents
		Categories	(25) House, (25) Money-fx
	Corpus 2		
		Size	100 Documents
		Categories	(50) House, (50) Money-fx
	Corpus 3		
		Size	200 Documents
		Categories	(50) Livestock, (50) oil, (50) corn, (50) bop
	Corpus 4		
		Size	500 Documents
		Categories	(250) Sugar, (250) corn
	Corpus 5		
		Size	1000 Documents
		Categories	(250)Sugar,(250)corn,(250)wheat,(250) Money-fx
	Corpus 6		
		Size	1000 Documents
		Categories	(100)Sugar, (100)corn, (100)wheat, (100) acq (100) Money-fx,(100)Livestock,(100)bop, (100)Oil, (100)crude,(100)ship

Table 2: Experiment 1 Dataset Breakdown

Experiment 2	12-400 Words Per Article		
	Corpus 1		
		Size	50 Documents
		Categories	(25) Oil, (25) Nat-Gas
	Corpus 2		
		Size	100 Documents
		Categories	(50) Coffee, (50) Sugar
	Corpus 3		
		Size	200 Documents
		Categories	(50) Grain, (50) Wheat, (50) Ship, (50) Crude
	Corpus 4		
		Size	500 Documents
		Categories	(250) Wheat (250) Grain
	Corpus 5		
		Size	1000 Documents
		Categories	(333) ACQ, (333) Wheat, (334) Crude
	Corpus 6		
		Size	1000 Documents
		Categories	(100)Sugar, (100)Grain, (100)wheat, (100) acq (100) Money-fx,(100)Coffee,(100)bop, (100)Oil, (100)crude,(100)ship

Table 3: Experiment 2 Dataset Breakdown

Procedure for testing CFs

In the case of each experiment we tested each corpus with the following procedure: First, we converted the documents in the corpus into the CF format. Secondly we used the CFs as input to each similarity measurement algorithm (SS, VSM, and LSI) to produce three unique similarity matrices – one similarity matrix for each algorithm that was applied. Thirdly, one at a time, we used as input each similarity matrix into LINGPIPE Clustering software that classified the results based on the hierarchical agglomerative single-link clustering method; we then evaluated the results of the text-categorization for the CF based approach.

Procedure for testing KEYWORDs

In the case of each experiment we tested each corpus with the following procedure: First, we removed all stop-words from all documents in the corpus so that they only contain KEYWORDS that add semantic content to the document. Secondly we used the KEYWORDS from each document as input to each similarity measurement algorithm (SS, VSM, and LSI) to produce three unique similarity matrices – one similarity matrix for each algorithm that was applied. Thirdly, one at a time, we used as input each similarity matrix into LINGPIPE Clustering software that classified the results based on the hierarchical agglomerative single-link clustering method; we then evaluated the results of the text-categorization for the KEYWORDs based approach.

Performance Evaluation Method

Although many studies used K-means clustering algorithm or its variants for text document clustering [23],[24],[32] K-means algorithm is not suitable for text document clustering using our CF-based similarity measurement because it does not make sense to calculate a mean similarity among a set of documents. Therefore, an agglomerative hierarchical clustering algorithm is used in our performance study.

Given a document corpus with N documents, each document initially belongs to its own individual cluster. We set the initial similarity threshold to be 1 and decrease the threshold with a small interval so that documents with similar semantics will be gradually merged into the same group. Since we already know the categories from which each document was obtained, the document clustering process stops when majority of documents from different categories fall into their respective clusters and further decreasing the threshold will result in two clusters containing documents primarily from two different

categories merged into one cluster. After the document clustering, we calculate the clustering accuracy as the number of documents correctly clustered into their categories divided by the total number of documents.

Besides clustering text documents based on our CF-based similarity measurement method, we also perform the document clustering using VSM and LSI as the document similarity measurement methods. For VSM, the cosine coefficients of the document vectors are used as the text document similarity measures. For LSI, we calculate the rank k approximation of term vector for each document and calculate their similarities using cosine coefficients of their term vectors. Then we use these similarity values to cluster the text documents. We repeat the same process under different k values and report the best clustering results for LSI; more details regarding the evaluation of our results is given in chapter 6.

CHAPTER FIVE

Results

The goal of the research is to determine if it is possible to classify more documents correctly using the Concept Forest/equivalent or by using the traditional models that are keyword-based. We used UCI-KDD Reuters-21579 as our test corpora.

We ran two experiments each consisting of five document corpuses containing fifty to one thousand documents each. The differing factor between the experiments were the number of words allowed per document in the corpuses; experiment one contained documents that contained greater than four hundred words and experiment two contained documents that contained four hundred words or less – for more information on the document data set see chapter 5. We define two terms below that will aid in comprehension of our result analysis; Cluster Accuracy and Cluster Breakage.

Accuracy

Accuracy: we define cluster accuracy to be the total number documents clustered correctly, over the total number of documents within the corpus.

Cluster Breakage

Cluster Breakage: when calculating the accuracy measure we need to consider cluster breakage. We define cluster breakage as the point where two clusters that do not share the same categorical identity are joined due to low document similarity throughout the corpus. If cluster breakage occurs, to determine the accuracy measure we look at a snapshot of the clusters before this event happens; when viewing this snapshot, it is important to notice that any document that does not belong to a cluster at this time is treated as an 'unclassified' document.

Experiment One – 400 Words or Greater

Comparing SS, VSM, & LSI (using Concept)

Focusing on Table 4: Results of Concept Clustering on Dataset with Articles Containing > 400 Words, it is shown that using an ontology-based approach using the Latent Semantic Indexing model results in better text-categorization on 4 out of 5 corpuses that were tested. The Vector Space Model performs just slightly better then the Semantic Similarity model scoring a higher accuracy percentage on 4 of 5 corpuses. Viewing these results in its entirety we can see that although clearly LSI is the better method to use, LSI does not produce a significant increase in the number of documents that were classified correctly. Each method SS, VSM, & LSI all return relatively similar results and return similar accuracy values. However, it is worthy to note that although LSI performs the better of the three measurements, on average the SS measurement misclassifies documents at a much lower rate (but unclassified documents at a much higher rate) then VSM and LSI when attempting to group documents into similar categories.

Sim. Measure	Corpus 50	Corpus 100	Corpus 200	Corpus 500	Corpus 1000 4 Categories	Corpus 1000 10 Categories
SS	37	61	110	351	537	384
VSM	39	62	117	342	548	433
LSI	(k=2) 25	(k=2) 82	(k=50) 127	(k=97)347	(k=275) 553	(k=123) 436

Table 4: Results of Concept Clustering on Dataset with Articles Containing > 400 Words

Comparing SS, VSM, & LSI (using Keywords)

In Table 5: Results of Keyword Clustering on Dataset with Articles Containing > 400 Words, it is shown that using a keyword-based approach using the Latent Semantic Indexing model results in better text-categorization on 5 out of 5 corpuses that were tested.

The Semantic Similarity measurement outperforms the Vector Space Model scoring a higher accuracy percentage on 5 of 5 corpuses; LSI and SS are the top performing algorithms using Keywords.

Sim. Measure	Corpus 50	Corpus 100	Corpus 200	Corpus 500	Corpus 1000 4 Categories	Corpus 1000 10 Categories
SS	39	79	125	366	563	425
VSM	35	50	118	348	552	224
LSI	(k=2) 44	(k=2) 82	(k=70) 131	(k=8) 398	(k=100) 616	(k=143) 413

Table 5: Results of Keyword Clustering on Dataset with Articles Containing > 400 Words

Comparing SS, VSM, LSI (Concept Vs. Keyword)

When comparing using the Concepts (see Table 4) against the Keywords (see Table 5) for corpuses that contain articles that are greater than 400 words, our research finds that the differences between Concepts and Keywords, in terms of SS, VSM, and LSI similarity measures are minimal. In this experiment using the Keywords for each similarity measure is slightly better but not significantly better then using the Concepts of the same similarity measures – resulting in higher accuracy values in each of the 5 corpuses tested in favor of a Keyword-based approach for articles whose length is greater then 400 words; this is not so for articles whose length is less then 400 words.

Experiment Two – 400 Words or Fewer

Comparing SS, VSM, & LSI (using Concept)

The second experiment focused on corpuses that contained 400 words or fewer. Our research results indicate that an ontology-based approach under this circumstance yields a very high accuracy percentage, classifying more documents correctly for each corpus using the Semantic similarity measure then using any other similarity measure. The VSM and LSI

models result in an average of a less then 43% correct classification rate for corpuses that contain articles less then 400 words. The results of this test are shown in Table 6: Results of Keyword Clustering on Dataset with Articles Containing <= 400 Words.

Sim. Measure	Corpus 50	Corpus 100	Corpus 200	Corpus 500	Corpus 1000 4 Categories	Corpus 1000 10 Categories
SS	37	80	107	340	616	502
VSM	25	50	50	250	334	210
LSI	(k=2) 27	(k=2) 62	(k=2) 50	(k=285) 256	(k=50) 401	(k=174) 330

Table 6: Results of Keyword Clustering on Dataset with Articles Containing <= 400 Words

Comparing SS, VSM, & LSI (using Keywords)

As can be seen in Table 7, using Keywords and the Semantic Similarity measurement provides the higher accuracy value and better classification results by a large margin. Latent Semantic Indexing has the second next best with an average accuracy of 54%, followed by VSM with an average accuracy of 41%.

Sim. Measure	Corpus 50	Corpus 100	Corpus 200	Corpus 500	Corpus 1000 10 Categories	Corpus 1000 10 Categories
SS	32	82	96	269	546	158
VSM	25	50	50	250	334	497
LSI	(k=2) 25	(k=2) 62	(k=2) 68	(k=75) 284	(k=100) 624	(k=227) 432

Table 7: Results of Keyword Clustering on Dataset with Articles Containing <= 400 Words

Comparing SS, VSM, LSI (Concept Vs. Keyword)

It is important to note that the VSM Keyword and VSM Concept results are identical for each corpus that was tested; this can be attributed to our algorithm in generating the CF. Because the articles contain a small number of words - in order to construct an ontology for the document there must exist relationships between the words – if a relationship cannot be

found then just those words are included in the ontology as is; so it is possible for an CF and the extracted keywords of a document to be the same, which is the case in this experiment.

From Table 6 and Table 7 using the Concepts with the Semantic Similarity measurement is the best combination in terms of generating a high number of correctly classified documents. The next best combination is Latent Semantic Indexing with Keywords as input.

Result Analysis

From the results of our research it has been shown that the idea of using a document's ontology to represent the document in place of its keywords, as conventionally is used, is a sound principle under certain conditions. The dominating condition with regard to our dataset and our testing methods was the length of the article i.e. the number of words occurring in the document. Our results indicate that as the average number of words in a document corpus increases the less of an impact that the ontology methods has so much so that the text-categorization results converge to that of a Vector Space Model or Latent Semantic Indexing Keyword-based approach. As of now research does not indicate to a series of evident disadvantages with using an ontology-based approach – only that under the condition that is mentioned above, the ontology-based approach may perform as good or with minor differences in results in favor of VSM or LSI.

Our research has proven that when the length of an article is small, the higher the importance of the remaining words in the documents to aid in word disambiguation and greater emphasis is placed upon resolving synonymy and polysemy issues. In a scenario as such as this an ontology-based approach that utilizes the Semantic similarity measurement for text-categorization proves to be the better method to use according to our research

results; simply put, the smaller the average length across a document corpus the more advantageous it is to focus on a concept-level representation of the articles within the corpus as opposed to focusing on the articles keywords.

Runtime Efficiency

Hardware Specification

We conducted our experimental studies on a DELL desktop computer equipped with a 1.0 GHz Intel Pentium IV processor and 512 MB RAM, running on the Red Hat Enterprise Linux Operating System.

Runtime Analysis

With the current hardware specifications the charts below displays the execution time for each experiment with regard to all corpuses that were tested. As can be seen in experiment one in Figure 9 and Figure 10 which represents corpuses that contain documents with four hundred words or greater we see that the method that produces the most correctly classified documents LSI/Keywords (LSI using the Keyword-based approach – Table 5) also has the worst execution time – seemingly showing exponential growth. For experiment two in Figure 11 and Figure 12 for documents that contain less then four hundred words the worst algorithm is LSI/Keywords – again LSI/Keywords shows exponential growth; the best case with the lowest runtime is SS/CF. SS/CF shows linear growth, and this also reflects the best method that results in the most correctly classified documents for documents that contain less then four hundred words.

So in terms of experiment one, as concerned with the trade-off of desiring high accuracy and low execution time, the algorithm of choice would be LSI/Keyword based – although this is the worst in regards to execution time, due to the extremely high accuracy

percentage the LSI/Keyword based approach justifies the execution cost; for experiment two the algorithm of choice would be SS/CF – this method has the highest accuracy among the tests and the lowest execution cost.

The fastest algorithm across all experiments is in using The Semantic Similarity measurement. In each case, using Keywords or SynsetID, the The Semantic Similarity measurement showed the quickest runtime in each experiment. The worst algorithm with our research in terms of runtime efficiency and across all experiments and corpuses is Latent Semantic Indexing.

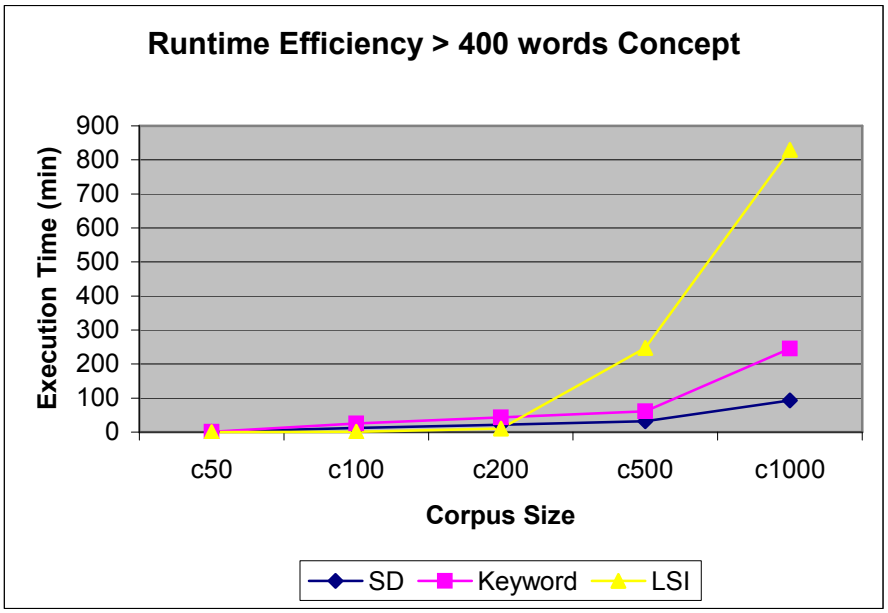

Figure 9: Runtime Efficiency Concept > 400 Words

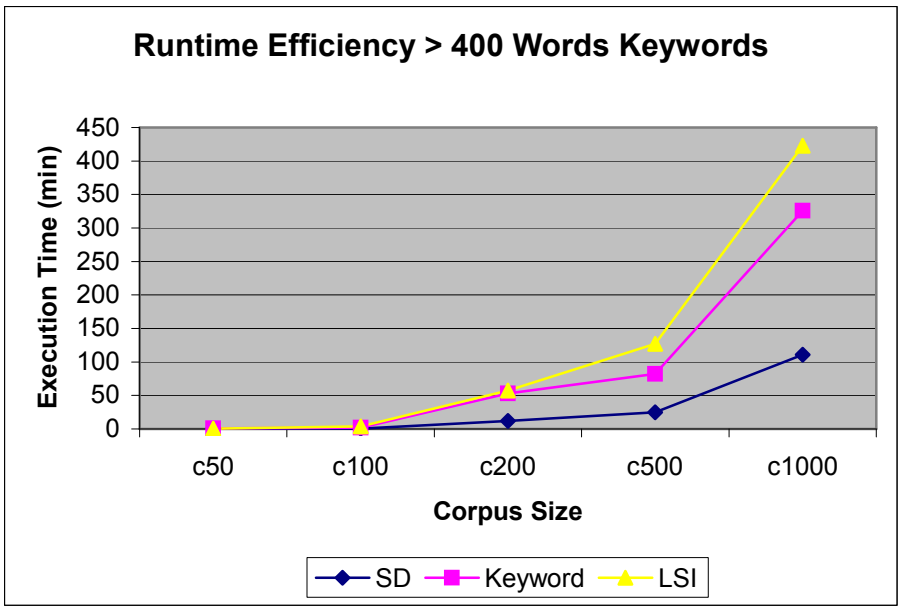

Figure 10: Runtime Efficiency Keywords > 400 Words

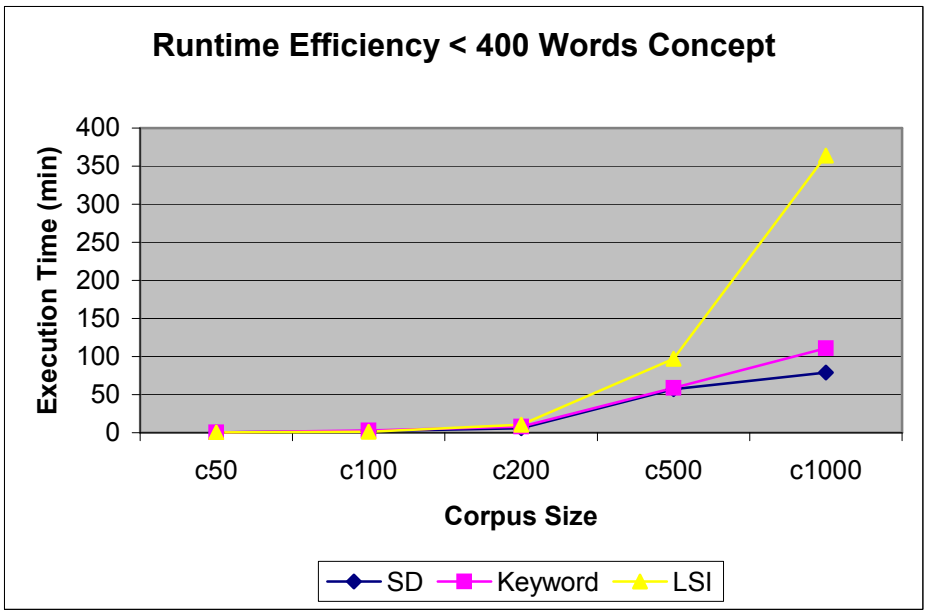

Figure 11: Runtime Efficiency Concepts < 400 Words

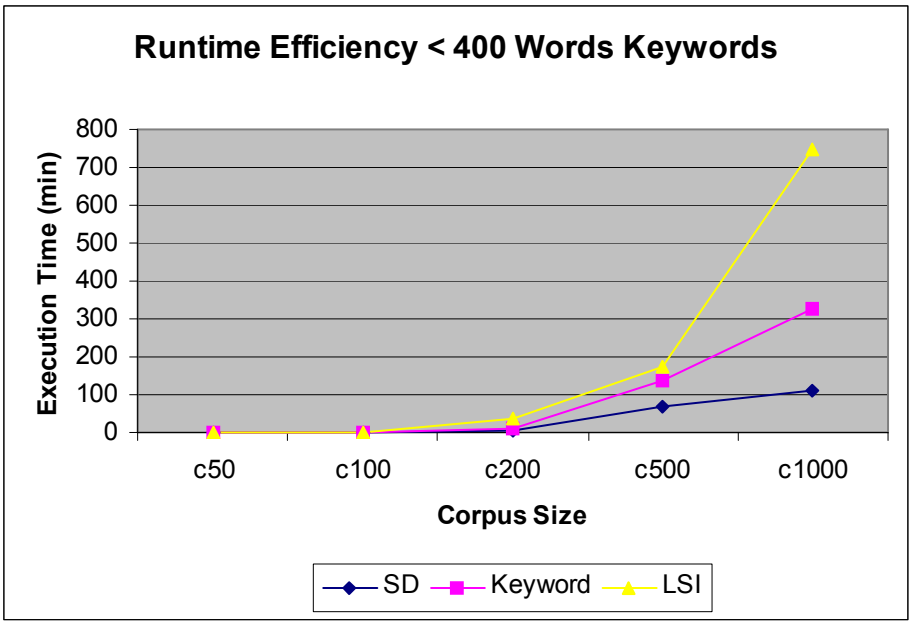

Figure 12: Runtime Efficiency Keywords < 400 Words

CHAPTER SIX

Disadvantages

Due to the nature of the evolving human language our proposed method relies completely on the maintainability of the WordNet lexical database. Because our algorithm heavily relies on WordNet to provide word disambiguation and to retrieve synsets to generate an equivalent Concept Forest for the document, WordNet becomes the least common denominator in determining the efficiency of ontology generation. So it can be stated that some of the disadvantages that arise from our algorithms to some degree can be attributed to the limits WordNet; those limits include representing common/uncommon Acronyms, phrases, and changing concepts in languages that requires a human component to update WordNet.

Through testing we have found the capability of WordNet to recognize acronyms is very limited to a small subset i.e. SUV, ASAP, ATM, AA, PS, etc. of the most popular acronyms. This is a disadvantage to the proposed system because it is an often occurrence for researchers and writers to define their own acronyms within context to the document – a form of shorthand notation; this happens often when an author refers to a sequence of words as an acronym to avoid unnecessary typing. WordNet does not provide data/synonyms for arbitrary acronyms – shorthand notations - that writers may create. A simple solution to this problem is presented in the future works section of this paper.

Another disadvantage of using the WordNet lexical database is its inability to identify common phrases i.e. car pool, wolf down, double check, etc. Through testing, this disadvantage has the most impact on a corpus whose documents represent abstract thoughts or 'comfortable conversation' (writing with no regard to sentence structure / using slang,

etc). This was found when testing the Reuters Newsfeed corpus, where the articles contained within were from a news forum of some CFt – hence the colloquial expressions; however this problem is in its least significance in documents that are well-written.

The last disadvantage that we have encountered pertains to the runtime efficiency of comparing large document ontologies. When generating a Concept Forest for a document that has a large set of (more the 500) keywords, the resulting ontology has the potential to grow exponentially – a document of 500 keywords may result in a Concept Forest of 1500 keywords and synonyms. When we tested a corpus that contained 100 documents with a large keyword threshold and generated the ontologies for each document in the corpus – we found that the *ontology* of this magnitude when compared with other *ontologies* of the same magnitude using (VSM, LSI, and set-based) the execution time of the algorithm exceeds 3 hours. In contrast when executing on the VSM or LSI algorithms on only the *keywords* the execution time is greatly reduced to approximately 15 minutes. This is also covered in more detail in the next section.

CHAPTER SEVEN

Conclusions & Future Work

In this paper, we propose a novel algorithm to extract a concept forest (CF) from a text document with the assistance of a natural language ontology, WordNet lexical database. Using concept forests to represent the semantics of text documents, we measure the semantic similarity of two text documents by simply comparing the term sets in their respective concept forests. This CF-based similarity measurement does not require analyzing the entire document corpus, an advantage over most existing document similarity measurement methods, including the popular VSM and LSI. Without needing to analyze the entire document corpus, our CF-based text document similarity measurement method is feasible for P2P environments where collecting the entire document corpus for analysis is impractical.

Our experimental studies also show that the CF-based text document similarity measurement method performs much better than both VSM and LSI methods when document sizes are relatively small. Furthermore, our CF-based document similarity measurement method is much more efficient regarding the total execution time used for corpus analysis and document clustering. Therefore, we believe the CF-based approach is a practical alternative to the existing keywords-based methods for information retrieval and text mining in text abstract databases, such as MEDLINE.

We are currently designing a graph-matching-based method to compare the similarity of two concept forests, hoping to provide a more sophisticated text document similarity measurement and improve the text document clustering accuracy. We are also implementing

a CF-based information retrieval system to effectively retrieve text abstracts from MEDLINE database.

REFERENCES

[1] GOOGLE: http://www.google.com

[2] G. Salton, A. Wong, and C. S. Yang (1975), A Vector Space Model for Automatic Indexing, *Communications of the ACM*, vol. 18, no. 11, pages 613–620.

[3] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6), 391-407.

[4] MEDLINE Fact Sheet, http://www.nlm.nih.gov/pubs/factsheets/medline.html

[5] Borgelt C, Nürnberger A. Fast fuzzy clustering of web page collections. Principles and Practice of Knowledge Discovery in Databases: Workshop on Statistical Approaches for Web Mining (SAWM), Pisa, Italy; 2004.

[6] Steinbach M, Karypis G, Kumara V. A comparison of document clustering techniques. *Knowledge Discovery in Databases: Workshop on Text Mining, Boston*, MA, August 20–23, 2000

[7] L. Khan, F. Luo, Ontology construction for information selection, *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence*, Crystal City, Virginia, 2002, pp. 122-127.

[8] Lee, C.S., Jian, Z.W. and Huang, L.K., A fuzzy ontology and its application to news summarization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. Volume 35, Issue 5. pp. 859-880.

[9] D. Elliman, J. Rafael, G. Pulido. Automatic derivation of on-line document ontology, *MERIT 2001, 15th European Conference on Object Oriented Programming*, Budapest, Hungary, 2001

[10] Lipika Dey, Ashish Chandra Rastogi, Sachin Kumar, Generating Concept Ontologies through Text Mining, *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI'06)*, pp. 23-32, 2006.

[11] Ong Siou Chin, N. Kulathuramaiyer, A. W. Yeo, Automatic Discovery of Concepts from Text, *IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006)*, pp. 1046-1049, December 2006

[12] Blaz Fortuna, Dunja Mladenic and Marko Grobelnik, *Semi-Automatic Construction of Topic Ontology*, Conference on Data Mining and Data Warehouses (SiKDD 2005) at at multiconference IS 2005.

[13] Navigli, R., Velardi, P. and Gangemi, A., Ontology learning and its application to automated terminology translation. IEE*E Intelligent Systems*. Volume 18, Issue 1. pp. 22-31.

[14] Sugumaran, V. and Storey, V.C., Ontologies for conceptual modeling: their creation, use, and management. *International Journal of Data and Knowledge Engineering*. Volume 42, Issue 3. pp. 251-271.

[15] V.W. Soo, C.Y. Lin, Ontology-based information retrieval in a multi-agent system for digital library, *Proceedings of the Sixth Conference on Artificial Intelligence and Applications*, Taiwan, 2001, pp. 241-246

[16] D.H. Widyantoro, J. Yen, A fuzzy ontology-based abstract search engine and its user studies, *Proceedings of the 10th IEEE International Conference on Fuzzy Systems*, Melbourne, Australia, 2001.

[17] A. Hotho, A. Madche, S. Staab, Ontology-based text clustering, *Proceedings of the IJCAI-2001 Workshop Text Learning: Beyond Supervision*, Seattle, USA, 2001, pp. 48-54.

[18] Christine Fellbaum (ed.), WordNet: An Electronic Lexical Database. The MIT Press, May 1998.

[19] S. Scott and S. Matwin. Text Classification using WordNet Hypernyms. *In S. Harabagiu, editor, Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, pages 38--44. Association for Computational Linguistics, Somerset, New Jersey, 1998.

[20] D. Koller, and M. Sahami, Hierarchically classifying documents using very few words, *Proceedings of the 14th international Conference on Machine Learning* ECML98, 1998.

[21] D. M. P. Kushal Dave, Steve Lawrence. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. *In Proceedings of the Twelfth International World Wide Web Conference, WWW2003*, Budapest, Hungary, May 20-24, pp. 519-528, 2003.

[22] De Luca EW, Nrnberger A. Ontology-based semantic online classification of documents: Supporting users in searching the web. *Proc of European Symp. on Intelligent Technologies* (EUNITE 2004), Aachen; 2004

[23] A. Kehagias, V. Petridis, V.G. Kaburlasos, and P. Fragkou, "A comparison of word- and sense-based text categorization using several classification algorithms," *Journal of Intelligent Information Systems*, 21(3), 2001.

[24] A. Hotho, S. Staab, and G. Stumme. Ontologies improve text document clustering. *In Proceedings of the IEEE International Conference on Data Mining*, pages 541-544, 2003.

[25] Dimitrios Mavroeidis et al., Word Sense Disambiguation for Exploiting Hierarchical Thesauri in Text Classification, A. Jorge et al. (Eds.): *PKDD 2005, LNAI 3721*, pp. 181–192, 2005.

[26] Youjin Chung and Jong-Hyeok Lee, Practical Word-Sense Disambiguation Using Co-occurring Concept Codes, *Machine Translation* (2005) 19: 59–82.

[27] Ernesto William De Luca, Andreas Nürnberger: Using clustering methods to improve ontology-based query term disambiguation. *Int. J. Intell. Syst.* 21(7): 693-709 (2006)

[28] Ying Liu, Peter Scheuermann, Xingsen Li, and Xingquan Zhu, Using WordNet to Disambiguate Word Senses for Text Classification, Y. Shi et al. (Eds.): *ICCS 2007, Part III, LNCS 4489*, pp. 780–788, 2007.

[29] J. Sedding and D. Kazakov. WordNet-based Text Document Clustering. *In Proc. of the Third Workshop on Robust Methods in Analysis of Natural Language Data (ROMAND)*, pp.104-113, Geneva, 2004.

[30] Thomas de Simone and Dimitar Kazakov. Using WordNet Similarity and Antonymy Relations to Aid Document Retrieval. *Recent Advances in Natural Language Processing (RANLP 2005)*, 21-23 September 2005, Borovets, Bulgaria.

[31] Chihli Hung, Stefan Wermter and Peter Smith, Hybrid Neural Document Clustering Using Guided Self-Organization and WordNet, *IEEE Intelligent Systems*, Vol. 19, No. 2, pp. 68-77, 2004.

[32] L. Jing, L. Zhou, M. Ng and J. Huang, Ontology-based Distance Measure for Text Clustering, SIAM Text Mining 2006 Workshop.

[33] Marta Sabou, Learning Web Service Ontologies: an Automatic Extraction Method and its Evaluation, Ontology Learning and Population (Editors: P.Buitelaar, P. Cimiano, B. Magnini), IOS Press, 2005

[34] Reuters-21578 Text Categorization Collection, http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

[35] Miller, George A., Christiane Fellbaum, Katherine J. Miller. *Five Papers on WordNet* August, 1993, retrieved May 4, 2005

[36] Luhn, H. P. *The Automatic Creation of Literature Abstracts*. IBM Journal of Research and Development 2 (2), p. 159-165 and 317, April 1958.

[37] Salton, Gerard. *Automatic Text Processing.* Addison-Wesley Publishing Company, 1988

[38] Rosario, Barbara. *Latent Semantic Indexing: An Overview Spring*, 2000 INFOSYS 240

[39] *Vector Space Model – An Overview*
http://isp.imm.dtu.dk/thor/projects/multimedia/textmining/node5.html

[40] Jing, Liping, Lixin Zhou, Michael K. Ng, Joshua Z. Huang. *Ontology-based Distance Measure for Text Clustering*

[41] A. Hotho, S. Staab, G. Stumme. *WordNet improves Text Document Clustering*, Proc. Of the Semantic Web Workshop at SIGIR-2003-26 Conference, 2003

[42] Blaz Fortuna, Dunja Mladenic, Marko Grobelnik. *Semi-Automatic Construction of Topic Ontology*, Department of Knowledge Technologies

[43] A. Thotho, S. Staab, G. Stumme. *Text Clustering Based on Background Knowledge*, Technical

[44] Ong Siou Chin, Narayanan Kulathuramaiyer, Alvin W. Yeo. *Automatic Discovery of Concepts from Text*, University of Malaysia – Computer Science

[45] J. Gonzalo, F. Verdejo, I. Chugur, J. Cigarran. *Indexing with WordNet Synsets can improve text retrieval*, In Proceedings ACL/COLING Workshop on Usage of WordNet for Natural Language Processing, 1998

[46] A. El-Hamdouchi, P. Willett. *Hierarchial Document Clustering Using Ward's Method*, Shrefield University

[47] Latent Semantic Indexing http://www.cs.utk.edu/~lsi/