Natural Language Document and Event Association Using Stochastic Petri Net Modeling

Michael Thomas Mills
Wright State University

Follow this and additional works at: https://corescholar.libraries.wright.edu/etd_all

Part of the Computer Engineering Commons, and the Computer Sciences Commons

Repository Citation
https://corescholar.libraries.wright.edu/etd_all/789

This Dissertation is brought to you for free and open access by the Theses and Dissertations at CORE Scholar. It has been accepted for inclusion in Browse all Theses and Dissertations by an authorized administrator of CORE Scholar. For more information, please contact library-corescholar@wright.edu.
NATURAL LANGUAGE UNDERSTANDING FOR DOCUMENT AND EVENT ASSOCIATION USING STOCHASTIC PETRI NET MODELING

A dissertation submitted in partial fulfillment of the
Requirements for the degree of
Doctor of Philosophy

By

MICHAEL THOMAS MILLS
M.S., Wright State University, 1979
B.E.E., The Ohio State University, 1970

2013
Wright State University

Nikolaos G. Bourbakis, Ph.D.
Dissertation Director

Arthur A. Goshtasby, Ph.D.
Director, Computer Science and Engineering Ph.D. Program

R. William Ayres, Ph.D.
Interim Dean, Graduate School

Committee on Final Examination

Nikolaos G. Bourbakis, Ph.D.

Krishnaprasad Thirunarayan, Ph.D.

Soon Chung, Ph.D.

Arnab K. Shaw, Ph.D., EE Department

Michael Talbert, Ph.D.
AFRL/RYA, WPAFB
ACKNOWLEDGEMENT

I would like to thank my advisor Dr. Nikolaos Bourbakis for giving me the opportunity to be creative and efficient. He encouraged me at difficult times and with tough challenges, but also his criticism helped me to complete this work.

I would also like to thank the members of my committee Dr. Soon Chung, Dr. Arnab Shaw, and Dr. T.K. Prasad and Dr. Mike Talbert for their important suggestions and support.
ABSTRACT


The purpose of this research is to design and implement a new methodology that captures the natural language understanding of events from English natural language text and model it using Stochastic Petri Nets. To establish a baseline of recent natural language processing (NLP) and understanding (NLU) research, two surveys are presented. One is a general survey in NLP and NLU methodologies for processing multi-documents. It summarizes and presents methodologies in terms of their features, capabilities, and maturity. The second survey focuses on graph-based methods for NL text processing and understanding and analyzes them in terms of their functional descriptions, capabilities and maturities. In recent years, NLP/NLU researchers have narrowed their domain to graph methodologies due to improved efficiency over older methods. Thus, to accomplish our goal, we firstly implemented a NL text to graph conversion method. This method extracts events in terms of their agents, actions, and patients from subject nouns, verbs, and object nouns within each phrase and sentence of a text and produces a graph consisting of nodes representing nouns and verbs and edges representing their relations. A significant effort went into handling complex sentences consisting of multiple phrases, active and passive sentences, and multiple agents, actions, and patients. The graph provides a baseline implementation, which we could relate to other graph methodologies and provide a structured approach to NLP and NLU from text. Next, we embedded a new NL text-graphs to Stochastic Petri Net (SPN) graph conversion methodology into our model to represent events associated with NL text. SPN graphs provide not only a structured representation that graphs provide, but also other capabilities, such as representing and adjusting timing using its transition components, constraining flow with its inhibiting places, stochastic behavior of its markings, and color markings [89, 90]. We use these added capabilities from SPN modeling to capture new NLU capabilities of events from NL text. We demonstrated sentence disambiguation of events.
# TABLE OF CONTENTS

I. INTRODUCTION ........................................................................................................... 1
II. GENERAL SURVEY ........................................................................................................ 5

A COMPARATIVE SURVEY ON NLP/U METHODOLOGIES ........................................ 5
FOR PROCESSING MULTI-DOCUMENTS [43] ................................................................. 5

2.1 Introduction ................................................................................................................. 5

2.2 Methodologies and Their Features ............................................................................. 7

2.2.1 Text Relationship Map with Latent Semantic Analysis (LSA) ......................... 7
2.2.2 Extraction methods for text summarization ......................................................... 8
2.2.3 Cluster based summarization .............................................................................. 8
2.2.4 Chaining lexically to formulate semantic relations ........................................... 9
2.2.5 Stochastic Petri-Net (SPN) representations for NLU ....................................... 11
2.2.6 Building concept representations from text ....................................................... 11
2.2.7 Learning ontology from text ............................................................................. 13
2.2.8 Redundancy removal for document synthesis ................................................... 18
2.2.9 Generated text semantically meaningful through coherence and logical order 20
2.2.10 Text generation methodologies ..................................................................... 22
2.2.11 Document processing and understanding ...................................................... 22
2.2.12 Other relevant methodologies ....................................................................... 24

2.3 Comparative Analysis of the Methodologies ............................................................ 26

2.3.1 Extraction approaches ...................................................................................... 26
2.3.2 Lexical Chaining Approaches ......................................................................... 28
2.3.3 Building concepts approaches ........................................................................ 29
2.3.4 Ontology approaches ...................................................................................... 31
2.3.5 Hierarchical approaches to summarizing large documents ............................. 33
2.3.6 Making generated text semantically meaningful through coherence and local order 35
2.3.7 Text generation methodologies .................................................................................. 35
2.3.8 Document processing / (low level) understanding .................................................. 35
2.4 Comparative Table of Methodologies ........................................................................ 35
2.5 Comparative Table of Maturity .................................................................................... 37
2.6 Conclusions .................................................................................................................. 40

III. GRAPH BASED SURVEY .......................................................................................... 42
3.1 INTRODUCTION ......................................................................................................... 42
3.2 Methodologies Description .......................................................................................... 44

3.2.1 Classification by clustering ....................................................................................... 44
3.2.1.1 In method 1, Clustering is used to induce (or find) topically related senses from
a graph of nodes representing nouns. Nouns paired by edges are weighted by the number
of paragraphs, which contain co-occurrences of the nouns [44]. .................................... 45
3.2.1.2 In method 2, clustering finds sets of similar words and hierarchies of concepts
by partitioning weighed undirected graphs [45]. ............................................................ 46
3.2.1.3 In method 3 (Moens, et al, 2005) performs multi-document summarization by
clustering objects representing similar sentences [46]. .................................................. 47
3.2.1.4 In method 4, clustering groups together similar documents [47]. ....................... 48
3.2.1.5 Method 5 (Nomoto and Matsumoto, 2003) finds topical clusters, finds the main
sentence to represent the cluster, and finds the concept diversity in text to reduce
redundancy of its generated summary [48]. ...................................................................... 48
3.2.1.6 Method 13 (Guo, Y., and Stylios, 2005) creates clusters to build conce-
pt representations from text [56]. See paragraph 3.2.3.5 (Event Indexing) for further
description .......................................................................................................................... 49
3.2.1.7 Method 17 (Cimiano, et.al., 2005) Clustering identifies themes of documents
from closely related concepts by using a meta-thesaurus, such as UMLS, and clustering
similar sentences [60]. Although Method 17 uses clusters, discussion is deferred to
paragraph 3.2.4.4 (since its main contribution is for concepts, relations, and semantics.)
............................................................................................................................................ 49
3.2.1.8 Method 20 (Pado and Lapata, 2007) (Discussed in paragraph 3.2.4.7) finds
diverse topical clusters from text, identifies the most important sentence for a topic
cluster, and uses that sentence to represent the cluster [63]. ........................................... 49
3.2.2 Methods 3, 6, 7, 8, and 9 by [46, 49, 50, 51, 52] use similarity measure to measure
similarities of meaning (i.e. word sense) and to merge clusters of similar concepts. ....... 49
3.2.2.1 Method 3 (discussed in paragraph 3.2.1.4 above) measures similarity between the most representative sentence object of a cluster and the rest of the objects (sentences) of each cluster to detect when a threshold is exceeded [46].

3.2.2.2 Method 6 (Moschitti and Zanzotto, 2008) uses machine learning to recognize textual entailment using tree pairs representing syntax in graph form.

3.2.2.3 Method 7 (Rao, et al, 2008) provides information on distance measures between nodes in a graph.

Such distances are useful in measuring weighed relationships between vertices in a graph and what they represent such as semantic similarities, intended sense (or meaning), etc.

The paper evaluates random walk algorithms, their performance over shortest path algorithms, and describes a new commute time measure. A pseudo inverse of the Laplacian is used to drive estimates for commute times between nodes on a graph. The method uses singular value decomposition to discard least significant eigenvectors. A random walk algorithm captures graph connectivity as well as path lengths.

3.2.2.4 Method 8 (Ambwani and Davis, 2010) uses weighed, directed graphs to model the range of influence of terms within a document and uses context to find the semantic relatedness of terms (within a sentence).

3.2.2.5 Method [9] (Minkov and Cohen, 2011) learns word semantic similarity measures from traversing a graph representation of a corpus of parsed text to extract word synonyms from text.

The method learns different graph walk models while traversing the graph and uses these models to extract word synonyms. It constrains paths of its random walk to specific word types that match the type of the word of interest. It uses history of its random walk to estimate edge weights to constrain its path, thus pruning paths with probabilities lower than a predetermined threshold.

3.2.3 Methods 1, 10, 11, 12, 13, 14, and 15 by [44, 53, 54, 55, 56, 57, 58] use nouns and verbs, including phrases and modifiers, to capture information from text and represent it in graphs.

3.2.3.1 Nouns, represented as nodes with weighted edges to other nodes representing various senses (or meanings), help determine the intended meanings of words [44]. See 3.2.1.1 for parts a through c of method 1.

3.2.3.2 Subjects, verbs, objects, and verb modifiers are used in graphical relations to match given questions to corresponding answers derived from the text.

3.2.3.3 A graph matching algorithm uses verbs with their arguments and nouns with their modifiers to build syntactic-semantic graphs. These verbs and nouns come from sentences, clauses, and phrases. Semantic relations are formed from clauses and noun modifiers and relation classes, including causal, conjunctive, participant, spatial,
temporal, and quality. The most important verb of a sentence is used as a head-word of
the sentence [54]. .......................................................... 54

3.2.3.4 Method 11 (Nastase. et al, 2006) is a graph matching algorithm that extracts
pairs of syntactic units from text and assigns a semantic relation to each pair. ............ 54

It incrementally learns from previous pairs and relations it has assigned plus some user
feedback. It builds a syntactic-semantic graph as it assigns semantic relations. (See
discussion.) This methodology matches graphs starting with a small amount of encoded
(manually labeled) knowledge from a dictionary. It extracts pairs of text units (such as
clauses, a verb and its arguments, a noun and each of its modifiers). It relies on
previously processed examples (stored as graphs) to find the most appropriate relation for
an extracted pair from text. It incrementally learns from semantic analysis. ............... 54

3.2.3.5 An In-Degree algorithm uses nouns, verbs, adjectives, and adverbs to find the
most probable sense (meaning) of words based on their context. Hyponyms found using
similarity measures are included in determining word sense [55]............................. 55

3.2.3.6 Event indexing uses actors (based on current events), time of day (for present
and previous events), causal (past event effects on current events), intention, and goals of
actors (from previous events) to form relations between concepts. Causal connections
consist of subject to object relations, formed from chains of verbs and nouns [56]....... 56

3.2.3.7 Method 14 by [57] (Kozarevak, Z, 2012) learns terms that express cause-effect
relations from text. It uses graph based metrics to re-rank extracted information and filter
and delete erroneous terms to increase accuracy of cause-effect relations............... 58

Method 15 by [58] (Liu, H., et.al., 2011) learns event rules by identifying context
dependencies from parsed annotated text. It extracts events by matching subgraphs of
sentences as it searches for a graph of an event rule. For event recognition, this method
searches for a subgraph that is isomorphic to dependency representations of previously
learned event rules. The dependency of event rules is implemented as a labeled directed
graph. The union of shortest dependency paths, each representing a training sentence,
produces a graph representation of an event [58]. ....................................................... 58

Performance: This method yields 41% to 52% F-score in detecting and identifying
biological events [58]. ......................................................................................... 58

3.2.4 Concept representations and their semantic relations, consisting of graphs and the
methods that create and use them, are important steps toward natural language
understanding (NLU). Methods 16, 19, 20, and 21 by [59, 62, 63, and 21] include the
concept level of representation and manipulation as well as other NLP methods 8, 12, 17
and 18 by [ 51, 55, 60 and 61] that serve as interim steps toward the creation of this level
of representation............................................................................................... 59
3.2.4.1 Method 8 by [51] (Ambwani and Davis, 2010) measures semantic relatedness using connection strength in the form of weights between node representations of terms. The neighborhood of terms (within a connection strength threshold) increases the semantic relatedness of other terms. This method is discussed in paragraph 3.2.2.4 (in similarity measure.)

3.2.4.2 Method 12 by [55] (Guo and Diab, 2009) incrementally learns semantic relations from semantic analysis and assigns semantic relations to pairs using a graph-matching algorithm. Its relation classes include causal, conjunctive, participant, spatial, temporal, and quality. The discussion is deferred to paragraph 3.4.3.4 (in subject-verb-object structures).

3.2.4.3 Method 16 by [59] (Morales, et al, 2008) maps text to concepts using ontology for concept extraction from text and uses (biomedical) domain knowledge from a metathesaurus to find synonym and “is-a” relations to summarize documents.

3.2.4.4 Method 17 by [60] (Cimiano, et al, 2005) acquires concept hierarchies using formal concept analysis that creates concept clusters.

3.2.4.5 Method 18 by [61] (Yeh, E., et.al., 2009) uses a random walk algorithm over a graph representation while measuring semantic relatedness of corresponding text.

3.2.4.6 Method 19 by [62] (Ye, et al, 2007) creates a concept lattice that indexes local topics within a topic hierarchy. Concepts in the form of topics are represented by nodes in the lattice.

3.2.4.7 Method 20 by [63] (Pado and Lapata, 2007) constructs semantic space models with annotated dependency relations and builds semantic context from a dependency graph, which maps dependency paths to words.

3.2.4.8 Method 21 by [64] (Rajaraman and Tan, 2002) generates a concept frame graph for knowledge discovery by constructing concepts and relations and uses a co-reference resolution algorithm to extract noun and verb clauses and phrases to generate synonym sets and relation parameters.

3.2.5 Other methods 22, 23, 24, 25, 26, 27 by [65, 66, 67, 68, 69. 70] provide graph based ranking of data in labels, find closest sense of words in labels, detect new information, generate grammars for a given text, resolve references to events, and generate graphs from word co-occurrence in text.

3.2.5.1 Method 22 by [65] (Mihalcea, 2005) uses a ranking algorithm to label the correct meaning of each word represented by a node while random walking over a graph. It consists of labeling sequence of data (words) by analyzing label dependencies. It is a sequence data labeling method applied to an all-words word sense disambiguation (WSD) problem by labeling the most likely sense (or meaning) of each word while random walking over the graph representing the text. An iterative graph based ranking.
of the labels uses a graph based ranking algorithm called Page-Rank (Brin and Page, 1998) [65].

3.2.5.2 Method 23 by [66] (Agirre, E., et al, 2006) uses unsupervised learning by training on a large sample of text (corpora) to find the sense (meaning) of words for labeling nodes of a graph. 64

3.2.5.3 Method 24 by [67] (Gamon, 2006) detects novelty (text, containing new information, compared to what is already found) at the sentence level by analyzing feature sets from graph representations of sentences. 65

3.2.5.4 Method 25 by [68] (Muresan, 2008) uses learning to map text to graph based meaning representation based on grammar induction. 66

3.2.5.5 Method 26 by [69] (Herdagdelen, et al, 2009) compairs two approaches for finding semantic relatedness. 67

3.2.5.6 Method 27 by [70] (Chen and Ji, 2009) resolves references within two or more sentences to events. 67

3.2.5.7 Method 28 by [71] (Biemann, 2010) constructs a word co-occurrence graph for each target (important) word using co-occurrence statistics from a large set of text (20 million sentences from the New York Times corpus). 68

Unsupervised learning from the corpus provides graph clusters with features that provide the kind of information normally used in systems with supervised learning. Representative graphs with different parameters are generated from the corpus. The target node in the graph (representing the target word) is directly connected to each node that represents a word significantly co-occurring with the target word in a sentence. Weights for each edge is calculated from the log-likelihood of the co-occurrence of the connected words (derived from relative frequency gathered from the corpus text). Graph parameters include size (number of co-occurrences or connections) and density. The Chinese Whispers algorithm (see M6) clusters the neighborhood of the graph by partitioning the nodes and representing each cluster as a word sense, thus providing word sense induction [71]. 68

3.2.5.8 Method 29 by [72] (Rao and Yarowsky, 2009) uses a map reduce algorithm in a semi-supervised (partially labeled) graph representation of natural language text for label propagation. 69

3.3 Functional components of graph-based methodologies ......................................................... 70
3.4 Definitions ............................................................................................................................... 74
3.5 Maturity Tables and Charts ................................................................................................. 76
3.6 Conclusions ............................................................................................................................ 80
3.6.1 Major functional components ............................................................................................ 81
3.6.2 Clustering: ........................................................................................................................... 81
Clustering can filter out common (relatively unimportant) nouns from a graph, leaving more pronounce nouns to determine the sense of words. The result is a more efficient, compressed graph representation of text. It can also calculate similarity measures for detecting similar documents and reduce redundancy at the concept level.

3.6.3 Similarity measure: ................................................................. 81
3.6.4 Nouns and verbs plus their phrases and modifiers: ....................... 81
3.6.5 Concept Representations are created in the following ways: ............. 82
3.6.6 Other methods not fitting in a group: ....................................... 82

Provide graph based ranking of data in labels, find closest sense of words in labels, detect new information, generate grammars for a given text, resolve references to events, and generate graphs from word co-occurrence in text............................................. 82

3.7 Maturity....................................................................................... 82

3.7.1 This survey has provided a summarization of graphical NLP and NLU methods (labeled as 1 through 26) with an analysis of estimated accuracy, coverage, performance, scalability and estimated maturity based on the information available in the referenced papers. 83

The tables of values show that this area of research, on the average, needs further improvement before they meet the desires of developers and users of products from this research area. Although significant improvements have been made over the years, a majority of these methods are still in progress. Some of the publications reviewed for this survey showed that systems have been developed from this research area and are used mostly in the medical field, but more mature systems are needed. Very little on maturity has been reported in these research areas. Maturity values calculated in this survey were estimated based on information provided in the papers. As this research area continues to progress, more efforts and published results on maturity should help system developers and users meet their needs in various NLP and NLU applications. ............. 83

IV. NATURAL LANGUAGE TEXT TO GRAPH REPRESENTATION ............... 84
4.1 Implementation details.................................................................... 84
4.2 Experimental Results of Graph Program ........................................ 93

V. GRAPH TO STOCHASTIC PETRI NET CONVERSION .................. 100
5.1 SPN Visualization....................................................................... 100
5.2 SPN Synthesis........................................................................... 102
5.2.2 Methods for gathering information needed for SPN synthesis: ........ 104
5.2.2 Discussion of tasks with approaches to solutions: ....................... 105
5.2.2.1 An automated means of detecting ambiguous sentences: ......... 105
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>76</td>
</tr>
<tr>
<td>3.2</td>
<td>76</td>
</tr>
<tr>
<td>3.3</td>
<td>78</td>
</tr>
<tr>
<td>3.4</td>
<td>78</td>
</tr>
<tr>
<td>4.1</td>
<td>82</td>
</tr>
<tr>
<td>4.2</td>
<td>83</td>
</tr>
<tr>
<td>4.3</td>
<td>86</td>
</tr>
<tr>
<td>4.4</td>
<td>86</td>
</tr>
<tr>
<td>4.5</td>
<td>87</td>
</tr>
<tr>
<td>4.6</td>
<td>87</td>
</tr>
<tr>
<td>4.7</td>
<td>88</td>
</tr>
<tr>
<td>4.8</td>
<td>90</td>
</tr>
<tr>
<td>4.9</td>
<td>91</td>
</tr>
<tr>
<td>4.10</td>
<td>91</td>
</tr>
<tr>
<td>4.11</td>
<td>92</td>
</tr>
<tr>
<td>4.12</td>
<td>92</td>
</tr>
<tr>
<td>4.13</td>
<td>93</td>
</tr>
<tr>
<td>4.14</td>
<td>93</td>
</tr>
<tr>
<td>4.15</td>
<td>94</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>4.16</td>
<td>Example of very long text (approximately two pages)</td>
</tr>
<tr>
<td>5.1</td>
<td>Visualization of graph to SPN conversion</td>
</tr>
<tr>
<td>5.2</td>
<td>Sample code using VertexShapeTransformer</td>
</tr>
<tr>
<td>5.3</td>
<td>Illustration of SPN synthesis</td>
</tr>
<tr>
<td>5.4</td>
<td>Graph to SPN conversion of medical example (two sentences)</td>
</tr>
<tr>
<td>5.5</td>
<td>Graph to SPN conversion of resistor example (two sentences)</td>
</tr>
<tr>
<td>5.6</td>
<td>Graph to SPN conversion of three phrases example</td>
</tr>
<tr>
<td>5.7</td>
<td>Graph to SPN conversion of complex sentence with phrases example</td>
</tr>
<tr>
<td>5.8</td>
<td>Graph to SPN conversion of ten sentences example</td>
</tr>
<tr>
<td>5.9</td>
<td>Graph to SPN conversion of long sentence example</td>
</tr>
<tr>
<td>5.10</td>
<td>Graph to SPN conversion of long text (image description) example</td>
</tr>
<tr>
<td>5.11</td>
<td>Graph to SPN conversion of long text (approximately two pages) example</td>
</tr>
<tr>
<td>6.1</td>
<td>Using state machine approach to modeling SPNs</td>
</tr>
<tr>
<td>6.2</td>
<td>The SPN Synthesis Process for finding impact of agents and patients in context</td>
</tr>
<tr>
<td>6.3</td>
<td>A step by step process in transforming a text file to an SPN synthesis file</td>
</tr>
<tr>
<td>6.4</td>
<td>SPN Synthesis Example of detecting and selecting interpretations from context</td>
</tr>
<tr>
<td>6.5</td>
<td>Diagram of the compression process</td>
</tr>
<tr>
<td>6.6</td>
<td>Pseudo code of the compression process</td>
</tr>
<tr>
<td>6.7</td>
<td>Steps in associating two different image files</td>
</tr>
</tbody>
</table>
6.8 Example 1 of associating two different image description files .................. 123

6.9 Example 2 of associating two different image description files .................. 127
### LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Feature definitions for extraction approaches</td>
<td>25</td>
</tr>
<tr>
<td>2.2 Comparing extraction approaches</td>
<td>25</td>
</tr>
<tr>
<td>2.3 Feature definitions for lexical chaining</td>
<td>26</td>
</tr>
<tr>
<td>2.4 Lexical chaining approaches</td>
<td>27</td>
</tr>
<tr>
<td>2.5 Performance (Accuracy)</td>
<td>27</td>
</tr>
<tr>
<td>2.6 Feature definitions for building concept approaches</td>
<td>28</td>
</tr>
<tr>
<td>2.7 Building concept approaches</td>
<td>28</td>
</tr>
<tr>
<td>2.8 Features for building ontology from text and NLP representation schemes</td>
<td>30</td>
</tr>
<tr>
<td>2.9 Comparing approaches to building ontology from text and NLP representation schemes</td>
<td>31</td>
</tr>
<tr>
<td>2.10 Factor definitions for hierarchically summarizing redundancy removal</td>
<td>32</td>
</tr>
<tr>
<td>2.11 Hierarchically summarizing large documents and redundancy removal</td>
<td>32</td>
</tr>
<tr>
<td>2.12 Compatibility definitions</td>
<td>33</td>
</tr>
<tr>
<td>2.13 Comparing key capabilities and approaches in survey</td>
<td>33</td>
</tr>
<tr>
<td>2.14 Maturity of methodologies</td>
<td>35</td>
</tr>
<tr>
<td>2.15 Comparison of maturity indicators</td>
<td>35</td>
</tr>
<tr>
<td>2.16 User and developer estimated weights for each maturity indicator</td>
<td>36</td>
</tr>
<tr>
<td>2.17 Comparison of maturity indicators (user and developer weighted)</td>
<td>36</td>
</tr>
<tr>
<td>3.1 Component Functions</td>
<td>69</td>
</tr>
</tbody>
</table>
# LIST OF TABLES (CONTINUED)

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2</td>
<td>73</td>
</tr>
<tr>
<td>3.3</td>
<td>74</td>
</tr>
<tr>
<td>3.4</td>
<td>74</td>
</tr>
<tr>
<td>3.5</td>
<td>75</td>
</tr>
<tr>
<td>3.6</td>
<td>76</td>
</tr>
<tr>
<td>4.1</td>
<td>89</td>
</tr>
<tr>
<td>7.1</td>
<td>131</td>
</tr>
</tbody>
</table>

## Table 3.2
Maturity Definitions

## Table 3.3
Accuracy, Coverage, Performance, Scale

## Table 3.4
Developer and User Weights

## Table 3.5
Maturity of Methods (Estimated, Un-weighted) plus their Accuracy, Coverage, Performance, and Capacity

## Table 3.6
Maturity Scores of Methods 2 (weight) x (estimated maturity)

## Table 4.1
Rules for combining duplicate vertices within a group when creating chains of events

## Table 7.1
Summary of Research Contributions
In today’s digital communication, information is represented in various forms or modalities [1-90]. The processing and understanding of various forms of information requires the efficient representation and association of multi-modal information. Thus, the association and understanding can be effectively done by converting multiple modalities into one model of representation. In this study the main research modalities involved are NL processing and understanding, image processing and understanding, document processing and computing models. Thus, below we offer a brief description of these research fields before we make their association.

Natural Language (NL) processing and understanding is an old research field with many research achievements and problems. It has been defined as the translation of machine-readable text into an internal representation. This definition provides three ways of viewing NL processing techniques: by the type of text read (e.g. dialogue, story, or scientific articles), by the type of internal representation used to store the transformation (e.g. semantic nets and graphs), or by the type of computational mechanisms (morphological lexicon, syntax, semantic and pragmatic) used to perform the transformation. There are several tools and systems developed for NL processing, such as spelling checkers (WP, MSW, WS, etc), grammar checkers (OW, WSG, etc), multi-lingual dictionaries, machine translation systems (limited use, EUROTRA, TOSHIBA, TAUM-M, etc.), NL search engines for databases [90].

The document processing is a newest field, which lately has being getting extra attention due to synergistic integration of image and NL understanding [90]. Methods deal with document processing and understanding are related with techniques such as segmentation of a page(s), separation of text from images, image analysis and understanding, words recognition, text understanding, association of text and images for knowledge discovery and representation, etc.. Some of these methods, such as document segmentation is presented, which is a "top-down" approach and produces good results under the condition that the examined page can be separated into blocks. Another algorithmic approach is presented, which is a "bottom up" process with good performance in several categories of pages with good spacing features, and "non overlapping" blocks. In addition, a method presented that separates images from text (typed or
handwritten) by maintaining their relationships. An extension of the document processing with a variety of applications such as human computer interaction (HCI), knowledge discovery, image/document databases accurate retrieval, etc is the document understanding that combines image understanding-interpretation and natural language processing -understanding.

The SPN model is very successful with many applications, especially in the simulation area and recently in knowledge engineering [90]. One very interesting and successful application of the SPN model is its use as a structural and functional representation scheme. The SPN model has the ability to represent structural knowledge extracted from images, like objects and their attributes, or described by NL language expressions. It also has the ability to efficiently represent the functional behavior of systems. The combination of this formalism with Augmented Semantic Grammars (ASGs) is a new approach to representing knowledge with a synergistic structural and functional model, which provides an efficient testbed for multimodal sources.

The main objective of this dissertation is to develop a new novel methodology to convert NL text sentences into Graphs and then into SPN graphs in an effort to offer an association of event described in NL text or images associated with text. To accomplish such a goal we offer we firstly present two surveys, then present the implementations of our model-methodology in two steps: (1.) An NL text to graph conversion program and (2.) an extension of our program to convert NL text to Stochastic Petri Nets from which we provide the greatest contribution of our research. Results from our research effort are also provided.

The organization of this dissertation is as follows:

Chapter II provides a comparative survey of natural language understanding (NLU) methodologies used for processing multiple documents. The scope of these methodologies is to generate a text output with reduced information redundancy and increased information coverage. The purpose of this chapter is to establish what methodologies exist and present their features, capabilities and maturities based on evaluation criteria selected by users and developers. Tables of comparison, at the end of the survey,
provide a quick glance of these technical attributes and maturity indicators abstracted from available information in the publications.

Chapter III provides a survey for transforming natural language sentences to graph representations. Its purpose is to provide a background of recently developed graph methodologies obtained from the literature. It describes various text to graph methodologies developed in recent years. This chapter also presents an analysis of the methodology’s approach, capabilities, and maturity. NL applications from methods in this survey include event resolution (ER), grammar annotation (GrA), information mining (IM), knowledgebase (K), labeling (Lab), novelty detection (ND), question/answer (QA), redundancy reduction (Red), semantic relatedness (SR), similarity measure (SM), summarization (Sum), textual entailment (TE), word sense disambiguation (WSD), and word sense induction (WSI).

Chapter IV describes our Java implementation of a program that transforms NL text into a graph representation. The program captures events described in a text by representing each phrase and sentence in the form of agents that perform actions on patients. Each agent, action, and patient of each phrase and sentence is represented as a node in a graph with their relationships represented as directional edges. The program also combines graph representations of phrases and sentences into more global representations of the text to capture the effects of actions throughout the text. Our program uses the Stanford parser [80, 81, 82] to obtain a parsed input stream of words with parts of speech tags, transforms the stream into data structures containing nouns and verbs representing subjects as agents, verbs as actions, and objects as patients of each phrase and sentence. The implementation generates data structures to support multiple phrases within sentences, multiple agents, actions, and patients per phrase, active and passive sentences, and complex sentences. It accounts for multiple patients impacted by one or more actions and multiple actions caused by one or more agents. It uses the Java Universal Network/Graph framework (JUNG2) [83, 84, 85, 86] containing Java libraries to generate graph visualizations of the agent, action, and patient nodes found within each phrase and sentence of a stream of text. The resulting graphs contain chains of agents, actions, and patients of the entire input text. These chains are formed by compressing multiple agent and patient nodes of the same label into single nodes. Compression of actions are avoided to keep from adding ambiguity such as erroneously adding more influence from agents than what is represented in the text.
Chapter V describes our graph to Stochastic Petri Net conversion and representation. It shows results of the same test cases we developed and used for our graph implementation. To provide visualization of each SPN component converted from graph nodes, we (1.) changed the shape of each agent and patient node from rectangles to ellipses, to represent SPN places, and (2.) changed the shape of action nodes from long rectangles to thin (small width) rectangles to represent transitions. We included some sample code we developed in order to use a transformer method from the JUNG2 library. To provide SPN functionality, we added inhibiter places to control SPN transitions, which reduce sentence interpretations based on an SPN synthesis guided by event association throughout the SPN model of the entire text being analyzed. We demonstrate sentence ambiguity reduction by event association using SPN modeling of NL text and present results in the form of SPN graphs.

Conclusions from our research are listed after chapter VI.
II. GENERAL SURVEY
A COMPARATIVE SURVEY ON NLP/U METHODOLOGIES
FOR PROCESSING MULTI-DOCUMENTS

Abstract: This paper provides a comparative survey of natural language understanding (NLU) methodologies used for processing multiple documents. The scope of these methodologies is to generate a text output with reduced information redundancy and increased information coverage. The purpose of this paper is to inform the reader what methodologies exist and their features, capabilities and maturities based on evaluation criteria selected by users and developers. Tables of comparison, at the end of this survey, provide a quick glance of these technical attributes and maturity indicators abstracted from available information in the publications.

2.1 Introduction

Over the last two decades, information, especially from the internet, has become so vast that professionals, from a number of disciplines, have difficulty keeping up to date within their respective fields. For example, medical doctors devote tremendous amounts of time to capture the latest developments from research areas within their field of specialty. A large amount of this time is wasted reading redundant information from various documents. Needed information may also be lost in the process of summarization. Advanced methods of search, database technologies, data mining, and other areas have helped, but not enough to meet the growing need from these professionals.

For the past 40 years, researchers have tried to address this problem by automatically or semi-automatically capturing information from single and multiple documents into less redundant text, typically in the form of summaries. If a sufficient solution will be found, the resulting increased capability would become a significant breakthrough and help researchers and professionals capture more information to advance their areas of specialty and collectively advance a multitude of technologies. In addition, several methodologies have been developed to advance the area of natural language processing in order to find...
solutions to this problem. However, no known methodology appears to capture the needed information and generate text with enough quality and speed to satisfy this need. Thus, this survey summarizes and compares current methodologies, which deal with the removal of redundancy for documents retrieved from different resources.

This paper surveys research methodologies related to the area of natural language processing (NLP) and natural language understanding (NLU). The purpose here is to document the progress in natural language understanding research and how it can be applied to capturing concepts from multi-documents and producing non-redundant text while attempting to maximize coverage of the significant information needed by the user. The required information could span from a single or multiple domain coverage. Thus, this paper explores the current state of NLU technology and seeks its robustness, time and space complexity, scalability for handling large numbers (n > 100) of technical documents, and technology gaps needing further research. The information in this survey summarizes papers from various technical journals and conference proceedings.

The methodologies under evaluation in this paper cover the following areas: (1) detection of important sentences, (2) concept extraction from text, (3) building concept graphs, (4) attribute and relation structures leading toward knowledge discovery from text, (5) increasing efficiency in the processes leading to concept representations, (6) generation of non-redundant text summaries, and (7) maximizing the readability (or coherence) of automatically generated or extracted text. Technologies used in these methodologies include machine learning, statistical and discrete approaches, cluster indexing, lexical chaining, concept lattices, models that follow cognitive psychology concepts, stochastic Petri-nets (SPNs) and fractal theory. Technologies taken from these areas of research provide a broad spectrum of methodologies or approaches that can provide a baseline for further NLU research. Moreover, methodologies that create summaries of single and multiple documents can be grouped into two major approaches: One approach includes capturing important concepts from text, using a collection of sample texts (called corpus) to train machine learning algorithms, minimizing the number of concepts without losing too much information, generating summarized text, and making the resulting text easily readable. Another approach is extracting the more
important sentences (or phrases) that can be directly converted to non-redundant summaries and bypassing the concept capture and generation steps.

2.2 Methodologies and Their Features

In this section we present a variety of methodologies classified according to their features. In particular this section covers the various groups: text relationship map with latent semantic analysis, extraction methods for text summarization, cluster summarization, formulated semantic relations, SPN representation for document understanding, concepts representation for text, learning ontology from text, synthesis of documents, generation of semantically meaningful text using logic order, text generation methods, document structural understanding, and other relevant methods. The methods presented here will be compared and evaluated based on their maturity. The overall results are presented in section 3.

2.2.1 Text Relationship Map with Latent Semantic Analysis (LSA)

Yeh et al (2008) present the two methodologies, text relationship map and latent semantic analysis, that they use together for text summarization. In particular, the first methodology (2008a) uses feature weights to create similarity links between sentences forming a text relationship map. Sentence position (within a paragraph or document), keywords (that can add or negate), centrality, and resemblance to the title, together determine feature weights that contribute to sentence importance within the document. The authors also use Latent semantic analysis (LSA) to extract and infer relations of words to their expected context. A sentence vs. word matrix analyzes use of words within context. Corpus-based information and scoring functions, using feature weights to trigger the creation of similarity links between sentences, are represented in a text relationship map (TRM), or graph. [41] Summary: This methodology captures various features that help in calculating the similarity of sentences throughout one or more documents. Their paper offers significant details about the methodology. This methodology, however, is based on the word level only.

The second methodology (2008b, LSA-based text relationship map (T.R.M.) approach by Yeh et al (2008)), derives semantically salient structures from a document. Latent semantic analysis (LSA) is used for extracting and inferring relations of words with their expected context. The authors use it to derive latent structures from a document. They elaborate an LSA method that derives semantic representation and
propose a method for generating a summary from a semantic representation. Four phases include:

1) "Preprocesses" partitions sentences using given punctuation and segments sentences into keywords using a toolkit called AutoTag (Academia Sinica, 1999). 2) Semantic model analysis uses a word-by-sentence matrix and produces a semantic matrix using singular value decomposition (SVD) and dimension reduction. 3) Text relationship map is produced by the semantic matrix. 4) Sentence selection uses the global bushy path from the text relationship map to select the important sentences that provide the summary"[41].

**Summary:** Their paper provides detail about the methodology used. Also, several features are used in the similarity calculation. This methodology is also based on the word level. The LSA approach uses a Word-Sentence matrix that can get very large due to the number of words in a document or in multi-documents.

### 2.2.2 Extraction methods for text summarization

Ko and Seo (2008) present a hybrid sentence extraction methodology that uses some context information augmented with mainline statistical approaches to find important sentences in documents. Their model combines two consecutive sentences into a bi-gram pseudo sentence representation to overcome feature sparseness. By using traditional statistical methods, they calculate a score based on sentence similarity to a query, location within a paragraph (first or last sentence, etc.), aggregation, and frequency of the same pseudo sentence. Each of these factors adds to the importance of the corresponding sentence by summing products of weights. A sliding window combines adjacent sentences to form a bi-gram. Once enough bi-gram representations are selected for a summary, each bi-gram is converted back to two sentences which are used in the resulting summary [19].

**Summary:** Test results of the hybrid sentence extraction approach showed that it outperformed other approaches listed by a small percentage. What the authors (of the hybrid approach) call context information is limited to two consecutive (i.e., adjacent) sentences with no global context capability implied. Normally, context would imply more extensive surrounding information than groups of two adjacent sentences.

### 2.2.3 Cluster based summarization

In this subsection we present two methodologies. The first methodology, Moens et al. (2005), extracts important sentences and detects redundant content across sentences. They use generic linguistic
resources and statistical techniques to detect important content from topics and patterns of themes throughout text. From this, they build hierarchical topic trees from text. Then, they segment topics and summarize at each level of topic detail. Their parser detects main grammatical constructs and finds semantic relations between content items. They use statistical techniques to cluster lexical and syntactic features of sentences, and then detect redundant content to generate summaries of multiple documents [25].

**Summary:** Moens et al. methodology provides a significant capability in automatically finding content from text and representing it by hierarchical topics and subtopics. This provides flexibility in selecting how much detail goes into the summary. From competitive testing at DUC 2002 and 2003, the performance of the methodology provided good results, even when compared with trained methodologies. Topic trees and themes are the main information sources to be captured using this methodology. Although these contribute to forming a summary, more queues could be added to enhance the accuracy of this approach. The authors discuss several improvements that could be made. This system incorporates several technologies to provide flexibility. It appears that system integration could be improved to make this a better product.

The second methodology by Radev et al. (2004) uses a Cluster Centroid-Based summarization technique called MEAD which detects topics and tracks to evaluate the results. This methodology measures how many times a word appears in a document, and what percentage of all documents in a collection contains a given word. A cluster is a set of words that are statistically important to a cluster of documents and are used to identify important (or salient) sentences in a cluster [30]. **Summary:** The authors state that the MEAD algorithms produced summaries similar in quality to summaries produced by humans for the same documents. Additional factors could be addressed to help provide higher quality output. Scores determined, using this methodology, are limited to word frequency, position, and sentence overlap. More factors could be added to improve redundancy removal of the resulting summary output.

### 2.2.4 Chaining lexically to formulate semantic relations

This category includes three methodologies. The first methodology by Silber and McCoy (2002) propose an algorithm that improves the execution time and space complexity of creating lexical chains from exponential to linear in order to make their computation feasible for large documents. Lexical chains are created as an intermediate representation to extract the most important concepts from text to be used for
generating a summary. They also evaluate their implementation of lexical chains as an efficient intermediate representative format. Silber and McCoy implicitly store every interpretation of source documents without creating each interpretation as a lexical chain, thus reducing the vast number of lexical chains from multiple word senses per noun instance. [35]

**Summary:** Silber and McCoy’s algorithm provide linear time for calculating lexical chains which is a big step from former exponential time complexity implementations they reference from 1997 implementations and earlier. Their focus is on efficiency of one part of the entire process. They leave some issues left for future work.

The second method by Manabu and Hajime (2000) provides a lexical chaining based on a topic submitted by a user. Lexical chains are sequences of words related to each other that form a semantic unit. This procedure increases coherency and readability of resulting summaries which yields improved accuracy or relevance to the user. (This has an objective increasing coherency and readability of a generated text summary similar to Barzilay and Lapata [2] but applies the lexical chaining methodology.) The methodology constructs lexical chains, calculates scores of the chains based on high connectivity with other sentences, and constructs clusters of words using the similarity score [21]. **Summary:** This methodology provides a higher-level calculation of semantic similarity and offers potential increase in accuracy. Results showed improved accuracy but left possibilities of ignoring other useful information. More improvements need to be made.

The third methodology by Reeve et al. (2006) proposes the implementation of a methodology by using lexical chaining for concept chaining (distinguished from term chaining) to identify candidate sentences for extraction for use in generating biomedical summaries. Their concept chaining process consists of text to concept mapping, concept chaining, identifying strong chains, identifying frequent concepts and summarizing. They use the resulting sentences to generate the summary [32]. **Summary:** Test results (90 % precision and 92 % recall) are high compared to results of other lexical chaining methodologies in this survey. Concept disambiguation is not implemented but planned for future work. Complexity appears not to be addressed. Internal evaluation was specifically toward quality of generated summary.
2.2.5 Stochastic Petri-Net (SPN) representations for NLU

The methodology by Bourbakis-Manaris, based on SPNs Modeling of the NL text sentences for Document Understanding, (1998). They describe four levels of processing: lexical to enforce case (subject-verb) agreement, syntactic to combine words into sentences, semantic to assign meaning to words and sentences, and pragmatic to form context from relations to previous sentences, paragraphs, topics, and information from related data. This paper focuses on the more difficult syntactic and pragmatic process. Multiple modalities or external forms of information such as speech, images, text, video, gestures, facial expressions, hand signs, and handwriting are proposed to add to the context formed by the pragmatic process [4]. Summary: The combination of ASGs and SPNs in this methodology provides significant capability in not only capturing semantic meaning from text but extracting contextual and other available information to resolve ambiguities. The methodology suggested in this paper shows how SPNs, used with ASGs, can model a tremendous amount of interrelationships that exist in both text and imagery. It provides significant potential for extended areas such as knowledge abstraction and representation and adding to their capabilities. The methodology presented in this paper also illustrates the potential for SPNs to model technologies in ways that significantly enhance their modeling capabilities compared to conventional (main line) approaches in using SPNs. The computational complexity, however, is high.

2.2.6 Building concept representations from text

This subsection includes three methods. The first method by Ye et al. (2007) proposes a concept lattice to represent text understanding and to extract text from multiple documents and generate an optimized summary. The concept lattice provides indexing of local topics within a hierarchy of topics. The topics, represented by nodes in the lattice, correspond to concepts that appear frequently throughout the original text or document(s) and sentences that contain these topics. The resulting summary contains an optimized set of local topics and a maximized coverage of concepts for a desired size of summary (number of sentences) [40]. Summary: The document concept lattice approach provides an efficient way to account for all possible word senses without calculating them all on line. This provides significant improvement in accuracy without the computational complexity. The approach reduces complexity from $O(n^2)$ to $O(1)$, or more likely $O(n)$ which is still linear. WordNet is required for this approach. This may limit some of the capabilities of this approach to WordNet implementation constraints.
The second method by Guo and Stylios (2005) investigates event indexing by applying cognitive psychology to create clusters for building concept representations from text. Their methodology extracts the most prominent (or important) content by lexical analysis at phrase and clause levels in multiple documents. This approach forms clusters of sentences with the same actor, sorts each actor cluster by time and space displacements, creates causal chains by linking across, creates intention chains by linking sentences (across clusters) with the same goal. The result is a two dimensional indexed cluster representation of groups of sentences. The EIS model parses sentences, analyzes semantic trees, clusters sentences with indices, performs cluster filtering and reduction, controls size, and outputs the resulting text summary. [15]. **Summary:** Working at the phrase or clause level is an advantage over word level. This reduces the number of possible combinations of pairs (phrase, sentence) instead of (word, sentence) for example. Multi-document capability is another plus for the user. Features such as actors, time/space displacements, causal chains, and intention chains add a significantly more capability to detecting sentence similarities. Reducing all this potentially multi-dimensional vector data to two dimensional index clustering is a significant savings in complexity, especially storage complexity. Testing and evaluation of causality and intention indexes were delayed until a future date. (The authors plan to include results in future publications.)

The third method by Cimiano et al. (2005) forms concept hierarchies using formal concept analysis (FCA) through learning. Their methodology automatically acquires (through learning) concept hierarchies from collections of text (corpus). A collection of text is tagged with parts of speech, parsed, converted to normalized form, smoothed, weighted, pruned using a distance threshold, transformed to formal context with implicit relationships between objects, converted to a lattice of concepts (through FCA), and compacted to a partial order representation of concepts. They use distance measure between vectors to determine concept similarity [7]. **Summary:** Automatic (unsupervised) leaning approach is a big plus, reducing the traditional manual work to near zero. The concept similarity calculation uses more characteristics that can result in greater accuracy of output text. The authors state “this is a first time
approach.” Similarity calculations are made at the concept and semantic level, using LSA. The approach appears to be integrated with the LoPar parser implementation, but benefits in capability are significant.

2.2.7 Learning ontology from text

This category includes ten methods. The first method by Bendaould et al. uses relational concept analysis (RSA) to formulate concepts through text-based ontology. The authors present a semi-automatic methodology that builds ontology from a set of terms extracted from resources consisting of text corpora, a thesaurus for a particular domain, and syntactic patterns representing a set of objects. A Description Logic (DL) language in the form of expressions is used to find the exhaustive definition for each concept and each relation of the ontology. Their methodology builds a glossary of terms, concept taxonomies using the Formal Concept Analysis (FCA), and transversal binary relation diagrams using Relational Concept Analysis (RCA) to extract traversal relations. The result is a concept relations and instances to support reasoning, including classification, instantiation, and consistency checking [3]. Summary: This is a very methodological treatment at the higher-level concept representation. This methodology is more for building ontology and less on capturing the information from text, but has significant capability. The computation could have high complexity, just based on the description.

The second method by Valakos et al. (2006) uses machine learning to build and maintain concept representations called allergens ontology. Building an ontology includes: selecting concepts, specifying their attributes and relations (between concepts), and filling (populating) their properties with instances. Their methodologies include: specification, conceptualization, formalization, , and maintenance. Author’s approach automatically creates training examples using the domain ontology with limited size and domain-independent rules. They enrich ontology with a lexico-semantic relation that is unsupervised (fully automatic) [37]. Summary: Authors machine learning approach provides a way to capture new knowledge in the form of concepts, their attributes and properties, and relations between them and maintain (or update) the knowledge with what has been established. The approach includes lexical to semantic relations to transform lexical to semantic information, which is a contribution toward proving concepts. Specific details on how to extract information to form concepts is not presented. The approach is specific to
maintaining ontology within a medical (allergen) domain but its general principles could be applied to other applications.

The third method by Zhou and Su (2005) uses machine learning to integrate evidence from internal (within the word) and external (context) to formulate named entity recognition, a method which extracts and classifies text elements into predefined categories of information. Their system called PowerNE is implemented using a Hidden Markov Model (HMM) to determine hidden states representing tags corresponding to portions of text or individual words from the text and a constraint recognizer algorithm to deal with data sparseness. The system seeks answers “who, what, and how much” and after further processing determines “what and how.” Evidences from the entity names include (a) internal deterministic evidences such as capitalization and digits, (b) semantic triggers, (c) appearance of a word in a list, and (d) context of a word such as other words in the text that mean the same thing, i.e. aliases of the word [42]. Summary: This named entity recognition approach provides significant and useful detail that could be applied to information extraction from text. Machine learning is applied to recognizing named entities and is used along with constraint recognition, Hidden Markov Models to determine tags, and mutual information to increase coverage of non-redundant information. This concept provides significant capabilities on the theoretical level but appears to need further development before product information with metrics is available.

The fourth method by Shunsfard and Barforoush (2004) proposes an automatic ontology building approach, starting with a small ontology kernel and implements text understanding to construct the ontology. Their model can handle multiple viewpoints, flexible to domain changes, and can build ontology from scratch without a large knowledgebase. Their hypothesis reduction and selection process consists of assigning a probability to each node of hypothesis space and choosing merge sets, priority ordering of child nodes, and depth first traversing without backtracking. The ontology consists of concepts, relations, and axioms which add information to the concept-relation structures. Their natural language processing portion uses some linguistic and world knowledge to extract conceptual knowledge from Persian text. Knowledge extracting consists of logical, template driven, and semantic analysis [34]. Summary: This system can create ontology from scratch by learning from text, thus significantly reducing manual interaction to
creating and building ontology. This methodology is based on an integration of learning, clustering and splitting of concepts, similarity measures, and several other techniques that, together, form a unique capability showing promise. The current implementation and testing has been limited to Persian text, but the authors plan to expand the system to other languages.

The fifth method by Hahn and Marko (2002) forms concepts from text through machine learning of both grammars and ontology and use evidence, or background knowledge, to steer refinement of generated text. This methodology is an integrated approach for learning lexical (syntactic) and conceptual knowledge as it is applied to natural text understanding. The lexical portion consists of grammatical knowledge containing a hierarchy of lexical classes which provide constraints on unknown words. Conceptual knowledge (or domain knowledge) provides a classification (or taxonomy) of concepts used to evaluate newly derived concept descriptions based on prior knowledge. The proposed model uses these different types of knowledge simultaneously in order to increase knowledge accuracy while constraining the growth of hypothesis space to keep the knowledge acquisition process going [16]. Summary: Evidence within both lexical and conceptual hypotheses is used together to bound the resulting number of hypothesis search space to a manageable quantity while refining the lexical and conceptual quality and, therefore, increasing the accuracy of text understanding. Complexity of the approach can be extensive but tractable.

The sixth method by Loh et al. (2003) provides a text mining approach to form concepts from phrases and analyzes their distributions throughout a document. The approach combines categorization to identify concepts within text and mining to discover patterns by analyzing and relating concept distributions in a collection. Categorization compares each phrase against rules that define a concept. Rules combine positive and negative words. The frequency of the same concept in a document plus how much it is referenced adds to its importance. The mining uses a software tool that counts the number of times a concept is present, generates a vector (or centroid) of concepts with their frequencies in a collection to find document themes, compares centroids to find common themes or differences, and discovers associations between concepts. The methodology combines collections and sub-collections to discover patterns, mines to discover class attributes or descriptors, and uses supervised learning to discover
reasoning patterns systems [20]. **Summary:** This approach captures concepts from phrases, finds patterns from concept distributions, and discovers themes within a document by collecting concepts and generating centroids to represent the collections. Together, these features contribute to a knowledge discovery technique, which contributes significantly toward mining concepts and relations from text. This approach was developed for decision support systems, thus limiting extensions into other domains. It also uses supervised learning to find reasoning patterns, thus requiring interaction from the user and possibly slowing the response of an implementation of this approach.

The seventh method by Rajaraman and Tan (2002) constructs a conceptual knowledge base, called a concept frame graph, for mining concepts from text. A learning algorithm constructs the concept map which is guided by the user, i.e. supervised learning. The authors’ approach consists of (1) constructing a conceptual knowledge base (CKB) consisting of concepts and relationships that are extracted from text and (2) CBK mining of the knowledge base for new information. The authors pre-process each document to a body of text, extracting all entities using a co-reference resolution algorithm, extracting 3-tuples consisting of noun clauses, verb clauses, and extended forms of noun phrases and verb phrases which are used to generate the synonym set (Synset) and relations (Rels) parameters of the concept frames. Their parts-of-speech (POS) tagger tags the extracted text and their rule-base extracts noun-verb-noun (NVN) 3-tuples. Their sense disambiguation algorithm uses WordNet and provides separate Synset parameters for each word sense of every word by computing a distance measurement to pick the correct word sense [31].

**Summary:** The approach captures conceptual knowledge from text by constructing a concept map to produce a knowledge base. This provides a high level representation including concepts, relations to other concepts, and relations to synonyms, thus providing representations that can be used to reduce redundancy at the high (concept) level. A clustering algorithm discovers word sense to reduce ambiguous words. The learning algorithm requires guidance from the user. Depending on the amount of manual guidance required, the learning time can be extensive approaching that of supervised learning. Current research practices have been getting away from supervised learning due to the amount of manual intervention.

The eighth method by Pado and Lapata (2007) proposes a general framework for semantic models that determines context in terms of semantic relations. Their algorithm constructs semantic space models
from text annotated with syntactic dependency relations to provide a representation that contains significant linguistic information. Three parameters guide their model construction by finding the most lexically meaningful syntactic structures, weighing importance of different syntactic relations, and finding how to represent the semantic space (e.g., word co-occurrence with other words, parts of speech, or subject-object relations.) Their algorithm builds the semantic context for words of interest from dependency paths defined on a dependency graph, specifies dimensions and provides inference over classes of basis elements, and specifies relative importance of different paths. The model operates on dependency relations and uses linguistic knowledge to form semantic space [29]. **Summary:** This methodology operates at the semantic level and finds context in terms of semantic relations and contains significant linguistic information. The authors state that their model provides a linear runtime performance. A GNU website is provided for a Java implementation of the general framework for semantic models. According to the authors, until further experimentation, the range of data (for the models used in this methodology, compared to those of traditional and LSA-based models) was not fully known but planned for future work. Specific capabilities of interest include word sense disambiguation, automatic clustering, lexicon acquisition, and similarity-based approaches.

The ninth method by Maedche and Staab (2004) presents a generic architecture for ontology learning which consists of components: ontology management (brows, validate, modify, version, evolve), resource processing (discover, import, analyze, transform input data - includes NLP system), algorithm library, and coordination (interact with ontology learning components for resource sharing and for algorithm library). Language dependent NLPs include a German shallow text processor called SMES (Saarbrucken Message Extension System) and an English GATE system. The Porter stemming algorithm is used to normalize natural language terms. Ontology learning algorithms include finding weights to terms in a set of documents assuming a corresponding occurrence of domain specific text indicates a relevant concept. Extracting relations within the taxonomy include using clustering, classifying, or detecting lexical and semantic patterns. Extracting binary relations using association rules (from data mining) can be used to obtain semantic structures by finding statistical co-occurrences in text. A taxonomy and an algorithm for learning generalized association rules result in concept pairs [23]. **Summary:** The methodology finds
semantic patterns and structures and concept pairs. This process of constructing ontology is semi-automatic requiring human intervention.

The tenth method by Dahab et al. (2008) discusses a way for constructing ontology from natural domain text using a semantic pattern-based approach. Their “TextOntoEx” tool extracts candidate relations from text and maps them to meaning representations to help construct an ontology representation. Their approach consists of (1) (manually) using a pattern editor to construct a library of semantic patterns that describe relations between two or more concepts, (2) selecting domain natural text from sources on the internet and convert it to structured format, and (3) extracting and constructing domain ontology from natural text. The domain and topic represent basic classes to enhance with extracted ontology. In other words, they are expanding the existing ontology (or knowledge base) by adding increments of knowledge from the extracted text that conform to the same patterns characteristic of the domain of the existing ontology. They match the converted paragraph with the pattern library and convert paragraphs into semantic pattern formats that serve as an intermediate format for the domain ontology [9]. **Summary:** This work provides semantic pattern formats for converted paragraphs. Manual editing, however, is required for library of semantic patterns.

### 2.2.8 Redundancy removal for document synthesis

This subsection includes three methods. The method by Bourbakis et al. (1999) presents a way for retrieving multimedia web documents and removal of redundant information from text and images. The authors describe (1) a search engine for multimedia web documents that more precisely describe user searches resulting in more relevant returns, (2) a structured query language called WebSSQL that supports queries based on similarity information to retrieve documents significantly closer to the intentions of the user, (3) a methodology for detecting and reducing redundancy of both paragraphs of text and imagery from multiple documents, and (4) combines paragraphs of text and associated images into a single integrated document containing no redundant paragraphs or images. The search engine enables the user to specify more precise conditions for more efficient matching of user queries by using real tables populated by an indexer instead of virtual tables [5]. **Summary:** Out of the papers surveyed, this is the only methodology that provides an integrated similarity detection and redundancy removal of both paragraphs of
text and corresponding images. This approach is also integrated with the authors’ developed query language that includes Webpage (text) and image similarity criteria to yield more definitive returns closer to the user’s intended query. Since the time of the article, other authors have created new features for similarity detection. More text reduction opportunities should be possible with some of the newer features various authors have created. Counts and histograms of text components can detect paragraph similarities up to a certain point. Future developments in capturing the meaning from multiple documents, using approaches such as this as a baseline, should increase similarity detection further, resulting in less text redundancy in the synthesized document.

The second method by Yang and Wang (2008), applies the hierarchical and redundancy sharing characteristics of fractal theory to increase the performance of text summarization when compared to non-hierarchical approaches. Salient features consisting of thematic, location, heading, and cue (pragmatic words) determine a sentence significance score which is used with a fractal summarization algorithm. The algorithm, in conjunction with a tree representation of the document, recursively calculates the fractal value of each child node, using a sentence quota and a range block significance score (from the salient features) [39]. Summary: This hierarchical approach to summarization provides multiple levels of abstraction, taking advantage of the multiple levels representation of fractal theory. More salient features might be added to make this approach more accurate.

The third method by Hilberg (1997) proposes an approach to produce and store higher levels of abstraction that represent sequences of words, sentences, (and might be extended to paragraphs and documents) in the higher (hidden) levels of a neural net. A key benefit of this approach is its near elimination of redundancy of its abstraction or representation and resulting compression of the represented form. Connections (which can be controlled by adding grammar constraints) are represented in higher levels of the neural net. Control of the connections is provided by control codes stored in the neural net nodes. The lowest level words of the natural language are stored in word nodes. Relations between words are captured in codes sent to neighboring nodes of the hierarchical network. This meta-level representation yields compression of the resulting representation. Concepts are formed at the highest meta-level represented by a single meta-word. As a result, a large text can be processed by a small number of
hierarchical levels within the neural net [18]. **Summary:** This proposal has some unique possibilities for representing abstraction. Getting this to work at a large enough scale (large or multi document) may be challenging. The learning of representative corpus of text may be computationally hard to make it work on a large enough scale to get beyond the prototype stage.

2.2.9 *Generated text semantically meaningful through coherence and logical order*

This group includes four methods. The first method by Barzilay and Lapata (2008, representing and measuring local coherence) provides a framework to increase readability and semantic meaning to automatically generated sentences such as a summary of multiple documents. The goal is to order sentences in a way that maximizes local coherence. An entity-grid, representing speech discourse, captures entities distributed in text. This 2-D grid (or matrix) shows presence or absence of discourse entities for each sentence and whether the entity is a subject (s), object (o), or neither (x). An algorithm produces a set of entity transition sequences that provide frequency of occurrence of duplicate transitions. These transition sequences represent distributed, syntactic and reference information characteristic of speech discourse. Barzilay and Lapata capture Grammatical role information from natural language parsers providing broad coverage dependency tree representations and statistical representations [2]. **Summary:** This methodology provides a needed capability to make generated text more coherent and readable. This entity distribution approach provides significant improvement in sentence meaning representation which can result in better automatically generated text. Results of testing showed increased accuracy. New approaches like this will need time to mature, but the benefits should be significant.

The second method by Stein et al. (2000) provides a technique that clusters documents, uses extraction to find main topics and organizes the resulting information for a logical presentation of a summary of multiple documents. This is an interactive approach that focuses on summarizing news line documents. It segments text into passages (paragraphs when feasible) and connects adjacent paragraphs when one or more contain background information for the other paragraph. Combinations of passages are scored to maximize the amount of overlap with a query topic. Their methodology first finds the topical summary of each single document (reducing text to 15%), selects which of these summaries will be included in the multi-document summary by clustering summaries describing similar topics, and includes
the most representative summaries of each cluster in the multi-document summary. They currently use a simple similarity measure but plan to replace it with a better performing one later. The selected summaries are organized by topic summary to keep the generated text form abruptly changing topics [36]. **Summary:** This methodology summarizes multi-document text and is designed to provide a smooth flow of the summary to the reader. It clusters single document representative summaries with similar topics to reduce redundancy and orders the generated summary for multiple documents based on paragraph similarity to minimize the jerkiness of topic changes from paragraph to paragraph resulting in improved readability. The multi-document summarizer currently uses simple similarity scoring approaches but plans to replace them with better performing ones.

The third method by Nomoto and Matsumoto (2003) provides a way to exploit diversity of concepts in text in order to evaluate information based on how well source documents are represented in automatically generated summaries. Their information centric approach finds diverse topical clusters within a text, finds the most important sentence for each topic cluster and uses it to represent the cluster, and generates a summary by collecting and reordering identified sentences to reduce redundancy. They provide an X-means algorithm to improve the traditional k-means approach in their clustering of sentences. They use a scoring model which minimizes loss of the resulting relevance of a summary to a potential query in order to reduce redundancy [28]. **Summary:** This approach provides an improvement in clustering on the information level. The paper provides detailed analysis of its approach verses other traditional approaches and favorable test results including a favorable comparison with human summarization. Although many of the test results of the diversity-based summarizer were superior to supervised learning systems, other results involving highly regular sentences with high agreement could not be exploited to reduce redundancy.

The fourth method by Marco et al. (2002) improves reading order of automatically generated text. The approach is implemented in a system and is designed to analyze heterogeneous documents. This approach assigns logical labels to various elements of documents and analyzes geometric features, spatial relations, and content. The approach captures the physical layout of a document and a logical representation that includes reading order. Neighborhoods of physical partitions are represented as
relations between nodes in a graphical model. The analysis extracts both document layout and logical. Reading order is modeled in a statistical representation [22]. Summary: This approach is implemented in a system that captures the physical and logical layout of generic documents. Most of the discussion focuses on the physical portions of a document and the reading order considers large chunks of what is on a page of a document. It applies more to the big (mostly physical) view of a document, little toward the actual knowledge or understanding level.

2.2.10 Text generation methodologies

Here in this category Dalianis method (1999) uses aggregation before generating text to eliminate redundant text in documents before they can be paraphrased (generated) into natural language. This methodology provides aggregation at the syntax level. Four types of aggregation include syntactic, elision, lexical (both bounded and unbounded), and referential. Syntactic aggregation leaves at least one item in the text to explicitly carry the meaning with no information loss. Elision (not used here) removes redundant inferred information but does not loose implicit information. Lexical aggregation replaces a set of items with a new item, leaving the meaning intact. Lexical can be bounded (keeping overall meaning intact) or unbounded (not keeping overall meaning intact – aggregated information is non retrievable). Referential aggregation (not used here) keeps a trace (or reference) using a pronoun to explicitly keep the aggregated information. Several aggregation rules can apply at different times to various internal data structures [10]. Summary: This approach provides four types of aggregation with rules, which should provide more information for generating significantly less redundant summaries. An update to this paper could provide a more accurate indicator for the state of this methodology.

2.2.11 Document processing and understanding

Aiello et al. (2002) present a methodology to capture the structural layout and logical order of text blocks within several documents and represents this information in connected graphs. Document understanding (as described in this paper) is representing the logical structure and consists of logical object classification and reading order detection. (1.) Document knowledge uses content mapping rules consisting of basic typographical information such as relative size of titles, font size, style, location of page number, and layout arrangement rules consisting of reading order (column or row), paragraph markings, etc.
natural language processing uses bi-grams and tri-grams to compute transition (such as order) probabilities between text blocks. (2.) Logical object classification uses seven feature vectors consisting of aspect relation, area ratio, font size ratio, font style, content size, number of lines, and neighbor of figure. (3.) Reading order detection is determined from geographical and content information [1]. **Summary:** This document level methodology captures physical layout of partitioned text blocks spanning over multiple documents with a complexity of $O(n^4)$. This provides only top-level information about a set of documents. Without being used in conjunction with other methodologies discussed in this survey, the information provided does not include information from within text blocks. Information within text blocks is a needed feature addressed by other methodologies.

Feldman et al. (2003) describe a natural language processing (NLP) system, called LitMiner, which uses semantic analysis to mine biomedical literature. This paper focuses on text mining of "relationships among genes, proteins, drugs, and diseases to facilitate an understanding and prediction of complex biological processes. The system uses information extraction (I.E.) which finds meaningful entities and relationships to enable text mining for desired information, as opposed to the text categorization approach which uses pre-defined, thematic categories and selects which categories to associate entities from the text. The system uses syntactic and semantic elements of text, looks for sequences and patterns of such elements, generates on-the-fly Boolean checks for specific attributes, and uses information extraction rule bases. A key module of the system is the rule based information extraction module which consists of multiple layers: infrastructure (libraries and NLP morphology tools), metadata management (to identify candidates and update scores), identification (to extract occurrences with synonyms), structure analysis (to identify sections on clues), main layer (to extract evidences using heuristics, local patterns), and lexical resources (for analysis) [12]. **Summary:** Although this paper specifically addresses the biomedical domain, it is generally useful in providing the various steps and different methodologies for text mining, plus describing, in detail, the specific system with good evaluation results for this type of system. Several of the key elements are interlinked with the biomedical domain. However, several of the methodologies presented appear to be applicable to several domains. (Different data bases and tools would be needed.) The system described requires some pre-processing and is a semi-automatic process with a visualization system.
### 2.2.12 Other relevant methodologies

Franceschet and Montanari (2004) method addresses time granularity using time based logic programming and parsing techniques. They provide a logical approach to representing reasoning for time granularity. By combining temporal logics with parsers over layered structures, they seek to represent natural language involving time. They define a new, temporal (or time based) automata and show properties relevant to Boolean operations, decidability, and expressive equivalence with respect to temporal logic. The goal is to benefit automata closure properties [13]. **Summary:** The approach provides some insight on how to represent time in various applications including natural language understanding. It is mainly theoretical, it goes deep into automata theory, and may be too much of a divergence from the purpose of this survey. It is referenced here as a theoretical approach to capture time information in a form of knowledge representation.

Friedman et al. (2001) describe a system for extracting pathways from journal articles. GENIES’s term tagging component identifies gene and protein names in text by using rules and external knowledge. The GENIES system uses a semantic grammar, but also includes syntactic knowledge interleaved with semantic and syntactic constraints and works with original complex sentences. It always attempts to get a complete parse for high precision, but uses alternative strategies such as segmenting and partial parsing to improve recall. It captures relations between interactions, including complex nested chains of interactions. GENIES parses complete journal articles, semantically classifies and captures a complete set of interactions and relationships between biological molecules, and handles normalized and agentive forms of verbs (e.g., inhibition and inhibitor), and assigns semantic features to verbs of interest (e.g., number of arguments expected, argument order). It recognizes 125 different verbs and partitions them into 14 broader semantic classes [14]. **Summary:** This system uses semantic grammar that interacts with syntactic knowledge for high precision attempts with improved recall. It is designed for complex text and assigns semantic features to verbs. This methodology is domain limited to medical (biological molecules, etc.).

Hahn et al. (2002) describe a natural language system for extracting medical information from reports. They describe an information extraction system called MedsynDikate, designed for information rich text understanding and domain knowledge. They also propose two alternate methods to support
knowledge acquisition and deal with system scalability issues: One is to automatically learn concepts from text. The other is to automatically transform existing repositories (i.e., UMLS data structures) into a description logic framework in order to have a large enough knowledge base and in a compatible format for their system. Their SynDikate system incrementally learns new concepts while extracting them from text and adds these new concepts to a given core ontology. When new words, not in knowledge base, are encountered, the parse trees interpret their impact on concepts and add them to the derivation of an annotated concept hypothesis [17]. **Summary:** The system learns new concepts from text and adds to its knowledge base. This methodology is also domain limited to medical documents.

Neustein (2001) uses sequence analysis to improve natural language understanding from conversations. A goal of this analysis of sequence packages (or frames) of speech is to uncover important information that might otherwise get unnoticed. It is proposed as a contributing component of natural language systems that model spontaneous speech. Context dependent features of discourse, ambiguities, speech fragments, and metaphors are among the vague portions of the spoken language are among the problem areas that this analysis attempts to address. Sequence package analysis breaks natural language communication into sequences and turns within sequences in order to determine the organization and verbal interactions of a dialogue. Such analysis provides information, more at a contextual (or global) level, than analyzing isolated portions of speech [27]. **Summary:** The proposed sequence analysis would address context dependency in natural language, especially in speech context. Success in this kind of analysis should provide benefits toward reducing ambiguity in natural language processing and understanding. This discussion is basically a proposed approach to a very difficult problem area and didn’t appear to be implemented at the time the paper was written. Little details of the approach were presented at the time this paper was published.

Wang and Yang (2006) analyze the impact that multiple language documents have on text summarization. Their methodology extracts key sentences based on salient features (features that stand out in importance). As they discussed in their paper on hierarchical summarization, they use fractal theory to explore hierarchical structure and salient features. In this impact analysis paper, the authors stress how text summarization involving multiple languages, English and Chinese in this case, have on their summarization
methodology. A pair of parallel documents in two languages normally have different structure and order but the authors could merge sentences into one, mix (and rephrase) sentences to form other sentences (or content), and align sentences. The different (English and Chinese) languages impacted the thematic weight of key words which caused some differences in the resulting text summary [38]. Summary: The author’s methodology summarized mixed language documents and revealed minor differences when extracting themes from the documents. This analysis of multi-language impact on document summarization is a beginning step to using multi-languages in natural language understanding. Much research in this area is needed before it can result in mature products.

2.3 Comparative Analysis of the Methodologies

The purpose of this comparative analysis in the following paragraphs is to capture the key contributions of each approach, group them in categories of methodologies, and compare them. After each group of approaches is located a table that compares their combined capabilities. At the end of this analysis are two large tables of all the approaches in this survey. The goal of these tables is to capture in a few tables and paragraphs what current capabilities exist in the natural language understanding area of research including its component technologies. A maturity table is also listed, but few of the approaches are mature enough to contribute to the maturity table.

2.3.1 Extraction approaches

In document summarization methodologies, clusters normally represent a collection of entities such as words, sentences, paragraphs, concepts, etc. in graphic form and weighted links between them represent degrees of similarities. In other approaches, links, between concepts, represent higher level relations forming semantic information, often associated with a low level of knowledge. Extraction of sentences, selected by their importance, is often used to generate the resulting summaries.

In Yeh et al (2008) summarization approach using a text relationship map and latent semantic analysis (LSA), the text relationship map consists of similarity links between paragraphs (corresponding to clusters of points on a graph) that are scored from a sum of features: sentence position, positive and negative key word, centrality (similarity with other sentences), and resemblance to title, each with weighting factors to calculate the importance of a sentence. The number of links from a paragraph (or
cluster) correlates with importance of the paragraph. They also calculate mutual information “\( \text{MI}(x,y) = \log \frac{p(x,y)}{p(x)p(y)} \) \[41\] and a genetic algorithm (with genome selection by fitness) in selecting the weights of the above features. Then, latent semantic analysis is added to derive semantic representations used to generate a summary \[41\].

Ko and Seo (2008, para. 4.1) add query and aggregation features in calculating the score for selecting sentences. They also combine two sentence representations into one to decrease the sparseness of matrices representing sentences. They use similarity of terms in a sentence to a user’s query rather than to the title of a document \[19\].

Moens et al (2005, para. 5.1) build hierarchical topic trees from the text, detect content from topics and patterns of themes in text, find main topics of sentences, and compute frequency of terms. They use two clustering approaches: (1) covering or minimizing clusters to keep similarities between objects (inside the clusters) and (2) k-medoid or maximizing the sum of similarities between objects and their medoid (partitioning objects into k-clusters) \[25\].

Radev et al. (2004, para. 5.2) provide an extraction algorithm based on a score calculated from centroid value, position, overlap, and redundancy. Clusters of words contain a level of importance corresponding to a cluster of documents. A centroid of a cluster is used to find sentences that correspond to the topic of the cluster. This methodology produces a summary from multi-documents \[30\].

| F1 | F2 | F3 | F4 | F5 | F6 | F7 | F8 | F9 | F10 | F11 | F12 | F13 | F14 | F15 | F16 | F17 | F18 | F19 | F20 | F21 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| Similarity | Context | Feature | Clusters | Semantic | Links | F6 | Aggregation | Feature | Overlap | Title | F11 | F12 | F13 | F14 | Topic | F16 | F17 | F18 | F19 | F20 | F21 |
| Links | F7 | Pos/Negative | F9 | F10 | F11 | F12 | F13 | F14 | F15 | F16 | F17 | F18 | F19 | F20 | F21 | F22 | F23 | F24 | F25 | F26 |
| F26 | F27 | F28 | F29 | F30 | F31 | F32 | F33 | F34 | F35 | F36 | F37 | F38 | F39 | F40 | F41 | F42 | F43 | F44 | F45 | F46 |

Table 2.1. Feature definitions for extraction approaches

27


2.3.2 **Lexical Chaining Approaches**

Lexical chains are sequences of words that relate to one another and form a semantic unit. The various approaches produce semantic units that range from semantic similarity, meta-chains, concepts, semantically related words, and themes.

Manabu and Hajime (2000, para. 6.2) construct lexical chains from user topics, calculate scores based on connectivity of chains, and construct clusters of words based on the resulting similarity score. They calculate semantic similarity based on cosine distance (or degree of word co-occurrence within the same text, or semantic unit) [21].

Siber and McCoy (2002, para. 6.1) propose an algorithm with linear execution time complexity to make the lexical chaining methodology feasible for large documents. By using meta-chains, their algorithm implicitly stores every interpretation of source documents without creating a lexical chain for each source document [35].

Reeve et al. (2006, para. 6.3) applies lexical chaining to (a higher level) concept chaining (instead of term chaining) to identify candidate sentences for summaries. Their lexical chaining not only links words to identify semantically related terms, but also themes. They use a Unified Medical Language System (UMLS) to create context maps from the source text. Strong chains are detected from concept repetition, close distance, large number of concepts in a chain, and frequent concepts [32].
Table 2.3. Feature definitions for lexical chaining

<table>
<thead>
<tr>
<th>F11</th>
<th>Topics (from user)</th>
<th>F27</th>
<th>Meta-Chains (Intermediate)</th>
<th>F34</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>F22</td>
<td>Connectivity of Chains</td>
<td>F28</td>
<td>Linear Time Complexity</td>
<td>F35</td>
<td>Number of Concepts</td>
</tr>
<tr>
<td>F23</td>
<td>Similarity Score</td>
<td>F29</td>
<td>Large Documents</td>
<td>F36</td>
<td>Frequency of Concepts</td>
</tr>
<tr>
<td>F4</td>
<td>Clusters</td>
<td>F30</td>
<td>Semantically Related Terms</td>
<td>F37</td>
<td>Complexity: Linear Time</td>
</tr>
<tr>
<td>F24</td>
<td>Concept Chaining</td>
<td>F31</td>
<td>Theme Identification</td>
<td>F38</td>
<td>Word Sense</td>
</tr>
<tr>
<td>F25</td>
<td>Semantic Similarity</td>
<td>F32</td>
<td>Context Map</td>
<td>F39</td>
<td>Score: Relation &amp; Distance</td>
</tr>
<tr>
<td>F26</td>
<td>Co-occurrence of word</td>
<td>F33</td>
<td>Concept Repetition</td>
<td>F40</td>
<td>Precision</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F41</td>
<td>Recall</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>F42</td>
<td>F-Measure</td>
</tr>
</tbody>
</table>

Table 2.4. Lexical chaining approaches

<table>
<thead>
<tr>
<th>Authors</th>
<th>F11</th>
<th>F22</th>
<th>F23</th>
<th>F24</th>
<th>F25</th>
<th>F26</th>
<th>F27</th>
<th>F28</th>
<th>F29</th>
<th>F30</th>
<th>F31</th>
<th>F32</th>
<th>F33</th>
<th>F34</th>
<th>F35</th>
<th>F36</th>
<th>F37</th>
<th>F38</th>
<th>F39</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manabu</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Siber</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Reeve</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.5. Performance (Accuracy)

<table>
<thead>
<tr>
<th>Authors</th>
<th>F40</th>
<th>F41</th>
<th>F42</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manabu</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Siber</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Reeve</td>
<td>90%</td>
<td>92%</td>
<td>NA</td>
</tr>
</tbody>
</table>

2.3.3 Building concepts approaches

The following approaches build concepts directly from text using methodologies other than lexical chaining. These methodologies produce higher level concept representations for generating summaries.

Guo and Stylios (2005, para 7.2) follow a cognitive psychology approach using event indexing. They create clusters of events to build concept relations from text. They extract the most important information content by lexical analysis at phrase and clause levels of multiple documents. They form a two-dimensional indexed cluster to represent groups of sentences [15].

Cimiano et al. (2005, para. 7.3) use formal concept analysis (FCA) to form concept hierarchies (a lattice of concepts) from collections of text (corpus) through machine learning. FCA is a conceptual
clustering technique, used to discover relationships between objects based on their attributes. This produces a context representation for the objects and consists of a triple containing a set of objects, their attributes, and binary indicators for their pairs. The result of this methodology transforms concepts into partially ordered formal concepts and compacts them into an ontological or concept hierarchy [7].

Ye et al. (2007, para 7.1) produce a concept lattice while extracting from multiple documents. Their concept lattice indexes local topics within a hierarchy of topics. They produce a summary by optimizing sets of local topics and maximizing concept coverage globally for the desired summary size (or sentence count). A fitness metric selects candidate sentences for the summary by maximizing answers to the questions who, what, where, when, why, how. They represent partial overlapping of subsets of concepts in the concept lattice and select the optimal combination of sentences minimizing overlapping (reducing redundancy). They develop a concept sense using WordNet to capture all the synonyms, sense explanations and “is-a” and “has-a” relations. Using word sense when detecting sentence similarities finds more words with the same meaning, reduces information redundancy, and results in smaller summaries with the same information content. This methodology reduced on-line computational complexity from $O(n^2)$ to $O(1)$, i.e. linear time complexity [40].

Table 2.6. Feature definitions for building concept approaches

<table>
<thead>
<tr>
<th>F43</th>
<th>Concept Lattice</th>
<th>F50</th>
<th>Intention Chains created</th>
<th>F57</th>
<th>Context – from Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>F44</td>
<td>2-D Indexed Cluster</td>
<td>F51</td>
<td>Semantic Trees analyzed</td>
<td>F58</td>
<td>Conceptual Clustering</td>
</tr>
<tr>
<td>F45</td>
<td>Event Indexing</td>
<td>F52</td>
<td>Hierarchies of Concepts</td>
<td>F59</td>
<td>Fitness Metric - Algorithm</td>
</tr>
<tr>
<td>F46</td>
<td>Clusters (of sentences)</td>
<td>F53</td>
<td>Hierarchies of Topics</td>
<td>F60</td>
<td>Answer (evaluation) who, …</td>
</tr>
<tr>
<td>F47</td>
<td>Extracts Phrase or Clause</td>
<td>F54</td>
<td>Automatic Learning</td>
<td>F61</td>
<td>Encode Candidate Summaries</td>
</tr>
<tr>
<td>F48</td>
<td>Sorts by Actors, time, space</td>
<td>F55</td>
<td>Formal Concept Analysis</td>
<td>F62</td>
<td>Globally Optimal Coverage</td>
</tr>
<tr>
<td>F49</td>
<td>Causal Chains created</td>
<td>F56</td>
<td>Concept Similarity</td>
<td>F63</td>
<td>Summary by Extraction</td>
</tr>
</tbody>
</table>

Table 2.7. Building concept approaches
2.3.4 Ontology approaches

Building and using Ontology as a means of acquiring or discovering knowledge from text: In certain domains, such as medical domains (an ontology) consisting of large knowledge bases, are used with text extraction systems to increase the level of knowledge captured by the ontology. The goal is to increase the accuracy in capturing knowledge from millions of documents. One drawback to this approach is its narrow domain dependency and the static databases that are tied to it. Also, knowledge resources are normally constructed manually by knowledge experts and are very expensive. However, some of the papers in this survey propose methods to make this semi-automated (as in semi-supervised learning approaches). Currently, certain medical and biomedical domains are using this approach. One advantage, as this approach continues to mature, is the high level and magnitude of knowledge it can represent.

Bendaould et al. (para. 8.1) use RSA (relational concept analysis) to create concepts through text-based ontology. Their methodology builds a glossary of terms from collections of text, by using available resources. They extract pairs and tuples using natural language processing (NLP) tools and build binary relation diagrams using RSA. This extracts relations using object properties and links to other objects. The ontology is used for (1) populating ontology, i.e. finding the most general class of an object, which is a subset of a class associated with a list of attributes, (2) comparing objects to see if they are in the same class, and (3) and finding the domain of a relation. Two approaches for building ontology from text (corpora) is (1.) using co-occurrence of terms in text and similarity measures to build a hierarchy of objects, (2.) using syntactic structure in describing an object near a verb to build a hierarchy of classes [3].

Valakos et al. (2006) use machine learning to build concept representations called allergens ontology. This includes selecting concepts, specifying attributes and relations, and filling properties with instances.
Shunsfard and Barforoush (2004) provide an automatic ontology building approach. Starting with small kernel, they extract knowledge from text.

Haun and Marko (2002) create concepts from text through machine learning of grammars and ontology.

Loh et al. (2003) use text mining to form concepts from phrases and analyzes their distributions. This is a knowledge discovery technique. It counts frequency of each concept, finds themes, compares centroids to common themes, and finds associations between concepts.

Rajaraman and Tan (2002) developed a concept frame graph, a conceptual knowledgebase for mining concepts from text. A learning algorithm constructs a concept map, and, through supervised learning, produces a knowledge base.

Pado and Lapata (2007) – provide a framework for semantic models that find context in terms of semantic relations and contain significant linguistic information. Dependency paths help their algorithm build a context of words of interest.


Dahab et al. (2008) – build ontology from natural domain text using a semantic pattern-based approach. A TextOntoEx tool extracts candidate relations from text and maps meaning representations used for constructing ontology. They provide an understanding of a domain in the form of concepts, relations, and axioms, using the OWL language to represent ontology.

Devedzic (2002) provides an overview of Ontology Engineering. Ontolingua, from Stanford University, is one of the most possible set of tools for developing ontology.

**Discussion:** Together, the above approaches provide a way to build Ontology from text. Bourbakis and Manaris (1998) present a Stochastic Petri-Net (SPN) based methodology to represent natural language (NL). The methodology performs semantic analysis of interpretations, image descriptions, and objects from images and pragmatic analysis for context (other sentences and paragraphs in a document). Documented capabilities of this methodology that go beyond other approaches investigated in this survey include using SPNs to capture context information for resolving ambiguities contained in natural
language, modeling both text and associated images, and the potential for automated increase of understanding by capturing multimodal information. Compared with Pado’s approach to context which captures content from neighboring sentences, Bourbakis captures contextual information from a number of available sources including not only neighboring sentences but multiple modes including imagery and database information. Table 9 below compares this methodology with other approaches in building ontology and NL Representation Schemes.

Table 2.8. Features for building ontology from text and NLP representation schemes

<table>
<thead>
<tr>
<th>Feature</th>
<th>F64</th>
<th>F65</th>
<th>F66</th>
<th>F67</th>
<th>F68</th>
<th>F69</th>
<th>F70</th>
<th>F71</th>
<th>F72</th>
<th>F73</th>
<th>F74</th>
<th>F75</th>
<th>F76</th>
<th>F77</th>
<th>F78</th>
<th>F79</th>
<th>F80</th>
<th>F81</th>
<th>F82</th>
<th>F83</th>
<th>F84</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Learning</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Mining</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Concept Map</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Knowledge Discovery</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Context</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Automatic</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 2.9. Comparing approaches to building ontology from text and NLP representation schemes

<table>
<thead>
<tr>
<th>Authors</th>
<th>F64</th>
<th>F65</th>
<th>F66</th>
<th>F67</th>
<th>F68</th>
<th>F69</th>
<th>F70</th>
<th>F71</th>
<th>F72</th>
<th>F73</th>
<th>F74</th>
<th>F75</th>
<th>F76</th>
<th>F77</th>
<th>F78</th>
<th>F79</th>
<th>F80</th>
<th>F81</th>
<th>F82</th>
<th>F83</th>
<th>F84</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bendaoud</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Valakos</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Shunsfard</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Haun</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Loh</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Rajaraman</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Pado</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Dahab</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Bourbakis</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

2.3.5 Hierarchical approaches to summarizing large documents

Yang and Wang (2008) use the hierarchical and redundancy sharing characteristics of fractal theory to increase performance of text summarization. Features include themantic, location, heading, cue (important words) to determine the score of sentence, and are used with a fractal summarization algorithm.
Hilberg (1997) uses neural nets (NN) to produce and store higher levels of abstraction representing word and sentence sequence in NN hidden levels. Their approach uses abstraction to eliminate redundancy. Connections and control codes are stored in the neural net. They use a Non-Markov model approach (i.e., deterministic) which yields more efficient abstraction. Segmentation cuts text into groups of words frequently found together.

Note: This is the only neural net approach found. Yet, finding word sequences is somewhat of an old approach, based on several of the papers surveyed. A more updated approach might involve using neural nets for abstraction along with semantic representation (such as found in Ye et al.’s approach. However, applying text understanding to neural nets is difficult and often involves a learning process that is computationally expensive.

In Bourbakis et al. (1999) redundancy elimination methodology, text similarity features used to trigger redundancy elimination of paragraphs include counts of characters (for size of paragraphs), words, and sentences, and histograms (frequencies) of characters and words, and marking beginning of sentences and end of paragraphs. When paragraph order is changed, two added features include starting word and word count (size) for each sentence.

Table 11 below compares this methodology with other approaches in a hierarchically summarizing large documents and redundancy removal or synthesis.

<table>
<thead>
<tr>
<th>Table 2.10. Factor definitions for hierarchically summarizing redundancy removal</th>
</tr>
</thead>
<tbody>
<tr>
<td>F85 Fractal Theory (redundancy)</td>
</tr>
<tr>
<td>F86 Hierarchical</td>
</tr>
<tr>
<td>F87 Salient (important) Features</td>
</tr>
<tr>
<td>F88 Thematic</td>
</tr>
<tr>
<td>F89 Location</td>
</tr>
<tr>
<td>F90 Heading</td>
</tr>
<tr>
<td>F91 Cue (pragmatic words)</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
2.3.6 Making generated text semantically meaningful through coherence and local order

Barzilay and Lapada (2008) use local coherence to increase readability and semantic meaning of automatically generated summaries from multiple documents. A two dimensional entity-grid (or matrix) shows binary (presence or absence) and type (subject, object, or neither) of discourse entities for each sentence. Unsupervised learning, with a corpus, is used to rank entities to generate improved sequential order of sentences.

2.3.7 Text generation methodologies

Dalianis (1999) uses aggregation before generating text summaries to eliminate redundant text in documents. They provide a set of aggregation rules that are based on corpus studies. Four types of aggregation include syntactic, elision, lexical, and reference type.

2.3.8 Document processing / (low level) understanding

Aiello et al. (2002) capture structural layout and logical order of text blocks within multiple documents and represents the information in connected graphs.

Feldman et al. (2003) use semantic analysis in their NLP system to mine biomedical literature.

Friedman et al. (2001) extract pathways from journal articles. GENIES uses a semantic grammar and syntactic knowledge interleaved with semantic and syntactic constraints.

2.4 Comparative Table of Methodologies

Table 2.11. Hierarchically summarizing large documents and redundancy removal or synthesis

<table>
<thead>
<tr>
<th>Authors</th>
<th>F88</th>
<th>F89</th>
<th>F90</th>
<th>F91</th>
<th>F92</th>
<th>F93</th>
<th>F94</th>
<th>F95</th>
<th>F96</th>
<th>F97</th>
<th>F98</th>
<th>F99</th>
<th>F100</th>
<th>F101</th>
<th>F102</th>
<th>F103</th>
<th>F104</th>
<th>F105</th>
<th>F106</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yang</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hilberg</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Bourbakis2</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

2.4 Comparative Table of Methodologies

Table 13 captures some of the main capabilities and approaches over a global comparison of papers throughout the survey. The intent of this comparison is to provide a collective picture of what main capabilities exist from the papers in this survey.
Table 2.12. Compatibility definitions

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Topics</td>
<td>C8</td>
<td>Overlap</td>
<td>C15</td>
<td>Answer Evaluation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2</td>
<td>Concepts</td>
<td>C9</td>
<td>Clusters</td>
<td>C16</td>
<td>Statistical</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3</td>
<td>Relations</td>
<td>C10</td>
<td>Quarry</td>
<td>C17</td>
<td>Word Sense</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4</td>
<td>Semantic</td>
<td>C11</td>
<td>Chains (Lex, Sem, Con)</td>
<td>C18</td>
<td>Large Document</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5</td>
<td>Hierarchical</td>
<td>C12</td>
<td>Hierarchical</td>
<td>C19</td>
<td>Multi-Document</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C6</td>
<td>Context</td>
<td>C13</td>
<td>Learning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C7</td>
<td>Aggregation</td>
<td>C14</td>
<td>Detect Themes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.13. Comparing key capabilities and approaches in survey

| Authors | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 | C9 | C10 | C11 | C12 | C13 | C14 | C15 | C16 | C17 | C18 | C19 |
| [1] Aiello |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [2] Barzilay |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [3] Bendaoud | √ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [4] Bourbakis1 | √ | √ | √ | √ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [5] Bourbakis2 | √ | √ | √ | √ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [6] Buitelaar |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [7] Cimiano |   |   | √ | √ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [8] Collins |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [9] Dahab |   | √ | √ |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [10] Dalianis |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [12] Feldman |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [13] Franceschet |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [14] Friedman |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [16] Hahn-1 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [17] Hahn-2 |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [18] Hilberg |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [19] Ko |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [20] Loh |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [21] Manabu |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [22] Marco |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [23] Meadche |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [24] Meng |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [25] Moens |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [26] Neumann |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [27] Neustein |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [28] Nomoto |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| [29] Pado |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
The above table shows capabilities from various approaches. Intuitively, as more pertinent information is captured, higher quality (minimal redundancy and maximum information coverage) should result. However, most of the performance qualities, as shown below, are not addressed. This may be due to the overall maturity of the technical area which is currently striving for accuracy as measured in the Document Understanding Conferences (DUC) that some of the authors reference. Performance time characteristics, other than computational complexity, appear to be a future effort, yet to be achieved.

### 2.5 Comparative Table of Maturity

For the following tables, features M1 through M15 provide an estimate of maturity with values 1 (low) through 9 (high). Due to the technical scope of the information in the papers covered by this survey, maturity was not directly present in the papers. Therefore, the maturity values in the following tables were extracted from the information present in the respective papers in this survey. Most all the papers addressed quality of resulting text (such as accuracy of summaries) and approaches rather than maturity. Most of the lower maturity values resulted from the lack of such information in the paper. In the tables below: complexity, memory required, training, and cost have an inverse effect on maturity indicators M1, M2, M11, and M12 with values calculated by $M_\_ = 10 - X$, where $M_\_$ is the maturity indicator and $X$ is the value extracted from the paper. The rest of the extracted values have a direct effect.
on their respective maturity indicators. The values in table 9 are weighted toward a user and developer perspective with weight values listed in table 8.

<table>
<thead>
<tr>
<th>Maturity Indicator</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complexity</td>
<td>M6</td>
<td>M7</td>
<td>M8</td>
<td>M9</td>
<td>M10</td>
</tr>
<tr>
<td>Memory Required</td>
<td>M2</td>
<td>M7</td>
<td>M8</td>
<td>M9</td>
<td>M10</td>
</tr>
<tr>
<td>Real Application</td>
<td>M3</td>
<td>M8</td>
<td>M9</td>
<td>M10</td>
<td>M11</td>
</tr>
<tr>
<td>Theoretical</td>
<td>M4</td>
<td>M7</td>
<td>M8</td>
<td>M9</td>
<td>M15</td>
</tr>
<tr>
<td>Scalability</td>
<td>M5</td>
<td>M8</td>
<td>M9</td>
<td>M10</td>
<td>M15</td>
</tr>
</tbody>
</table>

Table 2.14. Maturity of methodologies

<table>
<thead>
<tr>
<th>Author</th>
<th>Methodology</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aiello</td>
<td>Text Block Order</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Barzily</td>
<td>Coherence (fusion)</td>
<td>5</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Bendaud</td>
<td>RSA Relation</td>
<td>2</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>BourbakisI</td>
<td>SPN proposed</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>Bourbakis2</td>
<td>Redundancy Removal</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Cimiano</td>
<td>FCA Learning</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>Dahab</td>
<td>Concept from Text</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>Dalianis</td>
<td>Aggregation</td>
<td>3</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Feldman</td>
<td>Relationships</td>
<td>2</td>
<td>2</td>
<td>9</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Franceschel</td>
<td>Time Granularity</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Friedman</td>
<td>Extract Pathways</td>
<td>1</td>
<td>2</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>Guo</td>
<td>Event Indexing-Psy.</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>Hahn-1</td>
<td>Refinement</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>Hahn-2</td>
<td>Build Concept</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Hilberg</td>
<td>Neural Net</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ko</td>
<td>Extraction / Stat</td>
<td>6</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Loh</td>
<td>Knowledge Discovery</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>Manabu</td>
<td>Lexical Chains</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Marco</td>
<td>Document Order</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Meadche</td>
<td>Concept Learning</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Moens</td>
<td>Extraction-General</td>
<td>2</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Neusten</td>
<td>Sequence Analysis</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Nomoto</td>
<td>Diversity-Concepts</td>
<td>5</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>Pado</td>
<td>Semantic Space</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>Radev</td>
<td>Cluster / Topic</td>
<td>2</td>
<td>3</td>
<td>9</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Rajaraman</td>
<td>Knowledge Discovery</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>Reeve</td>
<td>Lexical Chains</td>
<td>2</td>
<td>2</td>
<td>9</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Shunsfard</td>
<td>ML to Build</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Silber</td>
<td>Lexical Chains</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 2.15: Comparison of maturity indicators
<table>
<thead>
<tr>
<th>Reference</th>
<th>Methodology or Extraction Tech</th>
<th>Point of View</th>
<th>Maturity Indicator:</th>
</tr>
</thead>
<tbody>
<tr>
<td>[36]</td>
<td>Stein</td>
<td>Clustering-Topics</td>
<td>M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12 M13 M14 M15</td>
</tr>
<tr>
<td>[37]</td>
<td>Valakos</td>
<td>ML to Build Concepts</td>
<td>1 2 7 9 4 5 2 3 5 1 8 5 3 7 1</td>
</tr>
<tr>
<td>[38]</td>
<td>Wang</td>
<td>Multilingual</td>
<td>6 6 6 3 9 3 2 2 3 1 1 4 5 2 2 1</td>
</tr>
<tr>
<td>[39]</td>
<td>Yang</td>
<td>Fractals – Hierarch</td>
<td>6 6 3 9 8 5 3 5 1 1 5 5 2 2 2</td>
</tr>
<tr>
<td>[40]</td>
<td>Yeh</td>
<td>Doc Concept Lattice</td>
<td>7 3 3 8 2 4 2 5 8 1 6 9 3 4 1</td>
</tr>
<tr>
<td>[41]</td>
<td>Zhou</td>
<td>LSA – Scoring-GA</td>
<td>1 1 7 9 2 2 3 7 5 1 2 2 6 2 5 1</td>
</tr>
<tr>
<td>[42]</td>
<td></td>
<td>ML - HMM</td>
<td>2 1 2 9 3 3 2 7 4 1 3 8 4 4 1</td>
</tr>
</tbody>
</table>

(Average)

Table 2.16. User and developer estimated weights for each maturity indicator

<table>
<thead>
<tr>
<th>Point of View</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
<th>M7</th>
<th>M8</th>
<th>M9</th>
<th>M10</th>
<th>M11</th>
<th>M12</th>
<th>M13</th>
<th>M14</th>
<th>M15</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>.6</td>
<td>.7</td>
<td>1</td>
<td>.2</td>
<td>1</td>
<td>1</td>
<td>.9</td>
<td>.6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Developer</td>
<td>1</td>
<td>1</td>
<td>.8</td>
<td>8</td>
<td>.6</td>
<td>.7</td>
<td>.8</td>
<td>1</td>
<td>1</td>
<td>.6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 2.17. Comparison of maturity indicators (user and developer weighted)

<table>
<thead>
<tr>
<th>Reference</th>
<th>Methodology or Extraction Tech</th>
<th>Point of View</th>
<th>Maturity Indicator:</th>
</tr>
</thead>
<tbody>
<tr>
<td>[19, 25, 30, 41]</td>
<td>Statistical / FCA / Hierarchy</td>
<td>User</td>
<td>3 3 7 1 3 4 4 4 5 2 4 6 3 4 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Developer</td>
<td>4 4 7 4 2 2 2 4 9 2 3 6 2 2 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>User</td>
<td>3 3 7 1 3 4 4 4 5 2 4 6 3 4 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Developer</td>
<td>4 4 7 4 2 2 2 4 9 2 3 6 2 2 2</td>
</tr>
<tr>
<td>[15, 21, 25]</td>
<td>Statistical / Hierarchy</td>
<td>User</td>
<td>3 3 6 1 4 4 5 4 5 2 5 7 3 4 2</td>
</tr>
<tr>
<td>[7, 35, 39, 41]</td>
<td>Statistical / Hierarchy</td>
<td>User</td>
<td>3 3 6 1 4 4 5 4 5 2 5 7 3 4 2</td>
</tr>
<tr>
<td>[21, 22, 35]</td>
<td>Statistical / Hierarchy</td>
<td>User</td>
<td>3 3 5 1 4 3 6 3 2 2 3 6 2 4 2</td>
</tr>
<tr>
<td>[13]</td>
<td>Statistical / Hierarchy</td>
<td>User</td>
<td>3 3 5 1 4 3 6 3 2 2 3 6 2 4 2</td>
</tr>
<tr>
<td>[7]</td>
<td>Statistical / Hierarchy</td>
<td>User</td>
<td>3 3 5 1 4 3 6 3 2 2 3 6 2 4 2</td>
</tr>
<tr>
<td>[15]</td>
<td>Statistical / Hierarchy</td>
<td>User</td>
<td>3 3 5 1 4 3 6 3 2 2 3 6 2 4 2</td>
</tr>
<tr>
<td>[7, 19, 23, 39]</td>
<td>Statistical / Hierarchy</td>
<td>User</td>
<td>3 3 5 1 4 3 6 3 2 2 3 6 2 4 2</td>
</tr>
</tbody>
</table>

39
The following lessons were learned: This survey revealed little commonality among the methodologies from each paper. However, the methodologies were able to be categorized into some general headings. Almost all of the maturity features in the above tables such as Complexity, Memory Required, Real Application, Theoretical, Scalability, Robustness, Multiple Applicability (or Multiple Domains), Portability, Implemented Prototype, Product, Training, Cost, Reliability, User Friendliness, Real-Time were not directly discussed in the papers and, therefore, indirectly abstracted from the information at hand. A resulting conclusion suggests that this area of natural language processing has not matured enough to provide this kind of product information.

Methodologies that were tested provided Precision and Recall results and some included complexity. Most were theoretical. According to a definition found on the Oracle web site, precision measures how well non-relevant information is screened (not returned), and recall measures how well the information sought is found.

Combining the capabilities from all the methodologies and approaches in this survey would include:

1. Extracting topics, concepts, and relations from text;
2. Representing semantic, hierarchical, context, aggregation, and overlap;
3. Forming clusters (representing sentences, paragraphs, documents, concepts, entities);
4. Finding similarities of topics, sentences, etc. to user queries;
5. Learning from collections of example text (corpora);
6. Detecting themes from text;
7. Using answers to questions (who, what, when, how) for evaluating text;
8. Using mainline statistical methods;
9. Detecting “word sense” to capture similar meanings to reduce information redundancy further;
10. Summarizing or capturing all key information from large documents;
11. Summarizing or capturing all key information from multi-documents.

If the above capabilities could be combined into a single approach, the result would produce significantly more accurate summaries. A few of the most capable methodologies show promise in providing an approximately optimized, minimum redundancy with maximum information coverage. However, more research needs to be performed in natural language understanding before maturity of these methodologies can transform into high volume, commercial products, as indicated by the above maturity table. Normally, providing the more capability to produce accurate text comes with a computational (time and space) complexity price, especially when heuristics are involved. However, breakthroughs in efficient graphical representations and algorithms can sometimes cut computational complexity significantly. Two papers in the survey claimed linear complexity. Some of the concept graphical approaches, chain, meta-chains, and hierarchical approaches provided impressive opportunities to compress and optimize resulting text. Finding an efficient methodology to accomplish all this would be a significant step toward eventual technical maturity.
III. GRAPH BASED SURVEY
GRAPH-BASED METHODS FOR NATURAL LANGUAGE PROCESSING-UNDERSTANDING – A SURVEY AND ANALYSIS

Abstract—This survey and analysis presents the functional components, performance, and maturity of graph-based methods for natural language processing (NLP) and natural language understanding (NLU) and their potential for mature products. Resulting capabilities from the methods surveyed include summarization, text entailment, redundancy reduction, similarity measure, word sense induction and disambiguation, semantic relatedness, labeling (e.g. word sense), and novelty detection. Estimated scores for accuracy, coverage, scalability, and performance are derived from each method. This survey and analysis, with tables and bar-graphs, offer a unique abstraction of functional components and levels of maturity from this collection of graph-based methodologies.

Keywords- Natural Language Processing, Natural Language Understanding, Graph Methods

3.1 INTRODUCTION

The vastness of information combined with the need for quick access to specific but comprehensive information has driven NLP and NLU research to provide the following capabilities: event resolution (ER), grammar annotation (GrA), information mining (IM), knowledgebase (K), labeling (Lab), novelty detection (ND), question/answer (QA), redundancy reduction (Red), semantic relatedness (SR), similarity measure (SM), summarization (Sum), textual entailment (TE), word sense disambiguation (WSD), and word sense induction (WSI). Over the past ten years, research in these areas has moved toward graph-based methods. The reduced complexity of graph methods over vector methods offers a more compressed and efficient concept representation of text. This paper presents a summary of such graph-based methods found in recent technical publications plus an analysis of their component functions and their maturity calculated from information found in the referenced papers. The goal of this survey-analysis is to provide the reader with enough detailed information along with tables and charts to capture the current state of the art in graph-based methods for NLP and NLU, including their component functions,
performance, and maturity. We conclude with an estimate of their near term potential for transforming the results of this research into products.

The following describes each of these research areas as NLP and NLU capabilities: (1) Summarization captures the main meanings of one or more documents down to a certain level of detail, or threshold. (2) Text entailment (TE), at the syntactic level, replaces all subsets of a text, with the encompassing text. At a semantic level, the concepts representing the text subsets are replaced with one larger, encompassing concept without losing any meaning of the original text subsets. The desired result is a shorter summary with no redundancy and no loss of meaning from the original texts. (3) By measuring similarity of text segments or their corresponding concepts, the resulting similarity measure (SM) can be used to merge clusters of similar concepts into a single concept. This compresses the resulting, summarized text without losing any of the representing concepts of the original text segments. (4) Word sense induction (WSI) identifies words with the same meaning. This meaning identification can increase the accuracy of similarity measure and yield a smaller text for summaries with less redundancy. (5) Word sense disambiguation (WSD) uses context around sentences to reduce or eliminate ambiguous sentences caused by words having multiple meanings. Once words with the same or similar meaning are identified using WSI, WSD can use all the words with the same meaning within a context of other words to reduce ambiguous text even further. The relation (or association) of two or more words within a context is often referred to as co-reference, i.e., the making of multiple instances of the same entity in text. (6) Semantic relatedness (SR) is the relation between concepts. Concepts, relations, and attributes are used to represent text at a semantic level and higher level abstractions of the meanings represented by a body of text. A potential goal for such higher level (or semantic) representation is natural language understanding. (7) Labeling is normally used to annotate parts of speech and senses (representing meanings) of words within a text. Labeling can be generated by manual, supervised, unsupervised, and semi-supervised methods. Supervised methods are trained using a manually labeled corpus (collection of text). Unsupervised methods are trained with a much larger collection of text. Semi-supervised methods train using a relatively small labeled corpus to derive labels for a much larger unlabeled corpus. These learning methods are used, within this context, to determine senses of words in unlabeled text. (8) Novelty detection uses some of the above capabilities to detect events of interest within a text.
The tables in this paper include the following information: (1) functional components from each methodology and resulting capability, (2) definitions of items used in calculating maturity [Keefer and Bourbakis, 2009] and of accuracy, coverage, performance, and scalability, (3) estimated importance of maturity factors to users, developers, and their average scores, and (4) estimated maturities based on the content of the papers and the definitions in the table 5. Bar charts include information on the average of estimated accuracy, coverage, performance, and scalability for each method and on maturities of methods segmented into regions.

Some methods such as disambiguation impact the effectiveness of other methods. For example, reducing redundancy improves the effectiveness of textual entailment and summaries. NLP or NLU systems may be based on different kinds of methods.

This paper provides functional component analysis and maturity information in an effort to inspire researchers to include more information on maturity, as the research progresses to bridge the gap to where more product information is publically available for graph-based methods in NLP/NLU applications. Finally the authors have excluded their method to avoid conflict of interest.

3.2 Methodologies Description

In this section we summarize a collection of graph-based methods, their performance and key features presented in their corresponding publications. The summary from each paper and a table on maturity definitions were used to estimate maturity numerical scores. Each method number (e.g., method 1) is used in each table and graph in this paper. Method numbers (e.g., method “1”) in paragraph headings are labeled with reference numbers (e.g., “[44]”) for easy reference.

3.2.1 Classification by clustering

Clustering, used primarily in methods 1, 2, 3, 4, 5, 14, and 17 by [44, 45, 46, 47, 48, 57 and 60] of this survey, provides node classifications, partitions, and pairing or grouping as it traverses a graph representation of the text or group of concepts. Chinese Whisper is an efficient clustering algorithm that works best in parallel or distributed architectures. Other clustering algorithms include: finding the medoid
of a set of nodes, covering, and finding the centroid of groups of nodes. Clustering is used in these methodologies for the following:

3.2.1.1 In method 1, Clustering is used to induce (or find) topically related senses from a graph of nodes representing nouns. Nouns paired by edges are weighted by the number of paragraphs, which contain co-occurrences of the nouns [44].

Method 1 (a through e) represents a group of methodologies presented by Korkontzelos, et al, 2009. In particular, method 1a (for word sense induction, WSI) connects each word with a target word if they co-occur one or more times in context. Graph clustering algorithms are used to induce topically related senses (meanings) of words. Method 1b (for WSI) forms clusters of word pairs. Within a cluster, each vertex stands for a word pair that co-occurs with target word and each edge represents the co-occurrence frequency. Method 1c evaluates graph connectivity by using eight measures to estimate free (i.e., tunable) parameters of word-pair. This process is called collocation in graph-based WSI methods. The method identifies the quality of induced clusters (standing for senses) with different parameter combinations, which aid unsupervised parameter estimation in WSI systems. Method 1d uses a corpus of nouns formed from words in context that reference target words. (The author states that these nouns have higher discriminative ability than verbs, adverbs, or adjectives.) Nouns are removed from the corpus when their distribution within their context falls below a threshold determined by a log-likelihood test. This process removes common (noisy or unimportant) nouns. The result consists of nouns that are more pronounced in determining a sense (or meaning) of target words they reference. Method 1e creates a collocation graph (representing word pairs) with the weight of each collocation calculated from the number of paragraphs, within the corpus, where each word pair exists. A smoothing technique compresses the sparseness of the graph by discovering new edges between vertices. A Chinese Whispers (CW) clustering algorithm (Biemann, 2006) is used to cluster the collocation graph, thus finding the senses of the target word. Each target word is then tagged with a sense (or meaning) according to the induced cluster [44].
**Performance:** The eight graph connectivity measures automatically selected parameter values that increased performance for both supervised and unsupervised evaluation schemes. Published scores included a recall of 84.8% [44].

**Discussion:** A log likelihood test (which finds the maximum of a concave curve) detects the distribution of common nouns that fall below a threshold so they can be weeded out of a corpus of nouns, leaving more pronounced nouns to determine sense (meaning) of target words. This statistical process has a functionality that is analogous to weeding out unpronounced signals with lower Eigen values using signal processing filters. Based on other papers in this survey, the Chinese Whispers clustering (or classification) algorithm is an efficient (parallel) algorithm.

3.2.1.2 In method 2, clustering finds sets of similar words and hierarchies of concepts by partitioning weighed undirected graphs [45].

Method 2 (Biemann, 2006) is a graph-clustering algorithm, called Chinese Whispers, which partitions nodes into classes of nodes resulting in a weighed, undirected graph. The Chinese Whisper algorithm is time linear in complexity to the number of edges in the graph. In document clustering, it finds sets of similar words and concept hierarchies. For word sense induction (WSI) applications, the edges represent word co-occurrence with other words in context. Graph clustering approaches, which are alternatives to traditional featured vector approaches, eliminate problems with sparse matrix with thousands of dimensions (arising from thousands of features, with only a few correlating). Instead of using distance metrics (such as cosine of angles between vectors), graph approaches encode similarity between objects as weights of edges between nodes.

Chinese Whisper algorithm partitions weighted undirected graphs. It finds groups of nodes that broadcast the same message to their neighbors (simulating an agent based social network). The algorithm follows:

```
initialize:
forall v in V: class(v) = i; //numbers nodes v

while changes:
forall v in V, randomize order; // changes above nodes to random order
```
\[ class(v) = \text{highest ranked class in neighborhood of } v; \] // [M6] (Biemann, 2006)

Node \( v \) is the node being investigated and node \( v_i \) is one of the neighborhood nodes of \( v \). The neighborhood of a node is represented by all the nodes directly connected to the node. In WSD, the neighborhood of nodes would represent words in context with the target word and a cluster would represent a sense [45].

**Performance:** The author states 1. “… randomized graph-clustering algorithm, which is time-linear in the number of edges.” 2. “… a very efficient graph-clustering algorithm is introduced that is capable of partitioning very large graphs in comparatively short time.” 3. “the power of the Chinese Whisper algorithm lies in its capability of handling very large graphs in reasonable time.” … 4. Most of its power lies in “size regions, where in other approaches’ solutions are intractable.” (Biemann, C., 2006)

Published scores: % Recall: for noun 75.5, verb 67.1, adjective 61.9; % Precision: for noun 90.0, verb 77.6, adjective 92.2. (Biemann, C., 2006)

**Discussion:** The clustering algorithm turns a graph representation of text into a more efficient graph representation used for finding (inducing) meanings (senses) of words. Nodes are compressed into clusters of similar words with co-occurrence information contained in the edges. The social network like functionality of the algorithm enables parallel execution for increased performance.

3.2.1.3 **In method 3 (Moens, et al, 2005) performs multi-document summarization by clustering objects representing similar sentences [46].**

This method detects the object of the cluster that corresponds to the sentence that most represents the sentences identified within each cluster. Since this methodology uses clusters and objects to detect similar sentences, this was included in this analysis of graph methodologies. Two clustering methods (covering and k-medoid) are used to detect sentences similar in content and select the most representative sentence. Covering minimizes the number of clusters (of objects representing sentences) so that the similarity between the most representative sentence object (or medoid) of a cluster and the rest of the objects (sentences) of each cluster exceeds a threshold. K-medoid partitions a set of objects into \( k \) clusters so that it maximizes the sum of similarities between the objects and their medoid. A threshold is
used in both methods to approximate the minimum or maximum to lessen the exponential complexity of these methods [46].

**Performance:** Both methods have exponential growth in computational complexity to the number of objects representing sentences from multiple documents. However, the authors state that the algorithms of this methodology are used for redundancy detection in their SUMMA (summarization) system.

**Discussion:** High complexity may be an issue for this methodology, even by lessoning its effect.

### 3.2.1.4 In method 4, clustering groups together similar documents [47].

Method 4 (Radev, et al, 2004) contains a centroid based clustering algorithm that groups similar documents into clusters. The clusters are used to classify relevant documents and to identify salient (or important) sentences in a cluster. The algorithm calculates a similarity measure of each document to each centroid. If the measured value is within a threshold, the document is included in the threshold. The similarity measure is calculated statistically by how many times a word appears in a document, multiplied by the percentage of all documents represented by the centroid that contains the given word. An event cluster keeps news articles from multiple sources in chronological order [47].

**Performance:** This MEAD system detects topics and tracks events. The authors conclusion states: “We found that MEAD produces summaries that are similar in quality to the ones produced by humans” [47].

**Discussion:** This methodology uses a clustering algorithm to calculate a similarity measure to detect similar documents.

### 3.2.1.5 Method 5 (Nomoto and Matsumoto, 2003) finds topical clusters, finds the main sentence to represent the cluster, and finds the concept diversity in text to reduce redundancy of its generated summary [48].
This is a summarization method that exploits diversity of concepts in text and reduces concept redundancy. The methodology finds diverse topical clusters from text, identifies the most important sentence for a topic cluster, and uses that sentence to represent the cluster, finds diversity of concepts from text, reduces their redundancy, and generates a summary from what is left. [48] (Nomoto and Matsumoto, 2003)

**Performance:** Experiments in paper only concentrated on semantic spaces that used a mapping function that maps dependency paths to words. More experimentation is planned. (Nomoto and Matsumoto, 2003)

**Discussion:** This is summarization approach which reduces redundancy at the concept level. Since the paper was written during its initial experimentation phase, no performance or maturity information was available.

3.2.1.6 **Method 13** (Guo, Y., and Stylios, 2005) creates clusters to build concept representations from text [56]. See paragraph 3.2.3.5 (Event Indexing) for further description.

3.2.1.7 **Method 17** (Cimiano, et.al., 2005) Clustering identifies themes of documents from closely related concepts by using a meta-thesaurus, such as UMLS, and clustering similar sentences [60]. Although Method 17 uses clusters, discussion is deferred to paragraph 3.2.4.4 (since its main contribution is for concepts, relations, and semantics.)

3.2.1.8 **Method 20** (Pado and Lapata, 2007) (Discussed in paragraph 3.2.4.7) finds diverse topical clusters from text, identifies the most important sentence for a topic cluster, and uses that sentence to represent the cluster [63].

3.2.2 **Methods 3, 6, 7, 8, and 9 by** [46, 49, 50, 51, 52] **use similarity measure to measure similarities of meaning (i.e. word sense) and to merge clusters of similar concepts.**

3.2.2.1 **Method 3** (discussed in paragraph 3.2.1.4 above) measures similarity between the most representative sentence object of a cluster and the rest of the objects (sentences) of each cluster to detect when a threshold is exceeded [46].
3.2.2.2 Method 6 (Moschitti and Zanzotto, 2008) uses machine learning to recognize textual entailment using tree pairs representing syntax in graph form.

A tree kernel function measures similarity between two trees (one representing a text string and the other a hypothesis text string) by counting their common substructures (or fragments). These kernels are applied to graphs consisting of tree pairs. This textual entailment approach emphasizes that kernel methods are needed to manage the large feature space of all possible syntactic tree fragment pairs used to form syntactic relations between text and hypothesis in learning algorithms [49].

**Performance:** This approach was too early in experimental stages for any performance or maturity results. Their experiments used Support Vector Machines which are required resources impacting maturity calculations [49].

**Discussion:** This is basically theoretical approach with no real prototype. It reveals the large space complexity from considering all possible syntactic tree fragment pairs.

3.2.2.3 Method 7 (Rao, et al, 2008) provides information on distance measures between nodes in a graph.

*Such distances are useful in measuring weighed relationships between vertices in a graph and what they represent such as semantic similarities, intended sense (or meaning), etc. The paper evaluates random walk algorithms, their performance over shortest path algorithms, and describes a new commute time measure. A pseudo inverse of the Laplacian is used to drive estimates for commute times between nodes on a graph. The method uses singular value decomposition to discard least significant eigenvectors. A random walk algorithm captures graph connectivity as well as path lengths.*

Algorithm description: First: Construct a graph, using every pair of words to compute similarity. For each word w in a graph, add an edge between w and all its parts of speech. For each part of speech, add edges to all its senses (or meanings). For each word sense, add edges to all of its hyponyms (“is a”) and hypernyms (“includes”). A random walk algorithm captures graph connectivity as well as path lengths and is used to calculate similarity. (PageRank is an example of a random walk algorithm used in search applications such as Google.) [50].

**Performance:** This methodology provides a new way to calculate weights of edges based on compute time which uses the pseudo-inverse of the Laplacian represented in a graph and discarding
least significant eigenvectors using singular value decomposition (SVD) to reduce noise resulting from the graph construction process. The resulting weights can be used to calculate relatedness or similarity for various types of NLP. The authors state in their conclusion: “The top-k eigenvectors and eigenvalues are computed using the iterative method by Lanczos-Arnoldi (using LAPACK) and the product of these matrices represents a smoothed version of the original Laplacian. The pseudo inverse is then computed on this smooth Laplacian. … Improvements are obtained by discarding bottom 20% of the eigenvalues.” (Rao, D., et al, 2008)

**Discussion:** Distances, in this context, can stand for various important components of NLP including similarity measure, meaning (or sense) induction, etc. The random walk algorithm used in this approach and others (within this survey) appears to be an efficient and popular algorithm yielding relatively good performance.

3.2.2.4 **Method 8 (Ambwani and Davis, 2010) uses weighed, directed graphs to model the range of influence of terms within a document and uses context to find the semantic relatedness of terms (within a sentence).**

This is an unsupervised method using no labeled data or training of parameters. A term is represented by a series of nodes in a graph. One node represents a sentence. Weights of edges represent semantic relatedness (or connection strength) between terms. The method segments topics by finding where cohesion in the graph is weakest representing the end of a terms influence within a document. It uses pointwise mutual information (PMI) to measure similarity (of topics) in nearby sentences. Topical coherence is computed by (1) using the relevance intervals (RI) of each term to model its influence and (2) using other terms in the neighborhood of two connected terms to increase their semantic relatedness. This reinforcing of related terms measures coherence between sentences. To calculate RI, a 325 million word corpus (7 years of New York Times) is used with a parts-of-speech tagger and a 50 word window to calculate Point-wise Mutual Information (PMI) used to measure relatedness between terms. RI is then used to segment a document into each term’s range of influence. The result is a weighed graph of connections across slices of the graph, each slice representing a sentence within a document [51].

51
Performance: Quotes from the paper related to performance includes: “… measure of semantic relatedness reinforces global co-occurrence statistics with local contextual information, leading to an improved representation of topical coherence.” This text segmentation “models topical coherence using long-range influence of terms and a contextually determined measure of semantic relatedness.” (Ambwani, G., 2010)

Discussion: This methodology uses weighted, directed graphs to find the range of influence of terms and how they are semantically related.

3.2.2.5 Method [9] (Minkov and Cohen, 2011) learns word semantic similarity measures from traversing a graph representation of a corpus of parsed text to extract word synonyms from text.

The method learns different graph walk models while traversing the graph and uses thes models to extract word synonyms. It constrains paths of its random walk to specific word types that match the type of the word of interest. It uses history of its random walk to estimate edge weights to constrain its path, thus pruning paths with probabilities lower than a predetermined threshold [52].

Performance: The authors executed and compared this graph method with two vector methods. Their results showed that “learning specialized graph walk models for different word types” yielded an increase in performance (in accuracy) [52].

Discussion: The learned models that resulted from these random walks, constrained by type information, reduced search space which would likely result in more efficient walks.

3.2.3 Methods 1, 10, 11, 12, 13, 14, and 15 by [44, 53, 54, 55, 56, 57, 58] use nouns and verbs, including phrases and modifiers, to capture information from text and represent it in graphs.

3.2.3.1 Nouns, represented as nodes with weighted edges to other nodes representing various senses (or meanings), help determine the intended meanings of words [44]. See 3.2.1.1 for parts a through c of method 1.

Method 1d (discussed in paragraph 3.2.1.1 above) uses a corpus of nouns formed from words in context that reference target words. (The author states that these nouns have higher discriminative ability than verbs, adverbs, or adjectives.) Nouns are removed from the corpus when their distribution
within their context falls below a threshold determined by a log-likelihood test. This process removes common (noisy or unimportant) nouns. The result consists of nouns that are more pronounced in determining a sense (or meaning) of target words they reference [44].

3.2.3.2 Subjects, verbs, objects, and verb modifiers are used in graphical relations to match given questions to corresponding answers derived from the text [53].

Methods 10a through 10c (Celikyilmaz, A., et al, 2009) support finding answers to questions using text entailment. The subject-verb portions of the methodology are implemented as a graphical structure. Method 13a uses a sentence semantic component analysis that encodes dependency trees (produced from the Stanford Parser) that include linguistic relationships specified as head (H) or verb, head-subject (H-S), head-modifier (or component) (H-M), and head-object (H-O). These features are used to match components of questions. Method 10b addresses Text Entailment Features from Paired Sentence Analysis. Semantic components (from M10a) of a question are compared with that of each sentence in the text to find an answer from the text. A graph based semi-supervised learning algorithm is used for entailment ranking. Method 10c is a graph summarization algorithm that uses a nearest neighbor approach to estimate representative vertices of a summary dataset. Closer vertices represent similar data points representing denser regions in the hyper-space. A group of similar data points are represented by a new vertex representing all vertices likely to have the same label [43].

Performance: The paper included subjective statements such as “can improve” but no quantitative statements indicating performance. For example, “semantic and syntactic features for textual entailment analysis has individually shown to improve the performance of the QA compared to the baseline.” and ... They “demonstrated that summarization on graph-based SSL can improve the QA task performance when more unlabeled data is used to learn the classifier model.” (Celikyilmaz, A., et al, 2009)

Discussion: Algorithms include a graph based semi-supervised learning algorithm for entailment ranking and a graph summarization algorithm that combines similar data points into single vertices (representing clusters of the data points).
3.2.3.3 A graph-matching algorithm uses verbs with their arguments and nouns with their modifiers to build syntactic-semantic graphs. These verbs and nouns come from sentences, clauses, and phrases. Semantic relations are formed from clauses and noun modifiers and relation classes, including causal, conjunctive, participant, spatial, temporal, and quality. The most important verb of a sentence is used as a head-word of the sentence [54].

3.2.3.4 Method 11 (Nastase, et al, 2006) is a graph matching algorithm that extracts pairs of syntactic units from text and assigns a semantic relation to each pair.

It incrementally learns from previous pairs and relations it has assigned plus some user feedback. It builds a syntactic-semantic graph as it assigns semantic relations. (See discussion.) This methodology matches graphs starting with a small amount of encoded (manually labeled) knowledge from a dictionary. It extracts pairs of text units (such as clauses, a verb and its arguments, a noun and each of its modifiers). It relies on previously processed examples (stored as graphs) to find the most appropriate relation for an extracted pair from text. It incrementally learns from semantic analysis.

The algorithm works with sentences, clauses, phrases, and words. Structural information, such as subordinate (embedded) or coordinate (equal level), is used to find pairs. Examples of subordinate include noun modifiers, verb arguments, and clause to a main clause in the sentence. Semantic relation candidates are found using three heuristics: word match, syntactic graph match, or dictionary of (manually built) markers. Semantic relations included three syntactic levels: inter-clause, intra-clause, and noun modifier relations. Six relation classes include: casual, conjunctive, participant, spatial, temporal, and quality.

A memory based learning accumulates knowledge to make predictions about semantic relations it finds to fit the current pair of input syntactic units (words, phrases, etc). A graph is built centered on a node representing a head word (most important word of a sentence such as a verb) and nodes connected to it that represent grammatically related words (such as modifiers or arguments). The edges are tagged with grammar indicators (subject, object, complement) and connective information (propositions, coordinators, subordinators, or nil). Node information such as part-of-speech is tagged at each node from the parser used by the system.
Graph matching starts by finding matches for head nodes, then their edges. Conditions for node matching include same part of speech, syntactic properties (such as the same verb sub-category structure, and same lemma (or grammar rule). Edge matching is guided by distance (or highest score from grammar and connection information listed above). When no matching graph is found, a simpler match of the word pair is sought from previously processed word pairs using the same process [54].

**Performance:** The authors have stated that “Graph-matching is most useful for assigning semantic relations between verbs and their arguments, but it also gives good results for inter-clause relations”. For noun-phrases, only noun-modifier pairs with syntactic structure were useful for semantic analysis. (Nastace, V. and Szpakowicz, S., 2006)

**Discussion:** This algorithm uses subject-verb-object as one of its structures. The purpose of the methodology is for graph matching of input text with previously processed text and dictionary markers such as propositions, coordinators, and subordinators associated with semantic relations. Semantic relations for this methodology are highly dependent on which domain is being used. The methodology uses an incremental approach in finding which semantic relations to use between concepts. Accuracy of the semantic representation increases as more of the text is processed.

**3.2.3.5 An In-Degree algorithm uses nouns, verbs, adjectives, and adverbs to find the most probable sense (meaning) of words based on their context.** Hyponyms found using similarity measures are included in determining word sense [55].

Method 12 (Guo, W., and Diab, 2009) performs word sense disambiguation (WSD) using an unsupervised learning approach with both WordNet and SemCor as lexical resources (linked word dictionaries containing synonyms, etc.) and a modification of the graph based In-Degree algorithm (described in Sinha and Mihalcea, 2007) to find the most probable sense of each word in a sentence based on context. The graph based In-Degree algorithm (with author’s extension of Sinha and Mihalcea, 2007) [13] creates a weighed graph with nodes representing senses of a word and edges with weights connecting the word to each sense. The sum of weights of the edges connecting a node (representing a word) is the In-Degree of the node (vertex). The sense with the maximum In-Degree value is the selected sense for the word. Different similarity measures are used: JCN (Jiang and Conrath, 1997) [14] for noun pairs, LCH
(Leacock and Chodorow, 1998) [15] for verb pairs, and Lesk (Lesk, 1986) [16] for within adjectives and adverbs and across different Parts of Speech (POS). All these similarity measures are normalized between 0 and 1 [0,1]. Similar senses are detected using hypernyms, hyponyms, similar attributes, similar verb groups, pertinymns, holonyms, and meronyms [55].

**Performance:** “… outperformed baseline and state of the art using unsupervised system (SM07) in overall accuracy across all data sets.” (Guo, W. and Diab, M., 2009) They modified In-Degree algorithm (introduced by Sinha and Mihalcea, 2007) and used linked dictionary resources: WordNet and SemCor. Published scores at SenseVal (SV) and SemeVal (SM) Conferences SV2: 0.629 SV3: 0.603 SM: 0.468. [55].

**Discussion:** Performs word sense disambiguation (or finding the most probable sense of each word in a sentence based on context) by creating a weighed graph with nodes representing senses of a word and edges with weights connecting the word to each sense.

3.2.3.6 Event indexing uses actors (based on current events), time of day (for present and previous events), causal (past event effects on current events), intention, and goals of actors (from previous events) to form relations between concepts. Causal connections consist of subject to object relations, formed from chains of verbs and nouns [56].

Method 13 (Guo, Y., and Stylios, 2005) performs event indexing and summarization using a cognitive psychology approach to creating clusters to build concept representations from text. Indices include actors in the current event, or times of occurrence of present or previous events, causal relationships of current to previous events, special relations between events, and intention relationships or relationship between goals of actors and the present event. An actor (called Protagonist) is a subject consisting of a noun phrase or pronoun referring to a subject or object or a noun phrase in one or more of the previous sentences. (Note: Actor or subject is the same as agent. Object is the same as patient.)

A causal relationship in this methodology is the relationship of a sentence to previous sentences or in the same sentence. (Note: A causal relationship might include the action an agent has on a patient, although not directly referred to in these terms.) Time information is gathered from stated times in
a sentence, tenses of verb phrases within a clause, or from the WordNet lexical resource containing
temporal relations. Location-related noun phrases are gathered as spatial information within a sentence.

More detail on actor (or Protagonist): The methodology resolves pronouns according to
gender and definite pronouns. Causality-indexing is used in the noun phrase level by creating causality
connections between each pair of subjects and objects, saves the connections, and joins them into chains.

The authors present the Causality-Indexing algorithm as follows:

“Algorithm: Causality_Indexing
Preconditions: All sentences have been parsed
Input: All sentences
Output: The causality connections and chains in both noun-phrase (NP) and clause levels
Steps:
Searching through all the documents to locate the verb synonyms of _cause_;
Confirming all the verb synonyms that function as a verb-phrase (VP) in a clause;
Locating the Subject and Object of each verb synonym;
Building up causality connection between each pair of subject and object;
Update local database for the causality connections;
Save all these connections and join them into chains;
Searching through all the documents to locate the noun synonyms of _cause_;
Locating all the noun synonyms that have predicative or relative clauses;
Locating these predicative and relative clauses;
Building up the causality connection between the predicative and relative clauses and any
previously given sentence in the same document.
Save all these connections and join them into chains. “ [56]
The authors use cluster of sentences with indexes, cluster filtering, reduction, and they
control the size of the outputted summary [56].

Performance/maturity: The authors describe each algorithm in detail, thus providing a
high availability score. Their methodology has been evaluated at the DUC 2003 Conference. The
methodology was new in 2004 [56].

Discussion: This event indexing (by actors, times of occurrence and causal, special, and
intention relationships) creates clusters to build concepts from text. Their causality indexing algorithm
produces chains of connectivity between subjects, verbs, and objects including their noun and verb synonyms. This approach appears to work best for text containing lots of events and actions. Descriptive text with little events and actions may not yield causal chains of any significance. However, most applications needing such a capability would find this type of approach useful, assuming it is mature enough to use.

3.2.3.7 Method 14 by [57] (Kozarevak, Z, 2012) learns terms that express cause-effect relations from text. It uses graph based metrics to re-rank extracted information and filter and delete erroneous terms to increase accuracy of cause-effect relations.

This is a bootstrapping method that uses patterns to represent cause-effect relations, learns while recursively building terms of a pattern, re-ranks what is extracted, then filters the terms. From a seed term and a recursive pattern as input, the method generates learned terms, which expand the pattern representing a knowledge expansion. The weighed sum of the incoming and outgoing edges of each node provides a ranking used with a threshold to delete erroneous terms in the expansion [57].

**Performance:** The author’s evaluation, using SemEval-1 Task-4 which classifies semantic relations between nouns, showed 89% accuracy after ranking 1500 terms.

**Discussion:** This approach learns cause-effect relations from patterns. The methodology appears intuitively straight forward and yields relatively high accuracy.

Method 15 by [58] (Liu, H., et.al., 2011) learns event rules by identifying context dependencies from parsed annotated text. It extracts events by matching subgraphs of sentences as it searches for a graph of an event rule.

For event recognition, this method searches for a subgraph that is isomorphic to dependency representations of previously learned event rules. The dependency of event rules is implemented as a labeled directed graph. The union of shortest dependency paths, each representing a training sentence, produces a graph representation of an event [58].

**Performance:** This method yields 41% to 52% F-score in detecting and identifying biological events [58].
Discussion: Although isomorphism normally yields a simpler model to work with, searching for an isomorphism is normally a computationally hard problem. Such cases can sometimes be partially resolved with approximate solutions. The authors merge rules across event types to increase precision. They investigated rule ranking to increase accuracy of learned rules.

3.2.4 Concept representations and their semantic relations, consisting of graphs and the methods that create and use them, are important steps toward natural language understanding (NLU). Methods 16, 19, 20, and 21 by [59, 62, 63, and 21] include the concept level of representation and manipulation as well as other NLP methods 8, 12, 17 and 18 by [51, 55, 60 and 61] that serve as interim steps toward the creation of this level of representation.

3.2.4.1 Method 8 by [51] (Ambwani and Davis, 2010) measures semantic relatedness using connection strength in the form of weights between node representations of terms. The neighborhood of terms (within a connection strength threshold) increases the semantic relatedness of other terms. This method is discussed in paragraph 3.2.2.4 (in similarity measure.)

3.2.4.2 Method 12 by [55] (Guo and Diab, 2009) incrementally learns semantic relations from semantic analysis and assigns semantic relations to pairs using a graph-matching algorithm. Its relation classes include causal, conjunctive, participant, spatial, temporal, and quality. The discussion is deferred to paragraph 3.4.3.4 (in subject-verb-object structures).

3.2.4.3 Method 16 by [59] (Morales, et al, 2008) maps text to concepts using ontology for concept extraction from text and uses (biomedical) domain knowledge from a meta-thesaurus to find synonym and “is-a” relations to summarize documents.
Clusters are used to identify closely related concepts as a theme in the document. This summarization method includes the following: a.) It represents a document as a graph consisting of concept nodes and relation edges. A UMLS Meta-Thesaurus is used to identify the correct concept for each term in the text and to disambiguate by extending concepts with their hyponyms (synonyms + is-a relationships). It merges sentence graphs into document graphs. b.) It clusters concepts with similar meaning (or theme) and recognizes themes in document graphs. The most central concepts in a cluster give sufficient and necessary information. c.) Sentences are selected based on similarity between sentences and clusters [59].

Performance: This method for summarizing biomedical literature represents a document as an ontology-enriched scale-free graph, the Unified Medical Language System (UMLS) concepts and relations. This way provides “a richer representation than the one provided by a vector space model.”

The authors have “identified several problems and some possible improvements. Firstly, as their method extracts whole sentences, long ones have higher probability to be selected, because they contain more concepts. The alternative could be to normalize the sentences scores by the number of concepts. Secondly, concepts associated with general semantic types in UMLS, as functional concept, temporal concept, entity and language, could be ignored, since they do not contribute to distinguish what sentences are significant. Finally, in order to formally evaluate the method and the different heuristics, a large-scale evaluation on the BioMed Corpus is under way …. [59].”

Discussion: The use of domain knowledge in this approach provides additional information to guide the mapping from text to concepts for summarizing documents. Intuitively, this added information could aid inference at various nodes in a concept graph to increase resulting summarization accuracies.

3.2.4.4 Method 17 by [60] (Cimiano, et al, 2005) acquires concept hierarchies using formal concept analysis that creates concept clusters.

This method automatically acquires the concept hierarchies from collections of text (corpus) that are tagged with parts of speech using an NLP parser called “LoPar.” The methodology creates a lattice of concepts using a concept clustering technique called Formal Concept Analysis (FCA).
Each vector contains verb dependencies in the form of verb-attribute pairs, consisting of subject, object, and prepositional phrase. A distance measure between vectors determines concept similarity used to detect and reduce redundancy. Output from the LoPar trainable parser generates vectors showing context syntactic dependencies between verbs and their corresponding attributes consisting of subjects, objects, and prepositional phrases. Using a process similar to text entailment, the generated concept representations are compressed to reduce redundancy [60].

**Performance:** Using FCA can cause the size of the lattice representation to be exponential to the size of the text and exponential time complexity. However, the authors state that their algorithms perform significantly more efficient than the usual FCA approach. Published scores: % Recall 65 and 37, % Precision 29 and 29, F-measure 44 and 38. [60].

**Discussion:** This methodology produces another concept lattice representation and includes concept hierarchy. Space and time complexity may be an issue.

### 3.2.4.5 Method 18 by [61] (Yeh, E., et.al., 2009) uses a random walk algorithm over a graph representation while measuring semantic relatedness of corresponding text.

It provides knowledge integration by using the linked word similarities of Wikipedia to detect similarity of meanings from different words found in text. It computes semantic relatedness between text pairs by building a graph representation of Wikipedia and using the link structure with different link types as added information for its semantic analysis. Then by using the random walk (or page rank like) algorithm to traverses the graph structure of Wikipedia, it maps each word to the graph creating a randomized vector. The PageRank algorithm computes a stationary distribution for each word in the text from the vectors. A score of each stationary distribution for each word pair provides a measure of vector similarity (or cosine similarity) of the pairs of text (based on all words of the respective text). The authors initialized the Wikipedia graph random walk using two methods, one based on dictionary and the other based on Explicit Semantic Analysis (ESA). They tested both methods and found ESA to perform better than the dictionary method and slightly better than other published methods using similar approaches with Wikipedia[61].
**Performance:** The authors state that “… even with a simple dictionary-based approach, the graph of Wikipedia links act as an effective resource for computing semantic relatedness. However, the dictionary approach alone was unable to reach the results of state of the art models using Wikipedia or using the same technique on WordNet.” … “by using ESA” (Explicit Semantic Analysis) “to generate the” … “distribution, we were able to introduce small gains using the random walk.” … “Performing random walks with personalized PageRank over the Wikipedia graph is a feasible and potentially fruitful means of computing semantic relatedness for words and texts.” [61].

**Discussion:** Wikipedia was used as a dictionary-like knowledge base, due to its enormous capture of knowledge. Other (perhaps more reliable) linked knowledge sources could be used in its place. This conceptual demonstration showed how a graphical representation of semantic relatedness could be determined and measured.

3.2.4.6 **Method 19 by [62] (Ye, et al, 2007) creates a concept lattice that indexes local topics within a topic hierarchy. Concepts in the form of topics are represented by nodes in the lattice.**

This method uses the concept lattice to extract text from multiple documents and generate an optimized summary. The concept lattice provides indexing of local topics within a hierarchy of topics. The topics, represented by nodes in the lattice, correspond to concepts that appear frequently throughout the original text or document(s) and sentences that contain these topics. The resulting summary contains an optimized set of local topics and a maximized coverage of concepts for a desired size of summary (number of sentences). This methodology produces a summary by extraction (containing key sentences from the original text) [62].

**Performance:** On-line complexity of using the WordNet lexical dictionary resource to compute all possible senses of each concept is reduced from O(n^2) to linear by computing sense similarity off line. The method uses a global selection strategy to minimize loss of information from the concept lattice. The methodology was run on document understanding Conferences (DUC) 2005 and 2006 evaluations [62].
**Discussion**: This concept lattice approach optimizes coverage of concepts within its graph representation, thus producing optimized summaries from this high level representation. It provides low complexity.

3.2.4.7 **Method 20** by [63] (Pado and Lapata, 2007) constructs semantic space models with annotated dependency relations and builds semantic context from a dependency graph, which maps dependency paths to words.

This semantic representation contains significant linguistic information. The algorithm builds a semantic context of words of interest from dependency paths defined on a dependency graph, specifies dimensions and provides inference over classes of basic elements, and specifies relative importance of different paths. [63]

**Performance**: Experiments in paper only concentrated on semantic spaces that used a mapping function that maps dependency paths to words. More experimentation is planned. [63]

**Discussion**: This methodology uses syntactic dependencies to build semantic space models containing words of interest in a graphical representation of knowledge. Since this was in the initial experimentation phase, no performance or maturity information is available.

3.2.4.8 **Method 21** by [64] (Rajaraman and Tan, 2002) generates a concept frame graph for knowledge discovery by constructing concepts and relations and uses a co-reference resolution algorithm to extract noun and verb clauses and phrases to generate synonym sets and relation parameters.

The concept frame graph guides the user in finding knowledge from text. The methodology constructs a conceptual knowledge base (CKB) consisting of concepts and relationships that are extracted from text and mines the knowledge base for new information. It pre-processes each document to a body of text, extracts all entities using a co-reference resolution algorithm, extracts 3-tuples consisting of noun clauses, verb clauses, and extended forms of noun phrases and verb phrases which are used to generate the Synset and Rels parameters of the concept frames. Their parts-of-speech (POS) tagger tags the extracted text and their rule-base extracts noun-verb-noun (NVN) 3-tuples. Their sense disambiguation algorithm
uses WordNet and provides separate synset parameters for each word sense of every word by computing a
distance measurement to pick the correct word sense [64].

**Performance:** The authors’ evaluation of their methodology was in progress at the time of
the paper. Experiments were performed using small test cases at that time but the authors planned to use
larger test cases.

**Discussion:** This concept frame graph approach is used for knowledge discovery from text.
Since the paper was written during its initial experimentation phase, no performance or maturity
information was available.

### 3.2.5 Other methods 22, 23, 24, 25, 26, 27 by [65, 66, 67, 68, 69, 70] provide graph based ranking of
data in labels, find closest sense of words in labels, detect new information, generate grammars for a
given text, resolve references to events, and generate graphs from word co-occurrence in text.

#### 3.2.5.1 Method 22 by [65] (Mihalcea, 2005) uses a ranking algorithm to label the current meaning of
each word represented by a node while random walking over a graph.

*It consists of labeling sequence of data (words) by analyzing label dependencies. It is a sequence data
labeling method applied to an all-words word sense disambiguation (WSD) problem by labeling the most
likely sense (or meaning) of each word while random walking over the graph representing the text. An
iterative graph based ranking of the labels uses a graph based ranking algorithm called Page-Rank (Brin
and Page, 1998) [65].*

**Performance:** The algorithm was illustrated and tested on an unsupervised word sense
disambiguation problem, targeting the annotation of all words in unrestricted texts. Through experiments
performed on standard sense annotated data sets, the graph-based sequence data labeling algorithm was
shown to significantly outperform the accuracy achieved through individual data labeling, resulting in a
statistically significant error rate reduction of 10.7%. The disambiguation method was also shown to
exceed the performance of previously proposed unsupervised word sense disambiguation algorithms.
Moreover, comparative results obtained under various experimental settings have shown that the algorithm
is robust to changes in classification granularity and context size. Published scores: % Recall and %
Precision: 54.2 fine grain, 55.3 course grain [65]. (Mihalcea, R., 2005)
**Discussion:** This graph based ranking algorithm statistically finds and labels (or tags) the most likely senses of words through dependency analysis.

**3.2.5.2 Method 23** by [66] (Agirre, E., et al, 2006) uses unsupervised learning by training on a large sample of text (corpora) to find the sense (meaning) of words for labeling nodes of a graph.

It consists of two graph algorithms for unsupervised induction and tagging of word sense (or intended use of words) based on corpora (sample of text). They found that a small sample of nouns was enough to optimize parameters of these algorithms for efficient word sense induction (WSI) [63].

**Performance:** Author’s discussion on efficiency: “Regarding efficiency,” their “implementation of HyperLex is extremely fast. Doing the 1800 combinations takes 2 hours in a 2 AMD Opteron processors at 2GHz and 3Gb RAM. A single run (building the MST – Minimum Spanning Tree of the hub of senses connected to target words, mapping and tagging the test sentences) takes only 16 sec.” [66].

**Discussion:** Nouns appear to contain the most helpful information for training algorithms to efficiently determine senses (meanings) of words in a sentence. It also needs more detail on how the two graph algorithms work.

**3.2.5.3 Method 24** by [67] (Gamon, 2006) detects novelty (text, containing new information, compared to what is already found) at the sentence level by analyzing feature sets from graph representations of sentences.

This methodology uses sentence-level term distances (or KL divergence, i.e. relative entropy) and point-wise mutual information (PMI) to produce graphs representing sentences.

**KL divergence and Point-wise Mutual Information are calculated as follows:**

“KL divergence = \[\sum_{w} p_d(w) \log \left( \frac{p_d(w)}{p_R(w)} \right)\]” where \(w\) = word shared between document \(d\) and document (set) \(R\). \(p_d\) and \(p_r\) = non-zero probability distributions of words in \(d\) and \(R\).

“PMI(i,j) = \log_2 \left[ \frac{P(i,j)}{P(i)P(j)} \right]” where \(P(i,j) = \text{number of sentences containing both words } w_i \text{ and } w_j\). \(P(i)\) and \(P(j) = \text{number of sentences containing words } w_i \text{ and } w_j \text{ respectively.}\)

*The weight of an edge (between vertices \(I\) and \(j\)) is “\(wt_{ij} = \left[ 1 + \frac{PMI(i,j)}{d^2_{ij}} \right]\)”* (Gamon, 2006) [67]
Features from a term distance (TD) graph are based on strengths and number of connections between words. When novelty (or new information) is detected, the corresponding new sentence graph is added to the existing (background) graph. When the information from the input graph already exists, the weight of the existing edge representing the information is incremented. The more a graph changes when information from input sentences is added, the more likely is the information of the input sentence new. 21 graph based features are used to assess the amount of graph change (i.e., new information) when a new sentence is added.

A text rank metric for text, similar to the PageRank metric for the web, calculates a weight for each vertex of undirected graphs. This corresponds to measuring the importance of the new information by detecting the change in scores distributed among the features [67].

**Performance:** (F-measure: 0.618.) “A highly connected term-distance based graph representation, with the addition of point-wise mutual information, is a computationally relative cheap approach.” (Gammon, M., 2006) Published scores: F-measure 0.618.

**Discussion:** Kullback-Leibler (KL) divergence (also known as relative entropy) is a difference between probability distributions that can be approximated as a distance measure. Point Wise Information (PMI) measures how much one word is associated with another. In this paper, it measures how much certain words occur together in a text. The result provides strengths of connections and number of connections between words in a graph representation.

**3.2.5.4 Method 25 by [68] (Muresan, 2008) uses learning to map text to graph based meaning representation based on grammar induction.**

This method uses a Lexicalized Well-Founded Grammar (LWFG) which is context free but extended with a partial ordering of non-terminals (due to the natural ordering of words in text), has a syntactic-semantic representation for each string, and has rules with two constraints: its composition must be semantic and ontology based. A small set of annotated grammar examples is used as training data for learning the grammars. A cover set algorithm induces the grammar, thus adding a new constraint learned from the representative grammar example for each step until all examples are covered (or used) [68].
Performance: This methodology appears to be in the inception/theoretical phase and therefore has not reached a high enough level of development for its performance to be measured [68].

Discussion: This approach consists of a learning framework that maps natural language to an ontology-based graph that represents the corresponding semantic meaning of text. The meaning of a text uses grammar induction and all questions and their answers that can be derived from the text.

3.2.5.5 Method 26 by [69] (Herdagdelen, et al, 2009) compairs two approaches for finding semantic relatedness.

This method provides a comparison of random walk with singular value decomposition (SVD) and vector space modes. The interest of this methodology was in measuring semantic relatedness. The random walk process transforms initial probability distribution of nodes to a final probability distribution based on a single end point node. Both directed and undirected graph approaches are considered. Indirect measures included Jensen/Shannon divergence (a variant from Kullback/Leibler divergence between probability distributions), cosine similarity, and dot product. They used a 2-billion word corpus (ykWaC), a dependency parser MINIPAER, and considered only co-occurrences of target word pairs connected with dependency paths [69].

Performance: Information related to performance includes: “A two step random walk model, based on indirect measures with dot product, consistently outperforms both SVD-based and plain vector models in terms of relative performance,” … “on very small datasets.” (Herdagdelen, A., et.al., 2009)

Discussion: This paper provided comparative information on the random walk process, a useful and efficient process used in other methodologies in this survey.

3.2.5.6 Method 27 by [70] (Chen and Ji, 2009) resolves references within two or more sentences to events.

It provides two methods of computing co-reference matrices used for event co-reference resolution in extracting content from text. (Event co-reference consists of references to the same event from two or more sentences and its resolution is resolving or detecting classes of these references.)
methodology forms clusters in graphs to represent separate events so it can resolve (or collect references to the same event) by using Entity-Constrained-Mention (ECM) F-Measure [70].

**Performance:** The paper investigated event co-reference resolution. It used two methods for computing a co-reference matrix: (1) Computing a co-reference formula, and (2) Applying a maximum entropy model. A high-performance event extractor is required for resolving events. Published scores: F-measures 0.8363, 0.8312 [70].

**Discussion:** Co-reference is an important component of NLP since it is used in finding word associations in context. In this case, it is used to find references to events within the text and to improve its accuracy (or resolution).

3.2.5.7 **Method 28** by [71] (Biemann, 2010) constructs a word co-occurrence graph for each target (important) word using co-occurrence statistics from a large set of text (20 million sentences from the New York Times corpus).

Unsupervised learning from the corpus provides graph clusters with features that provide the kind of information normally used in systems with supervised learning. Representative graphs with different parameters are generated from the corpus. The target node in the graph (representing the target word) is directly connected to each node that represents a word significantly co-occurring with the target word in a sentence. Weights for each edge is calculated from the log-likelihood of the co-occurrence of the connected words (derived from relative frequency gathered from the corpus text). Graph parameters include size (number of co-occurrences or connections) and density. The Chinese Whispers algorithm (see M6) clusters the neighborhood of the graph by partitioning the nodes and representing each cluster as a word sense, thus providing word sense induction [71].

**Performance:** This methodology builds a classification model. A machine learning algorithm (similar to a Naïve Bayes classifier) picks the most suitable cluster features per target word. It uses a Wikipedia dump (60 M sentences) for cluster feature induction and a New York Times corpus (20 M sentences) for training. The authors state that “Co-occurrence cluster system outperforms baseline” (2nd best in SemEval 2007 lexical sample task). “WSD setup is competitive … using minimal linguistic
preprocessing and no word sense inventory information … except for training examples.” (Biemann, C., 2010) System assigned “acceptable substitutions in over 91% of cases, …, with 5.4% error rate.” (Biemann, C., 2010) Published scores: % Precision 88.5.

**Discussion:** Co-occurring words within a number of sentences within a large corpus of text can transform into a graph and adjust graph producing parameters. The Chinese Whispers algorithm (also used in methods 1, 6, and 13) forms clusters from the graph to represent different meanings or word senses, thus providing word sense induction.

### 3.2.5.8 Method 29 by [72] (Rao and Yarowsky, 2009) uses a map reduce algorithm in a semi-supervised (partially labeled) graph representation of natural language text for label propagation.

The authors have modified the algorithm to be scalable for very large graph representations with linear processing time and localized in memory space so it is suitable for parallel processing, making its memory space also scalable. The authors show the importance of label propagation by providing a scalable parallel algorithm that can provide ranking of nodes to derive lexical relatedness between terms applied to disambiguation, paraphrasing, question answering and machine translation. It also provides polarity induction for sentiment mining applications. The graph based map-reduced algorithm is adapted for large scalable graph representations of natural language text and applied to label propagation using a semi-supervised (with nodes initially 20% labeled and 80% unlabeled) classification approach. Label propagation is a random walk methodology. The algorithm provides linear computation and localized space for efficient memory allocation/de-allocation. Random walk captures connectivity structure besides distance measure. Label propagation uses the small percentage of labeled nodes in a large graph to define the probability over the labels (including calculating probability distribution among unlabeled nodes). Random walks over a weighed undirected graph calculates minimum of the Laplacian by solving a system of linear equations (instead of the computational complexity of using dense matrices that represent pseudo-inverses of the Laplacian), thus arriving at a distance measure of the graph with linear and parallelized computations. The authors associate label propagation with the popular PageRank random walk model which can quickly jump to its initial state, thus saving computation time. The algorithm is parallelized so it
can use local information, thus using constant memory and reducing the required data access to contiguous chunks of memory. This significantly reduces data accesses (number of memory reads/writes). [72].

**Performance**: Random walks for classification involves constructing the graph Laplacian and using its pseudo-inverse as a kernel. However, for very large graphs, the pseudo-inverses are dense matrices requiring $O(n^2)$ space which can be prohibitive. Author’s alternative: Their iterative label propagation algorithm can parallelize using a map reduce model (introduced by Dean and Ghemawat, 2004) [34] with its solution derived from a set of linear equations. The algorithm uses fixed memory regardless of the size of the graph and scales linearly in the size of data and the number of processing elements in the cluster. Published scores: F-measures for nouns 0.5853, verbs 0.8340, adjectives 0.7295. [72].

**Discussion**: The methodology ranks nodes to derive lexical relatedness between terms which makes it useful for disambiguation, paraphrasing, question answering and machine translation. The label propagation it produces transforms a partially labeled graph into a more fully labeled graph which could be interpreted as a semi-supervised approach in automatically applying word sense labels for disambiguation. The treatment of scalability issues for large data applications by scaling graphs linearly reduces space complexity significantly.

### 3.3 Functional components of graph-based methodologies

Table 1 dissects the information from each method into its functional components and method capabilities to help present a functional analysis encompassing all NLP and NLU methods summarized in this survey. It shows functions, capabilities, names of algorithms, and type of learning for each methodology. Major functions of interest from this table are discussed within the conclusions. Capabilities (abbreviated in table 1 and described in the introduction) include event resolution (ER), grammar annotation (GrA), information mining (IM), knowledgebase (K), labeling (Lab), novelty detection (ND), question/answer (QA), redundancy reduction (Red), semantic relatedness (SR), similarity measure (SM), summarization (Sum), textual entailment (TE), word sense disambiguation (WSD), and word sense induction (WSI). The 85 individual functional components collected from all the methodologies discussed
in this survey and listed in table 1 can be categorized into the functional areas. Major functional areas of interest include: Causality (including Noun-Verb-Noun relations), Clusters, Concept Representations (including Semantic Representations), Events, Sense (or Meaning) Determination and Distance Measure (of similarities, terms, meanings, etc.).

Table 3.1. Component Functions

<table>
<thead>
<tr>
<th>Methodologies: =&gt;</th>
<th>(Method numbers)</th>
<th>Group of Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authors: =&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Capabilities: =&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functional</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Components:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Causal relationship, connections, indexing

Chinese Whisper

Classification

Cluster - Concepts - Similar Meaning

Cluster - Minimize (Partition k clusters)

Cluster - paired words

Cluster - Topics

Concept Graph

Concept Lattice - Topics in lattice

Concepts (nodes)
<table>
<thead>
<tr>
<th>Methodologies: =&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group of Methods</td>
</tr>
<tr>
<td>Relations (edges)</td>
</tr>
<tr>
<td>Conceptual Knowledge Base</td>
</tr>
<tr>
<td>Connectivity of graph</td>
</tr>
<tr>
<td>Co-occurrence</td>
</tr>
<tr>
<td>Co-reference (Find Word Associations)</td>
</tr>
<tr>
<td>Coverage - Maximize - of Concept</td>
</tr>
<tr>
<td>Covering (for clustering)</td>
</tr>
<tr>
<td>Document - Similar, Category, Cluster</td>
</tr>
<tr>
<td>Event - Actors, Time, Causal Relations</td>
</tr>
<tr>
<td>Event - Location (Spatial) - Noun-Ph</td>
</tr>
<tr>
<td>Event Indexing</td>
</tr>
<tr>
<td>Events - sentences - time of events.</td>
</tr>
<tr>
<td>Hubs - (WSD) - from edges</td>
</tr>
<tr>
<td>In-Degree Algorithm</td>
</tr>
<tr>
<td>KL - Divergence</td>
</tr>
<tr>
<td>K-Medoid (for clustering)</td>
</tr>
<tr>
<td>Laplacian Inverse</td>
</tr>
<tr>
<td>Laplacian Minimum (Linear Equations)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
<th>08</th>
<th>09</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methodologies: =&gt;</td>
<td>01</td>
<td>02</td>
<td>03</td>
<td>04</td>
<td>05</td>
<td>06</td>
<td>07</td>
<td>08</td>
<td>09</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>------------------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>Map-reduce</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mutual Information</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(also, see PMI)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nearest Neighbor</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noun, Verb,</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Adjective, Adverbs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(pairs)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nouns in context</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Noun-Verb-Noun</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>phrase</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page Rank</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pointwise Mutual</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information (PMI)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relative Entropy</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random Walk</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select sentence</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>similar to cluster</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Semantic</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Equivalence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recognition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sense</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Similar - Verb</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>groups</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Similarity</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smoothing -</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>compress sparse</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subject-Verb</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Term distance</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(sentence level)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

73
### Methodologies:

<table>
<thead>
<tr>
<th>Methodologies: =&gt; Method numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
</tr>
</tbody>
</table>

### Group of Methods

<table>
<thead>
<tr>
<th>Text Rank Algorithm -</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theme Recognition</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Thesaurus (to identify concept)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Topic - Hierarchy - Index locally</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Topic - related</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Verb - Attribute Pair</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Verb Dependency Vector</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Weighed Graph</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Weights - Log Likelihood</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

### 3.4 Definitions

The table 2 titled maturity definitions (from Keefer-Bourbakis, 2009) [78] shows definitions for each maturity element used for calculating maturity. Numerical values for each maturity element are extracted from information in technical papers describing each method. The purpose of each maturity table is to show the relative state of the research area as a whole. Included in the table of weighed maturity features are estimated Accuracy, Coverage, Performance, and Scalability numbers from both developer and user perspectives. The tables and charts in this survey contain estimated scores based on information available in this area and are not intended to reveal how any one methodology is better than another.
Table 3.2: Maturity Definitions (Keefer and Bourbakis, ATRC, DIP TR-2009) [78]

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Availability - The ability to obtain/implment the system based on the description of the method expressed in mathematical formula, pseudo-code, or compiled code. A higher score indicates that a satisfactory amount of information is presented in the description to replicate the system. For example, a system with a score of 10 will contain a clear description of the method and code that could be implemented; whereas a system with a score of 5 may only have a mathematical formula and short process description.</td>
</tr>
<tr>
<td>Co</td>
<td>Cost - The amount of money needed to use and/or implement the system based on the description provided. This score reflects the cost of equipment as well as implementation complexity.</td>
</tr>
<tr>
<td>FI</td>
<td>Further Improvements - The methodology has the potential for further enhancement. A higher score indicates that a methodology can be improved upon, whereas a system with a lower score is considered more mature and less likely to be improved upon.</td>
</tr>
<tr>
<td>MC</td>
<td>Model Complexity - Complexity of model used in the methodology. For example a system utilizing a neural network or wavelet is considered more complex than one that uses a run length smoothing algorithm.</td>
</tr>
<tr>
<td>O</td>
<td>Originality - The methodology is based on original algorithms and/or mathematical operations; or the synergistic combination of simple methods composing a new method. A method that is referenced in the literature as original is given a higher score than one that builds on another method.</td>
</tr>
<tr>
<td>P</td>
<td>Prototype - The methodology has been successfully implemented at the experimental stage and produced desirable results. Scores for this aspect were also affected by the results presented. A paper that presented comparative results scored higher than one that presents an illustrative example.</td>
</tr>
<tr>
<td>RP</td>
<td>Released Product - The methodology has been implemented in a commercial setting. This aspect has a value of either 1 or 3, where the few methods/systems that have been utilized in a commercial setting are given a slight advantage over others.</td>
</tr>
<tr>
<td>Re</td>
<td>Reliability - The methodology produces expected results under normal operating conditions.</td>
</tr>
<tr>
<td>Ro</td>
<td>Robust - The methodology produces acceptable results under extenuating circumstances. This score is based on the features of the methodology as compared to methodologies in a similar category.</td>
</tr>
<tr>
<td>Sp</td>
<td>Speed - Reported processing time for sample tests. Note that some authors do not report performance metrics. For these we give a score of 3 out of 10.</td>
</tr>
<tr>
<td>U</td>
<td>Usability - The methodology offers a user-friendly interface so that the user can work easily with it. Systems that require no user input are given a higher score than those that require input parameters or training data.</td>
</tr>
</tbody>
</table>
| M       | Maturity - A measure that combines the scores of the different aspects.  
\[
Maturity = U + O + ((A*P*RP) + (Re*Ro*Sp))/(Co*FI*MC)
\]
Table 3.3: Accuracy, Coverage, Performance, Scale

<table>
<thead>
<tr>
<th>Acc</th>
<th>Accuracy – Captures intended meaning (sense) at concept and lexical representations and generated text.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cov</td>
<td>Coverage – Handles nearly all senses of a word. Disambiguates all (at least, important) words. Captures all if not most of meaning in generated summary or text entailment.</td>
</tr>
<tr>
<td>Per</td>
<td>Performance – Resulting scores in precision, recall, and f-measure from testing Conferences such as Document Understanding Conferences (DUC), Disambiguation Evaluation (SensEval), and Semantic Evaluation (SemEval). Values are normalized to a compatible scale with other factors in the table.</td>
</tr>
<tr>
<td>Sca</td>
<td>Scalability – Handles multiple and large documents. Handles large database or uses large dictionary.</td>
</tr>
</tbody>
</table>

**App**

Application: **Ent** (Entailment), **ER** (Event Resolution), **GrA** (Grammar Annotation), **IM** (Information Mining), **K** (Knowledgebase), **Lab** (Labeling), **ND** (Novelty Detection), **QA** (Question/Answer), **Red** (Redundancy Reduction), **Sim** (Similarity Measure), **SR** (Semantic Relatedness), **Sum** (Summarization), and **WSD** or **WSI** (Word Sense Disambiguation or Induction).

### 3.5 Maturity Tables and Charts

Table 4 provides estimated importance factors for developers and users. These factors are used to weigh maturity and other (accuracy, coverage, performance, and scalability) scores in terms of their average importance. What’s important to developers (such as mathematical descriptions or pseudo-code) is sometimes different from what’s important to users (such as user friendliness) and, therefore, need to be weighted differently. However, both developers and users want the maximum accuracy, coverage, performance, and scalability since they show favorable results from developers’ efforts and help users achieve their requirements.

Table 3.4: Developer and User Weights

<table>
<thead>
<tr>
<th>Method</th>
<th>A</th>
<th>Co</th>
<th>Fl</th>
<th>MC</th>
<th>O</th>
<th>P</th>
<th>RP</th>
<th>Re</th>
<th>Ro</th>
<th>Sp</th>
<th>U</th>
<th>Acc</th>
<th>Cov</th>
<th>Per</th>
<th>Sca</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>10</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>User</td>
<td>7</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Average Weight / 10</td>
<td>0.85</td>
<td>1</td>
<td>0.5</td>
<td>0.6</td>
<td>0.5</td>
<td>0.8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5 contains an estimated maturity of each method, which is calculated from maturity components defined in table 2. Table 5 also includes estimated accuracy, coverage, performance, and scalability values with the column headings defined in table 3. The acronyms in the column headings are also defined in tables 2 and 3.
Table 3.5: Maturity of Methods (Estimated, Un-weighted) plus their Accuracy, Coverage Performance, and Capacity

<table>
<thead>
<tr>
<th>Method*</th>
<th>A</th>
<th>Co</th>
<th>Fl</th>
<th>MC</th>
<th>O</th>
<th>P</th>
<th>RP</th>
<th>Re</th>
<th>Ro</th>
<th>Sp</th>
<th>U</th>
<th>M</th>
<th>Acc</th>
<th>Cov</th>
<th>Per</th>
<th>Sca</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>17</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>70</td>
<td>5</td>
<td>5</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>9</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>13</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>12</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>8</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>14</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>1</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>14</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>12</td>
<td>6</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>1</td>
<td>5</td>
<td>8</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>13</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>9</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>21</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
<td>1</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>8</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>11</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>9</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>9</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>14</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
<td>3</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>13</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>16</td>
<td>9</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>11</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>11</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>10</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>18</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>8</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>14</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>2</td>
<td>9</td>
<td>5</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>5</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>22</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>1</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>55</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>23</td>
<td>10</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>8</td>
<td>4</td>
<td>18</td>
<td>8</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>24</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>4</td>
<td>23</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>9</td>
<td>1</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>11</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>26</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>9</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>27</td>
<td>10</td>
<td>1</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>14</td>
<td>3</td>
<td>2</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>28</td>
<td>6</td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>14</td>
<td>9</td>
<td>3</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>29</td>
<td>10</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>2</td>
<td>18</td>
<td>5</td>
<td>5</td>
<td>7</td>
<td>9</td>
</tr>
</tbody>
</table>

The following bar charts (Figures 1 and 2) show plots from values of table 5 containing maturity components, resulting maturities, and other (accuracy, coverage, performance, and scalability) values. Figure 1 shows maximum, average, and low lines of reference.
Figure 3.3.5-1

Figure 2 shows the desired and average lines of reference. Since desired values of both developer and user were at the maximum (best possible value), all of the methodologies fell significantly below the desired score for accuracy, coverage, performance, or scalability values.

Figure 3.3.5-2

Table 6 contains weighted maturity scores. The scores in table 5 are multiplied by the average of the developer and user levels of maturity importance from table 4. Thus, table 6 is weighted toward developer and user points of view. The symbols in the column headings are defined in table 2. Maturity values and normalized maturity values (to a maximum value of 10) are at the far right of table 6.

<table>
<thead>
<tr>
<th>Method*</th>
<th>M</th>
<th>( M ) normalized</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>3.2</td>
</tr>
<tr>
<td>2</td>
<td>49</td>
<td>10.0</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>2.1</td>
</tr>
</tbody>
</table>
The bar graph in figure 3 portrays the weighed, maturity values from table 6. Reference lines show in figure 3 show estimated, minimum, desired scores of developers and users and the maximum

|   | 4   | 5   | 6   | 7   | 8   | 9   | 10  | 11  | 12  | 13  | 14  | 15  | 16  | 17  | 18  | 19  | 20  | 21  | 22  | 23  | 24  | 25  | 26  | 27  | 28  | 29  |
|---|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 4 | 5.4 | 6.3 | 2.7 | 7.2 | 4.5 | 8.1 | 4.5 | 3.6 | 3.6 | 2.7 | 4.5 | 11  | 2.2 |
| 5 | 0.9 | 1.8 | 4.5 | 4.5 | 4.5 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 5   | 1.1 |
| 6 | 9   | 4.5 | 4.5 | 6.3 | 6.3 | 4.5 | 0.9 | 1.8 | 1.8 | 1.8 | 1.8 | 8   | 1.7 |
| 7 | 9   | 0.9 | 2.7 | 7.2 | 8.1 | 2.7 | 0.9 | 1.8 | 1.8 | 4.5 | 2.7 | 13  | 2.6 |
| 8 | 8.1 | 0.9 | 6.3 | 7.2 | 8.1 | 8.1 | 0.9 | 2.7 | 2.7 | 2.7 | 2.7 | 13  | 2.6 |
| 9 | 6.3 | 2.7 | 4.5 | 5.4 | 6.3 | 7.2 | 0.9 | 4.5 | 1.8 | 5.4 | 3.6 | 11  | 2.3 |
| 10| 9   | 0.9 | 4.5 | 7.2 | 7.2 | 7.2 | 0.9 | 3.6 | 2.7 | 2.7 | 1.8 | 12  | 2.4 |
| 11| 8.1 | 0.9 | 3.6 | 6.3 | 8.1 | 6.3 | 2.7 | 2.7 | 2.7 | 3.6 | 19  | 4.0 |
| 12| 8.1 | 0.9 | 6.3 | 5.4 | 4.5 | 7.2 | 0.9 | 1.8 | 3.6 | 5.4 | 2.7 | 10  | 2.1 |
| 13| 8.1 | 1.8 | 5.4 | 4.5 | 8.1 | 4.5 | 0.9 | 2.7 | 4.5 | 3.6 | 2.7 | 13  | 2.6 |
| 14| 7.2 | 2.7 | 6.3 | 5.4 | 7.2 | 7.2 | 0.9 | 4.5 | 4.5 | 5.4 | 5.4 | 14  | 2.9 |
| 15| 5.4 | 2.7 | 6.3 | 7.2 | 7.2 | 6.3 | 0.9 | 4.5 | 3.6 | 3.6 | 3.6 | 12  | 2.4 |
| 16| 8.1 | 1.8 | 7.2 | 4.5 | 8.1 | 0.9 | 0.9 | 1.8 | 1.8 | 1.8 | 1.8 | 10  | 2.1 |
| 17| 5.4 | 1.8 | 7.2 | 8.1 | 8.1 | 5.4 | 0.9 | 0.9 | 1.8 | 0.9 | 1.8 | 10  | 2.1 |
| 18| 9   | 0.9 | 6.3 | 4.5 | 6.3 | 7.2 | 0.9 | 2.7 | 3.6 | 3.6 | 6.3 | 16  | 3.3 |
| 19| 7.2 | 1.8 | 2.7 | 2.7 | 3.6 | 7.2 | 0.9 | 2.7 | 3.6 | 2.7 | 3.6 | 13  | 2.6 |
| 20| 2.7 | 1.8 | 8.1 | 4.5 | 8.1 | 2.7 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 9   | 1.9 |
| 21| 0.9 | 1.8 | 8.1 | 4.5 | 7.2 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 0.9 | 8   | 1.7 |
| 22| 9   | 0.9 | 2.7 | 5.4 | 5.4 | 7.2 | 0.9 | 7.2 | 5.4 | 3.6 | 3.6 | 24  | 4.9 |
| 23| 9   | 0.9 | 2.7 | 6.3 | 3.6 | 7.2 | 0.9 | 4.5 | 2.7 | 7.2 | 3.6 | 17  | 3.4 |
| 24| 8.1 | 0.9 | 2.7 | 4.5 | 8.1 | 7.2 | 0.9 | 3.6 | 2.7 | 5.4 | 3.6 | 21  | 4.3 |
| 25| 8.1 | 0.9 | 7.2 | 8.1 | 8.1 | 2.7 | 0.9 | 1.8 | 1.8 | 1.8 | 1.8 | 10  | 2.1 |
| 26| 7.2 | 2.7 | 4.5 | 6.3 | 5.4 | 4.5 | 0.9 | 2.7 | 1.8 | 2.7 | 1.8 | 8   | 1.6 |
| 27| 9   | 0.9 | 4.5 | 6.3 | 8.1 | 7.2 | 0.9 | 2.7 | 1.8 | 2.7 | 1.8 | 13  | 2.6 |
| 28| 5.4 | 0.9 | 2.7 | 6.3 | 5.4 | 7.2 | 0.9 | 2.7 | 2.7 | 2.7 | 3.6 | 13  | 2.6 |
| 29| 9   | 0.9 | 3.6 | 8.1 | 8.1 | 7.2 | 0.9 | 2.7 | 7.2 | 6.3 | 1.8 | 17  | 3.4 |

The bar graph in figure 3 portrays the weighed, maturity values from table 6. Reference lines show in figure 3 show estimated, minimum, desired scores of developers and users and the maximum
score. The bar graph in figure 4 shows the normalized, weighted maturity values from table 6 (far right column).

![Maturity (Weighted Average)](image1)

Figure 3.3.5-3

![Maturity (Normalized Weighted)](image2)

Figure 3.3.5-4

3.6 Conclusions

The conclusions to this survey are divided into two portions. Section VI-A discusses major functional components selected from methodologies summarized in this survey and categorized in table 1. Section VI-B provides maturity conclusions based on the corresponding tables and graphs in this survey. It also includes accuracy, coverage, performance and scalability conclusions.
3.6.1 Major functional components

The organization of major functional components in table 1 shows what NLP and NLU capabilities can be produced using certain functional components existing in different collections of methods from this survey. The conclusions for methods are organized into the following groups: (1) clustering, (2) similarity measure, (3) influence of noun, verbs, and their modifiers, (4) concept and semantic representations, and (5) other methods not belonging to a particular group are listed below.

3.6.2 Clustering:

Clustering can filter out common (relatively unimportant) nouns from a graph, leaving more pronounce nouns to determine the sense of words. The result is a more efficient, compressed graph representation of text. It can also calculate similarity measures for detecting similar documents and reduce redundancy at the concept level.

3.6.3 Similarity measure:

Distances in graphs (such as number of nodes between start and finish) can represent important components of NLP including similarity and senses (i.e., meaning) of words. This is normally accomplished while traversing nodes using an algorithm such as random walk. Weighed, directed graphs can be used to find the range of influence of terms and how they are semantically related.

3.6.4 Nouns and verbs plus their phrases and modifiers:

a.) Within a corpus used for training, nouns were found to have a higher discriminative ability than verbs, adverbs, and adjectives.

b.) Subjects, verbs, objects, and verb modifiers used in graphical relations enable entailment ranking and graph summarization algorithms to combine similar data points into single vertices to represent clusters of data points, which compress the graph representation.

c.) An In-Degree algorithm uses nouns, verbs, adjectives, and adverbs to find the most probable sense of words based on context by creating a weighed graph with nodes representing senses of a word and edges with weights connecting the word to each sense.
d.) An event indexing method produces chains of connectivity between subjects, verbs, and objects including their noun and verb synonyms. This approach appears to work best for text containing lots of events and actions.

3.6.5 Concept Representations are created in the following ways:

a.) Use domain knowledge with ontology to map text to concepts. This provides additional information for summarizing documents. Intuitively, this added information would add inference at various nodes in a concept graph to increase the resulting summarization accuracies.

b.) Creating a concept lattice to index local topics within a topic hierarchy. This optimizes coverage of concepts within its graph representation which optimizes summaries.

c.) Use a concept clustering technique called formal concept analysis to produce a concept lattice with a concept hierarchy. Vectors used in this process contain verb-attribute pairs. A distance-measure between the vectors determines concept similarity.

d.) Build semantic space models from syntactic dependencies to produce a graphical representation of knowledge.

e.) Generate a concept frame graph to guide the user to mine new information from text.

3.6.6 Other methods not fitting in a group:

Provide graph based ranking of data in labels, find closest sense of words in labels, detect new information, generate grammars for a given text, resolve references to events, and generate graphs from word co-occurrence in text.

3.7 Maturity

The following maturity discussion tells how some maturity values were derived from some respective papers. Methods with relatively high maturity values such as method 6 resulted because of their use of well known and used algorithms such as Chinese Whisper clustering algorithm which is used by major search engines. However, originality, in this case was low because of use of algorithms already developed earlier by others. The factors that were multiplicative rather than additive in the maturity

82
calculation had a larger impact on the final values. Since originality was additive, its impact on maturity was significantly less than the multiplicative factors such as robustness. Collective results of the 26 methods averaged to a relatively low value. Thus, more work needs to be done on maturing NLP and NLU research so that results can better meet developer and user needs. Results of accuracy, coverage, performance, and scalability showed that the average as well as the maximum scores still left significant room for improvement to meet desires of both developers and users.

3.7.1 This survey has provided a summarization of graphical NLP and NLU methods (labeled as 1 through 26) with an analysis of estimated accuracy, coverage, performance, scalability and estimated maturity based on the information available in the referenced papers.

The tables of values show that this area of research, on the average, needs further improvement before they meet the desires of developers and users of products from this research area. Although significant improvements have been made over the years, a majority of these methods are still in progress. Some of the publications reviewed for this survey showed that systems have been developed from this research area and are used mostly in the medical field, but more mature systems are needed. Very little on maturity has been reported in these research areas. Maturity values calculated in this survey were estimated based on information provided in the papers. As this research area continues to progress, more efforts and published results on maturity should help system developers and users meet their needs in various NLP and NLU applications.
IV. NATURAL LANGUAGE TEXT TO GRAPH REPRESENTATION

We developed a program, written in Java, to transform the agents, actions, and patients from subject nouns, verbs, and object nouns in each phrase and sentence of a text document into graphs. We used a natural language (NL) parser [80, 81, 82], developed by Stanford University, that converts the NL text into a parse tree, which our program reads as a string of parentheses, symbols, and words. Figure 4.1 shows the parsed output of a simple sentence: “I saw Tom with the telescope.” The symbols consist of parts of speech tags, which precede each word, and non-terminal nodes within the parse tree, which identify the beginning of phrases, sentences, noun phrases, and verb phrases. The non-terminal nodes, such as “S”, “NP”, “VP”, are symbols standing for beginning of a phrase or sentence, a noun phrase, and a verb phrase respectively. We use these symbols to guide the flow of our program in extracting and storing words as agents (i.e. ”I”), actions (i.e., “saw”), and patients (i.e., “Tom” and “telescope”) into arrays and converting these words and their relations into vertices and edges of a graph.

![Figure 4.1: Parsed input from NL parser, used as input to program.]

4.1 Implementation details

The data structures, consisting of various arrays, are indexed to efficiently store and access multiple agents, actions, and patients from phrases and sentences and provide information for generating the graph structures. The program detects whether a sentence is active or passive voice and changes the direction between agents and patients for passive sentences to maintain consistent flow of action represented in the resulting graphs. As the program extracts each word, detects its function (agent, action,
patient, or other), stores each word with associated markings into arrays based on their position (or level) in the parse tree, and accounts for multiple agents, actions, and/or patients within a sentence or phrase, it constructs a graph of agents, actions, and patients. To construct a graph, the program invokes methods from a graph visualization library called the Java Universal Network/Graph Framework (JUNG-2.01) [83, 84, 85, 86], developed by the University of California at Irvine. JUNG Methods “add.vertex()” and “add.edge()” convert submitted words into vertices, convert submitted relations between these words into edges, and produce visualizations of the resulting graphs in both java swing views and image files. The resulting image files are presented as graphs within this document. Figure 4.2 shows a simple example of the program’s basic flow from the parsed input to generated, internal, data structures and graph output.

The parentheses “(“ and “)”) and non-terminal nodes S, NP, VP, obtained from the parse tree, synchronize the flow of the program while the input stream of symbols and words are read and inserted in arrays and vertices to construct the graphs. A counter increments for each open parentheses and decrements for each closed parentheses which are read from the input stream to indicate the height (or level) of each symbol in the parse tree. Comparing the relative values of the levels of two or three symbols (i.e., “S”, “NP”, and
“VP”) can help determine a noun’s relationship to a near-by verb. Levels and order of word help determine whether the noun is an agent or a patient. The symbol “S” indicates the start of a phrase or sentence. Each symbol with its level is pushed into a stack and popped from the stack to maintain the proper order of method invocations which insert words into arrays and graph vertices. At the lowest (or terminal) ends of the parse tree, each word from text is preceded with a parts of speech tag. Nouns from the parse tree are tagged: NN, NNS, NNP, NNPS, and PRP. Verbs are tagged: VB, VBG, VBD, VBZ, VP, and VBN. The first letter of the tag (such as N or V) indicates a noun or verb, and the rest (such as VBG) indicate the type of noun or verb. The program uses the tags to detect nouns and verbs and extracts them from the input string. It uses verb tag VBG to recognize verbs that are gerund verbs to generate phrases distinct from the rest of the sentence input. When detecting a second “S” symbol, it increments indexes and counters for a phrase. It uses verb tag VBN to detect passive verbs. Then it reverses agent and patient nouns associated with that verb to maintain the agent-action-patient flow of the arrays and graph.

Queues hold nouns, to be used as agents (or passive patients), until verbs arrive from the input stream to be used as actions of the corresponding agent nouns. If the program tries to use the noun before the verb has arrived from the parsed input stream, it would not have enough information to tell whether the noun was an agent or patient. The program handles all the word extraction from input, stores words and associated information in arrays, maintains counters and indexes, and generates graph nodes and edges in one pass to maximize performance whenever practical and feasible. The agent nouns are then polled from the queues in the sequence they arrive from the input stream. The queues save and delay the nouns until verb tags arrive from the input stream to indicate what actions to perform on the nouns depending on whether a verb is active or passive voice, which determines where the nouns are inserted in arrays and graphs, and detecting the beginning of a phrase from an “S” symbol, which increments indices for storing a new phrase from within the current sentence.

Figure 4.2 also shows the stacks and queues saving nouns until verbs arrive from the input with their corresponding parts of speech symbols (or tags). Depending on the verb’s part of speech and respective order of arrival from the input stream, the nouns are inserted as agents or patients with their corresponding actions obtained from the verbs. To maintain phrase and sentence integrity, the first agent,
action and patient of a phrase or sentence (if they exist) is inserted into arrays using the same index. When multiple agents, actions, and/or patients are present, all except the first are stored in a separate array. Separate counters, Boolean indicators, and indexes are maintained to enable quick access to these arrays. The reason for creating and maintaining separate arrays for multiple word types was for keeping phrase and sentence identity associated with each word and for efficient access, assuming a majority of phrases and sentences are more likely to contain single agents, actions, and patients. Each method for word (agent, action, or patient) insertion into arrays and graphs is designed to handle multiple agents, with each agent causing one or more actions, and each action impacting one or more patients, with sentences or phrases within sentences containing this flexibility in structure.

The program invokes methods for inserting these words into vertices of the graph being constructed shortly after they are inserted into arrays. Directed edges in the graph show relations from agents to each of their actions and from actions to each of their patients. The program also stores graphical information such as color and shape of nodes into these arrays to be used later in graph visualization routines. Changes in node shape are used for visualizing Stochastic Petri Nets addressed in the next chapter. Action nodes in an SPN graph have a different shape (a thin rectangle) than agents and patients.

Phrases containing agents, actions, and patients produce the same array and graph structures as sentences. Figure 4.3 shows array and graph structures containing the agents, actions, and patients of a sentence containing phrases. The same index is used to reference either phrases or sentences, since the same array is used for both. Phrases can function as agent phrases or patient phrases in relation to the action verbs, external to the phrase but within a sentence. Thus, an agent phrase references one or more actions (verbs), within a sentence, and an action can reference one or more phrases serving as patients of the sentence. As a result, sentences containing phrases produce graphs representing both phrases and sentences with edges relating agent phrases to action verbs and action verbs to patient phrases. Each phrase or sentence can contain one or more agent, action, and patient or only two of the three (an agent performing an action, or an action affecting a patient). Each phrase (containing agent, action, and patient) is stored in an array and inserted into a graph, with edges between each phrase and its respective action within the sentence containing the phrases. Sentences containing such phrases are often referred grammatically as
compound sentences. Thus, the program is designed to handle multiple agents, actions, and patients and multiple agent phrases and patient phrases.

Figure 4.3 Array and graph structures containing the agents, actions, and patients of a sentence containing one agent phrase and two patient phrases

Figure 4.4 Graph output (to the right) is from example in figure 4.3. The three phrases are shown consisting of an agent, action, and patient. Vertices “caused” and “made” are actions caused by one phrase (my seeing telescope) impacting two other phrases (Tom hide) and (Nancy run home).

In addition to the above, the program also handles active and passive sentences by analyzing the tag symbol of the verbs within a phrase or sentence. Figure 4.5 shows the graph structures containing a passive sentence in one of its graph segments and is an example of a long sentence. The example text (shown in figure 4.5) came from the Stanford parser website [80]. When our program detects a passive verb, it switches the stored agents and patients of the associated passive action to maintain efficient
queuing of the arrays and to show the path of actions in the graph. It reverses the direction of edges in a graph which are adjacent to a passive action vertex.

Text input corresponding to graphs shown to the right:
“The strongest rain ever recorded in India shut down the financial hub of Mumbai, snapped communication lines, closed airports and forced thousands of people to sleep in their offices or walk home during the night, officials said today.”
(Reference: Example from Stanford Parser documentation [80, 81, 82].)

Figure 4.5 Example of a long sentence with several actions impacting several patients and one passive phrase.

Figure 4.5 shows the graph output of both a long sentence containing several actions impacting patients and one passive phrase. The text for the phrase contains “rain recorded in India.” Since “recorded” is a passive verb, the adjacent arrows are switched to “India recorded rain.” “India” becomes the agent of other actions.

Another example of a long sentence, but with multiple phrases, is shown in figure 4.6. The agent and action “My seeing” witnessing a phrase “Jack kick the ball” produced the action “caused” on the phrases “me to run and tackle Tom” and action “keep” on two patient phrases “him from scoring” and “winning the game.” Note that the graphs only contain the agents, actions, and patients of the sentence, which is an objective of the program.

Input Text:
My seeing Jack kick the ball, caused me to run and tackle Tom, while he was running, and keep him from scoring more points and winning the game.

Figure 4.6. A complex sentence containing three phrases.
Besides generating arrays and graphs representing phrases and sentences within a text, the program can detect chains of causes and effects throughout a significantly large text by globally capturing the effect of actions from agents in one sentence to agents and patients of other sentences throughout the text. One potential purpose of such a graph is to provide a more global automated analysis of causes and effects over an entire document. The JUNG2 library automatically combines vertices with the same vertex identification (ID) number (noun or verb) resulting in a compressed graph. However, some restraints by the program are necessary to avoid introducing ambiguous interpretations. If a directed sparse graph contains multiple vertices with the same ID number (i.e., has the same noun or verb), the JUNG2 library will automatically remove duplicate vertices including their adjacent edges. The program constrains this behavior by only allowing the same agent and patient nodes to combine but keeps the same action nodes from combining to keep sentence ambiguity from growing. Vertices with the same label remain separate unless they also have the same vertex ID. Figure 4.7 is an example of a combined graph (i.e., vertices with the same ID). The program finds vertices with the same ID by searching, within a “for loop”, for matching ID numbers. To improve efficiency, two ID numbers are changed from Java strings to hash code integers before they are checked for matching ID numbers.

Fig 4.7 a. Uncompressed: 10 agents (blue) and 14 patients (yellow) are shown.

Fig 4.7 b. Compressed: 4 agents and 8 patients are shown.
If the program allowed multiple agents producing multiple actions on multiple patients can potentially become ambiguous in certain portions of the combined graph. This can happen when a group of agents cause the same action or group of actions to effect a group of patients in a sentence. Therefore, we developed the following rules (or steps) when combining the agents, actions, and patients from multiple sentences throughout a text.

Table 4.1 lists rules for globally combining the agents, actions, and/or patients from multiple phrases and sentences throughout a graph representing a text containing several sentences. The same actions with the same (and no other) agents can be treated as one object having one or more patients.

Table 4.1: Rules for combining duplicate vertices within a graph when creating chains of events

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Patients</td>
<td>Combine all of the patients having the same labeled name.</td>
</tr>
<tr>
<td>2. Actions</td>
<td>Avoid combining actions with the same name to keep from adding ambiguity to the sentence. However, if more than one action has the same name and each of them have the same agents or the same patients, then the actions could be combined.</td>
</tr>
<tr>
<td>3. Agents</td>
<td>Combine all of the agents having the same labeled name.</td>
</tr>
<tr>
<td>4. Phrases</td>
<td>Since the agents, actions, and patients within a phrase are handled the same as in sentences, the same rules 1, 2, and 3 apply.</td>
</tr>
<tr>
<td>5. Combining Agent and Patient Phrases</td>
<td>Assuming the text is reasonably written to follow accepted grammar rules, phrase duplication is expected to be rear. Therefore, combining duplicate agent phrases and patient phrases is unlikely to add any benefit to the research.</td>
</tr>
</tbody>
</table>
Figure 4.8 shows a diagram of what the combined graph in figure 7 could look like when the rules in table 4.7 are followed. The diagram shows possible enhancements such as compression 1 (enclosed by the blue dotted circle) and changing “was seen” to “saw” in compression 2 after swapping the agent and patient nouns of a passive sentence to maintain consistent flow of agent to action to patients in the graph.

Compression 2, enclosed by the red dotted ellipse, would be an exception that violates a rule and yet not cause increased ambiguity. It could reduce redundancy by combining the nouns “I” and “we” and the verbs “saw” and “looked”, assuming it had help from a linked dictionary, such as WordNet, to identify similar words. Also, it would have to recognize that “I” is a subset of “we” and that “we saw table” would contain “I saw table” even though “I saw table” was not included in the original text or in the graph diagram. Such enhancements could be added in future program enhancements.
4.2 Experimental Results of Graph Program

The following graph diagrams are outputs from the graph program which uses parsed input from the Stanford parser, processes and stores words as described in this chapter, and feeds output to the JUNG2 graph visualization library which creates vertices and edges from the methodologies within the graph program.

1. Two sentences: A medical example of events leading to an aneurysm. Text Input:

A worn pump in a pace maker caused Tom’s heart to beat faster which increased blood pressure in his arteries. Too much blood pressure can affect the brain and cause an aneurysm resulting in possible death.

![Uncompressed (9 patients)](image1)

![Compressed (7 patients)](image2)

2. Two sentences: A resistor example of events.
a. Text Input: A resistor in a radio burned out and short circuited a transformer which caused the radio’s speaker to make a very loud noise. The loud noise caused the driver of the car, which contained the radio, to drive off the road and run into a barn containing several chickens that became very alarmed and laid several eggs.

![Uncompressed (13 patients)](image3)

![Compressed (9 patients)](image4)
3. Three phases in sentence - Text Input:
My seeing the telescope caused Tom to hide and made Nancy run home.

4. Complex sentence with phrases – Text input:
My seeing Jack kick the ball, caused me to run and tackle Tom, while he was running, and keep him from scoring more points and winning the game.

Figure 4.11 Since each agent and patient node has unique names, no compression is needed. From the graph, “My” in phrase “My seeing telescope” caused “Tom to hide” and made “Nancy run home.

Figure 4.12 Since each agent and patient node has unique names, no compression was needed. Figure 4.12 (a) is uncompressed, (b) is with compression on with no changes due to unique names.
5. Ten sentences – Text Input:

I saw Tom with the telescope. The telescope sat on the table. The table and telescope was seen by all of us. We had a great time at the party. It was Tom's birthday. Tom received several presents from all of us. The telescope was a Tom's favorite present. The telescope appeared expensive. I looked through the telescope. A great time was had by all.

Fig. 4.13 a. Uncompressed: 10 agents (blue) and 13 patients (yellow) are shown.

Fig. 4.13 b. Compressed: 4 agents and 7 patients are shown. As in all the graphs of this section, none of the actions are compressed to avoid adding ambiguity.

6. Long sentence – Text Input: From example in Stanford parser documentation [80,81,82]

The strongest rain ever recorded in India shut down the financial hub of Mumbai, snapped communication lines, closed airports and forced thousands of people to sleep in their offices or walk home during the night, officials said today.

Figure 4.14 is a graph of a long sentence (found as an example in the Stanford parser documentation. Since all the words are unique, compression would not improve the graph. Intuitively, from the text, “rain” should be the agent. However, “rain recorded in India” is passive. Therefore, the program reverses the roles of “rain” and “India” and makes “India” the main agent of the long sentence.

The graph shows India “closed financial hub, closed airports, recorded rain, forced people ….” The sentence shows “rain” as the agent, but since the initial phrase is passive, rain and India are switched with India as the agent in the graph.
7. Long text – Example description of an image - (Font is decreased to compress area of text.): [90]

Image-1, Extracted NL Outcome:
Number of objects: 4 objects recognized in the input image.
Type of objects: A yellow car, a brown airplane, a black helicopter and a silver wrench-tool
Location of the objects: The airplane is on the upper left part of the image. Its position is “nose-SE-side”. The helicopter is on the upper right part of the image. Its position is “nose-NE-side”. The wrench-tool is on the upper center part of the image. Its position is “open-edge-NE-side”. The car is on the lower center part of the image. Its position is “nose-W-side”.
Associations of the objects:
The car is 2.1 the length of the airplane, 2.3 the length of the helicopter, 1.3 the length of the wrench-tool;
The car is the biggest in length object;
The airplane is the second in length object;
The helicopter and the wrench-tool have similar length;

Image-2, Extracted NL Outcome:
Number of objects: 4 objects recognized in the input image.
Type of objects: A yellow car, a brown airplane, a black handle hummer and a silver wrench-tool
Location of the objects: The airplane is on the lower central part of the image. Its position is “nose-NE-side”. The hummer is on the middle-diagonal of the image. Its position is “back-side-top-NE” The wrench-tool is on the right-down part of the image. Its position is “open-edge-top-NW-side”. The car is on the upper left part of the image. Its position is “right-side-nose-NE”.
Associations of the objects:
The hummer is the longest object with 2.1 times the length of the wrench-tool, with 3 times the length of the airplane, with 1.3 the length of the car;
The car is 1.2 the length of the wrench-tool, 1.8 of the airplane;
The car is second in length object;
The airplane is the smallest object;

4.15 a. Uncompressed

4.15b. Compressed: Noticeably fewer agents and patients are shown.

Figure 4.15 An example of a description of an image. [90]
8. Very long text (approximately two pages when font is larger font):

1. Introduction

Over the past several years, information, especially from the internet, has become so vast that professionals, from a number of disciplines, have difficulty keeping up to date within their respective fields. For example, medical doctors devote tremendous amounts of time to capture the latest developments from research areas within their field of specialty. A large amount of this time is wasted reading redundant information from various documents. Needed information may also be lost in the process of summarization. Advanced methods of search, database technologies, data mining, and other areas have helped, but not enough to meet the growing need from these professionals.

For the past 40 years, researchers have tried to address this problem by automatically or semi-automatically capturing information from single and multiple documents into less redundant text, typically in the form of summaries. If a sufficient solution will be found, the resulting increased capability would become a significant breakthrough and help researchers and professionals capture more information to advance their areas of specialty and collectively advance a multitude of technologies. In addition, several methodologies have been developed to advance the area of natural language processing in order to find solutions to this problem. However, no known methodology appears to capture the needed information and generate text with enough quality and speed to satisfy this need. Thus, this survey summarizes and compares current methodologies, which deal with the removal of redundancy for documents retrieved from different resources.

This paper surveys research methodologies related to the area of natural language processing (NLP) and understanding (NLU). The purpose is to document the progress in natural language understanding research and how it can be applied to capturing concepts from multi-documents and producing non-redundant text while attempting to maximize coverage of the significant information needed by the user. The required information could span from a single or multiple domain coverage. Thus, the paper explores the current state of NLU technology and seeks its robustness, time and space complexity, scalability for handling large numbers (n > 100) of technical documents, and technology gaps needing further research. The information in this survey summarizes papers from various technical journals and conference proceedings.
The methodologies under evaluation in this paper cover the following areas: (1) detection of important sentences, (2) concept extraction from text, (3) building concept graphs, (4) attribute and relation structures leading toward knowledge discovery from text, (5) increasing efficiency in the processes leading to concept representations, (6) generation of non-redundant text summaries, and (7) maximizing the readability (or coherence) of automatically generated or extracted text. Technologies used in these methodologies include machine learning, statistical and discreet approaches, cluster indexing, lexical chaining, concept lattices, models that follow cognitive psychology concepts, stochastic Petri-nets (SPNs) and fractal theory. Technologies taken from these areas of research provide a broad spectrum of methodologies or approaches that can provide a baseline for further NLU research. Moreover, methodologies that create summaries of single and multiple documents can be grouped into two major approaches: One approach includes capturing important concepts from text, using a collection of sample texts (called corpus) to train machine learning algorithms, minimizing the number of concepts without losing too much information, generating summarized text, and making the resulting text easily readable. Another approach is extracting the more important sentences (or phrases) that can be directly converted to non-redundant summaries and bypassing the concept capture and generation steps.

2. Methods and Features

In this section we present a variety of methodologies classified according to their features. In particular this section covers the various groups: text relationship map with latent semantic analysis, extraction methods for text summarization, cluster summarization, formulated semantic relations, SPN representation for document understanding, concepts representation for text, learning ontologies from text, synthesis of documents, generation of semantically meaningful text using logic order, text generation methods, document structural understanding, and other relevant methods. The methods presented here will be compared and evaluated based on their maturity. The overall results are presented in section 3.

2.1 Text Relationship Map with Latent Semantic Analysis (LSA)

Yeh et al (2008) present the two methodologies, text relationship map and latent semantic analysis, that they use together for text summarization. Yeh et al (2008a) uses feature weights to create similarity links between sentences forming a text relationship map [41].

Advantages: This methodology captures various features that help in calculating the similarity of sentences throughout one or more documents. The paper gives significant detail about the methodology.

Disadvantages: This methodology is based at the word level.

Yeh et al. (2008b) LSA-based text relationship map (T.R.M.) approach derives semantically salient structures from a document. Latent semantic analysis (LSA) is used for extracting and inferring relations of words with their expected context [41].

Advantages: The paper gives significant detail about the methodology. Several features are used in the similarity calculation.

Disadvantages: This methodology is based at the word level. The LSA approach uses a Word-Sentence matrix that can get very large due to the number of words in a document or in multi-documents.

2.2 Extraction Methods for Text Summarization

Ko and Seo (2008) present a hybrid sentence extraction method that uses some context information augmented with mainline statistical approaches to find important sentences in documents. Their model combines two consecutive sentences into a bi-gram pseudo sentence representation to overcome feature sparseness [19].
Uncompressed

Compressed (noticeably fewer agents and patients are shown.)

4.16 a. Uncompressed  

4.16 b. Compressed

Figure 4.16 Graphs (uncompressed and compressed) are shown for a very long text (one to two pages).
V. GRAPH TO STOCHASTIC PETRI NET CONVERSION

5.1 SPN Visualization

We converted our implementation of graphs, discussed in the previous chapter, into Stochastic Petri Nets (SPNs). The visualization of each graph node conversion consists of (1.) changing the shape of each agent and patient node from rectangles to ellipses, to represent SPN places, and (2.) changing the shape of action nodes from long rectangles to thin (small width) rectangles to represent transitions. Labeled names are left inside the agent and patient ellipses and overlap the center of the action (thin) rectangles. Colors for agent and patient places and for action transitions remain the same as for their corresponding graph node counterparts for consistency. Any other nodes that appear on the graph (such as nodes within prepositional phrases) will appear as places with no color (or white). A transition is inserted between any two places in the SPN graph. Colors (such as red and green) will also be selected for markings in inhibitor places used to select various interpretations of sentence meanings, discussed later in this chapter. Bourbakis and Mills [90] illustrate the conversion of agents and patients in graph to SPN places and the conversion of actions in a graph to SPN transitions. Figure 5.1 shows graph node and edge conversions to SPN places and transitions.
Figure 5.1 shows the visualization conversion from graph to SPN. Top portion illustrates their appearance. Bottom two graphs show generated graph and SPN outputs: A graph of agent, action, and two patients, and a graph and SPN graph of the entire simple text: “I saw Tom with the telescope.”

The shape and color of the nodes are implemented using the Java graphics library and a transformer interface feature provided by the apache commons collections portion of the JUNG2 library [87]. Transformers can iteratively change shape, color, labeling, and other characteristics of each node and edge of a graph and can interface Java graphics with JUNG2 graphs which are implemented with different (i.e., incompatible) type objects. Figure 5.2 is a sample of our Java code using a shape transformer for SPN.

```java
// Java Code using a transformer to visualize the shape of each SPN component in an SPN model:
import edu.uci.ics.jung.graph.DirectedSparseGraph; // This type of graph is used by graph and SPN.
import edu.uci.ics.jung.visualization.VisualizationViewer; // JUNG visualization viewer
import edu.uci.ics.jung.visualization.renderers.Renderer; // Used for rendering vertex of graph/SPN
import java.awt.geom.Point2D; // For Java graphics
import java.awt.Color;
import java.awt.Shape; // This is the Java shape portion of Java graphics.
import java.awt.Paint;
import java.awt.geom.Ellipse2D; // Used for SPN agent and patient components.
import java.awt.geom.Rectangle2D; // Used for SPN action components
import org.apache.commons.collections15.Transformer; // This provides the transformer capability.
...
Transformer<Integer, Shape> vertexShape = new Transformer<Integer, Shape>(){
    public Shape transform(Integer i){ // This transform method provides the needed data.
        Point2D center; // if statement below creates thin rectangle for SPN transition of Action
        if((sen.gvNode[i] == sen.gAction)||(sen.gvNode[i]==sen.gMoreActions)){
            Rectangle2D shape = new Rectangle2D.Double(-10,-20,10,40); // location and size
            // The above negative values are essential for automatic placement of components.
            shape.setFrame(new Rectangle2D.Double(-10,-20,10,40));
            sen.gvShape[i] = shape;  // gvShape is an array storing colors of components.
            sen.gvShape[i] = shape; // returns shape of the i th component to the transformer
        } // end Action
        else {// Create Ellipse for SPN Place to represent Agent or Patient.
            Ellipse2D shape = new Ellipse2D.Double(-20,-10,60,20);
            sen.gvShape[i] = shape;
            return sen.gvShape[i]; // returns shape of the i th component to the transformer
        } // end else (Agent or Patient)
    } // Shape transform
}; // end Transformer vertexShape
if(sen.SPN) // Execute Vertex Shape Transformer with shape from array gvShape(vertex).
    vv.getRenderContext().setVertexShapeTransformer(vertexShape);
```

Figure 5.2 shows a portion of sample code to show how a transformer is used to manipulate the shape of an SPN component depending on whether it is an agent or patient place or an action transition. Also shown are imports of Java and JUNG2 library portions that are necessary for the transformer to work.
The above code and the following discussion on JUNG2 transformers should help readers of this dissertation who need to use such transformers in their graph and SPN diagram visualizations to implement such transformers. The above sample code in figure 5.2 shows a transform method that provides data from our program that is fed to the transformer.

The statement: “vv.getRenderContext().setVertexShapeTransformer(vertexShape);” invokes the Transformer named “vertexShape” to get the desired shape that we specify in the encapsulated transform method. From “Transformer<Integer, Shape>”, the integer input is each graph node (or component in the SPN graph) that is iteratively provided by the RenderContext() method of the visualization viewer “vv” instantiated from the JUNG2 library. The Shape output is what the transformer provides back to the visualization viewer which we specify in the transform method from what is stored in our gvShape array. Index “i” references each node of the graph (or SPN component) and is automatically and iteratively provided by the RenderContext() which stores all provided characteristics of each graph vertex and edge (that we convert to SPN places and transitions connected by edges). Similar transformers with the same or similar code structure are used for specifying colors and labels of each graph vertex or SPN component.

5.2 SPN Synthesis

5.2.1 SPN graph structures for representing single and combined interpretations

Bourbakis and Mills [90] propose two steps in synthesis of SPN Graphs: (1.) Combine all interpretations of each sentence to create a complex SPN model that holds all meaningful interpretations. (2.) Reduce interpretations by using additional information from context to remove ambiguity from the NL sentence. They use color markings in places to enable (or disable) alternative interpretations. Synthesis rules such as: (1) one-to-many (one input to a place causing more than one transition which results in many outputs from the transitions at the end points) and (2) many-to-one (many inputs to one place causing transitions effecting one place with one output) is one approach to adhere to place and transition placement in SPN graphs [90]. The one-to-many structure could model expanding interpretations of a sentence and many-to-one could model reducing interpretations of a sentence. Inhibiting places could provide a means for reducing interpretations as contextual (or event association) information is discovered through synthesis.
SPN synthesis is illustrated in Figure 5.3 which is from Bourbakis and Mills [90]. Two graphs (a. and b.) show two interpretations of the sentence “I saw Tom with the telescope.” G1 and G2 are equations of the two graphs. SPNs of these graphs are shown c. and d. Combined interpretations are in e.

\[ G1 = N_1 \alpha_{12}^r N_2 \alpha_{23}^r N_3 \alpha_{34}^r \]
\[ N_4 \alpha_{45}^r N_5 \alpha_{56}^r N_6 \]
\[ N_2 \alpha_{26}^r N_6 \]

Figure 5.3 a. Graph G1 interpreted as “I saw Tom using the telescope.” [90]

\[ G2 = N_2 \alpha_{14}^r N_4 \alpha_{45}^r N_5 \alpha_{56}^r N_6 \alpha_{62}^r N_2 \alpha_{23}^r N_3 \]

Figure 5.3 b. Graph G2 interpreted as “I with the telescope saw Tom.” [90]

Figure 5.3 c. SPN of graph G1 interpreted as “I saw Tom using the telescope.” [90]

Figure 5.3 d. SPN of Graph G2 interpreted as “I with the telescope saw Tom.” [90]
The SPN synthesis processes of converting two or more SPN graphs, each representing a different interpretation, into one complex SPN graph containing all of the interpretations, each selectable by a different color token, uses two rules: (1) one-to-many and (2) many-to-one. These two rules are illustrated in figure 5.3 f.

![Figure 5.3 e. SPN containing both interpretations. Places with green and red markings enable SPNs of Fig 5.3 c and d. [90]](image)

Figure 5.3 e. SPN containing both interpretations. Places with green and red markings enable SPNs of Fig 5.3 c and d. [90]

5.2.2 Methods for gathering information needed for SPN synthesis:

5.2.1.1 Agent-action-patient chains provide a synthesis backbone for gathering event association information that would help drive proper interpretation of a sentence resulting in sentence ambiguity
reduction. When combining duplicate agents and patients of an agent-action-patient chain, edges that are combined increment the weight of the resulting edge each time it reduces two edges into one. (Weights of edges in a graph can be converted to probabilities or time delays of transitions in SPN models.) Then the edge with the larger weight is selected to reduce the number of interpretations, thus reducing ambiguity in a sentence. Another approach is to count event associations of each agent and patient to the object in the participle phrase such as “with the telescope” where “with” is the participle and “telescope” is the object.

5.2.1.2 Different modes of information provide inference toward SPN synthesis of interpretations to reduce sentence ambiguity. Bourbakis and Mills [90] include an example of image data being converted to SPNs. If SPNs from image data are connected to SPNs from text, than a sentence such as “I saw Tom with the telescope” could be combined with SPN information about distances between I, Tom, and the telescope. If the distance between Tom and the telescope is much closer than I and the telescope, then a proper meaning interpretation would be that I saw Tom who had the telescope. Otherwise, I would be closer to the telescope and therefore could be holding the telescope to see Tom.

5.2.2 Discussion of tasks with approaches to solutions.

The above suggested approaches require the following tasks:

5.2.2.1 An automated means of detecting ambiguous sentences:

Our implemented solution is to key on prepositional phrases. For example: “I saw Tom with the telescope” contains the prepositional phrase: “with the telescope.” If this phrase was the first part of the sentence, then the sentence would read “With the telescope, I saw Tom.” which means I had the telescope and used it to see Tom. Thus, the prepositional phrase can occur either at the beginning or ending of that particular sentence. We construct an SPN model that contains each position that the prepositional phrase (or phrases) can occur in the sentence. Consider each noun in the sentence, that is used in a context outside of the sentence, to be associated with the object within the prepositional phrase. In the example, nouns I and Tom could be associated with the noun telescope in the propositional phrase. The parser output stream provides a parts of speech tag “pp” which appears before each participle phrase. This result in another SPN place containing “with” could trigger the construction of a color marking to enable an interpretation.
5.2.2.2 Detecting and gathering the right information that would impact which interpretation to select:

Some information sources impacting interpretation when ambiguity is present in a sentence include:

1. Time of encompassing event, (2) locality of a noun (in the prepositional phrase) with respect to localities of other nouns in the sentence in question, and (3) presence of the same noun associations in other sentences or phrases and (4) their distance from the sentence in terms of number of nodes in a path or number of SPN places. We chose to implement the presence of the same noun associations in the context of other sentences and phrases by counting the number of event associations of each agent and patient within the sentence (or sentences) under investigation for ambiguity. This was selected due to its minimal complexity involving a simple counting approach, its uniqueness for detecting events through the agent, action, patient backbone of our graph representation, and for its originality.

5.2.3 Detailed steps in SPN synthesis to find event association from context.

In order to find the event associations of agents and patients associated through actions on objects from a participle phrase, the following steps are taken to detect possible interpretations of a sentence within the SPN graph representation of a sentence or paragraph.

1. Generate separate participle phrases in graphs for each interpretation (copy patient’s participle phrase and to a new participle phrase for the agent).
   Example: I saw Tom with the telescope. \( \rightarrow \) I with the telescope saw Tom with the telescope.

2. Combine each SPN interpretation into one SPN graph
   For each interpretation: add a place with a colored marking to enable (or disable) the interpretation

3. Use SPNs from other sentences with the same label.

4. Use context to control which interpretations get enabled by the colored markings from association count. (For example: I with the telescope saw Tom.)

5.2.4 The SPN Synthesis Process

The goal of SPN synthesis is to reduce event association ambiguity in a sentence. It uses SPN graphs to control the interpretation of a sentence by detecting its effect on other sentences within its context. It detects context (external event association) that controls the states of colored markings to reduce the number of interpretations in an SPN graph representation of a sentence.

The compressed graph (or SPN graph) provide an efficient medium for detecting context effecting state of enabling colored markings of SPNs. All agents and patients have unique word labels. Thus, all
events of an agent or patient happen at one location in the compressed graph/SPN graph. The agent or patient (within the sentence) having the highest count of event associations (within its context) gets its prepositional phrase (i.e., interpretation) enabled by its state being set.

To synthesize the context of each agent or patient in an SPN graph representing the text, only one surrounding region of that word needs to be analyzed. Since each agent has an adjacent action between it and its patient, then all event associations of the agent (surrounded by its context) are within two neighborhoods from its vertex. Two neighborhoods away from a vertex can be reached by finding all its successor vertices of each successor of the vertex. This involves finding successors of the unique agent’s successors.

Counting associations of an agent (or patient) vertex can be accomplished by counting successors (patients) of its successors (actions) of the agent that has the same word as the agent or patient or object (within its prepositional phrase).

For each agent and patient with its propositional object in a sentence with multiple interpretations, the following is required: (1) Locate agent (or patient) in the compressed graph (or compressed SPN graph) with the same label (word) as the agent (or patient). (2) Find all of its successors. (These are actions.) (3) Find all successors of the above successors. (4) Locate which ones have the same label as the object of the agent’s (or patient’s) prepositional phrase. (5) For each object found, increment a counter dedicated to the agent (or patient) which is in an array indexed by the agent’s or patient’s sentence (or phrase) index. (6) Compare counts of competing agents and patients within the sentence (with multiple interpretations). (7) Then, the agent or patient with the most counts gets enabled by its place’s state which enables access to its prepositional phrase. (8) The interpretation with the most counts is enabled (with green) and all other within the same sentence (or phrase) get disabled (with red). Table 5.

5.3 SPN results of program output showing conversion from graphs to SPN diagrams.

The following graph to SPN diagram conversions are outputs from the graph program which uses parsed input from the Stanford parser and which feeds output to the JUNG2 graph visualization library that created vertices and edges from the methodologies within the graph program. Stochastic Petri Net (SPN)
diagrams are created by changing the shapes of the nodes using transformers discussed in this chapter. These examples represent compressed graph to SPN conversions. To maximize visual clarity of the graphs, only agent, action, and patient nodes are shown in the graphs and SPN graphs within this section.

5.4.a Compressed (7 patients)  
5.4.b SPN Compressed (7 patients)

5.5 Two sentences: A resistor example of events.  
a. Text Input: A resistor in a radio burned out and short circuited a transformer which caused the radio's speaker to make a very loud noise. The loud noise caused the driver of the car, which contained the radio, to drive off the road and run into a barn containing several chickens that became very alarmed and laid several eggs.

5.5.a.  
5.5.b.

5.6 Three phases in sentence - Text Input: My seeing the telescope caused Tom to hide and made Nancy run home.
5.7 Complex sentence with phrases – Text input:

My seeing Jack kick the ball, caused me to run and tackle Tom, while he was running, and keep him from scoring more points and winning the game.

5.8 Ten sentences – Text Input:

I saw Tom with the telescope. The telescope sat on the table. The table and telescope was seen by all of us. We had a great time at the party. It was Tom's birthday. Tom received several presents from all of us. The telescope was a Tom's favorite present. The telescope appeared expensive. I looked through the telescope. A great time was had by all.
5.8.a Compressed: 4 agents and 8 patients are shown.

5.8.b SPN diagram of compressed graph.

5.9 Long sentence – Text Input: From example in Stanford parser documentation

The strongest rain ever recorded in India shut down the financial hub of Mumbai, snapped communication lines, closed airports and forced thousands of people to sleep in their offices or walk home during the night, officials said today.

5.9.a Uncompressed Graph

5.9.b Uncompressed SPN

5.10 Long text description of an image. (Example text is in chapter IV.)
5.10.a Compressed Graph of Image description

5.10.b SPN Diagram of converted compressed graph.

5.11 Graph and SPN Diagram from Long text (approximately one to two pages)

5.11.a SPN diagram of Compressed graph.

5.11.b SPN diagram of Compressed graph.

Figure 5.11 Graphs and an SPN diagram are shown for a long text (one to two pages).
VI. IMPLEMENTATION DETAILS AND RESULTS OF SPN SYNTHESIS

This portion of the dissertation contains detailed justification for using SPNs (in particular, Colored SPN’s) as state machines in the SPN synthesis, detailed (step by step) implementation details of how the SPN synthesis discussed in the previous chapter was conducted to find solutions to ambiguity in a sentence. It also contains a discussion and implementation details of SPN synthesis of text generated from images. The purpose of the latter work is to show how other modes of information, in this case textual descriptions of imagery, can be used in the SPN synthesis. Updated results of this work are also included.

6.1 Justification for Using SPNs to Model a State Machine Approach to NLU

Our use of SPNs as state machines is explained in symbolic form and illustrated in figure 6.1.

![Figure 6.1 a. Basic state machine](image)

![Figure 6.1 b. V changes state of A to A' and P to P'](image)

![Figure 6.1 c](image)

Figure 6.1. Shows how a state machine SPN in 6.1c with Agent A → Verb V → Patient P’ is derived from a basic state machine in 6.1a. 6.1b shows transition V changing state A to A’, and state P to P’, each over a time interval t1-t0.

With the state machine approach, as illustrated in figure 6.1, transition V represents an event. The state of agent A at time t0 may change state to A’ at time t1. It causes an action V to change the state of patient P at time t0 to P’ at time t1. To represent events in natural language, our focus is on agent A causing action V, represented by a transition, which results in the patient receiving a state P’ after a period t1-t0. Other state changes such as A transitioning to A’ and P transitioning to P’ may be occurring somewhere within the represented NL text, but a particular event of interest (to the reader of a sentence) is represented by an
agent (place A) impacting via a verb or action (transition V) resulting in a change to the object or patient (place P’). Since time elapses during the occurrence of an event, then the state of A exists at time $t_0$ and P’ exists at a later time $t_1$ with the transition V taking a finite amount of time, within the period of $t_1 - t_0$. Thus, with the state machine approach, we can model events using states. Using them in conjunction with SPNs, we can conveniently model an entire text containing several events.

Another factor in the justification for using Stochastic Petri Nets with its state machine capability to represent natural language is to show its compatibility with the mathematical building blocks of Colored Petri Nets (CSPNs), as defined in Haas’ book [89]. The following definitions come from the SPN building blocks of the SPN mathematical model.

From Peter Haas, Stochastic Petri Nets – Modeling, Stability, Simulation, Chapter 9, Page 387:

Places: $D = U_{e \in D} (\{d\} \times UD(d)) = \{d_1, d_2, \ldots, d_L\}$

Transitions: $E = U_{e \in E} (\{e\} \times UE(e)) = \{e_1, e_2, \ldots, e_3\}$

Immediate Transitions: $E' = U_{e \in E'} (\{e\} \times UE(e)) = \{e_1, e_2, \ldots, e_M\}$

Set U of colors. Color domains $UD(d) \subseteq U$ for $d \in D$

and $UE(e) \subseteq U$ for $e \in E$ with values of non-negative integers.

Input (output) incidence function: $w_+ (w_-)$ is defined on:

$U_{e \in E, d \in D} (\{e\} \times UE(e) \times \{d\} \times UD(d))$ with values of non-negative integers.

This “determines when a transition is enabled…”[89]

Colored Petri Nets include the set U of colors defined above. The colors serve as an enumeration of different types of tokens residing within SPN places. When modeling something that can be broken into similar or repetitive portions and these portions can be represented by colors, then its complexity can be significantly simplified as noted in some of Hass’ CSPN examples. This approach is similar to using the representation capability of UML in system designs.

6.2 How this modeling approach contributes to natural language understanding

Using CSPNs provide a way to model states within places to enable transitions of agents to impact patients. This is how we model events throughout a text. Since events occur within a larger context
surrounding a sentence or phrase, we use the state of these events to convey associations between agents or patients to objects within a phrase such as a participle phrase of a patient. Then by counting the number of associations of a particular noun (agent or patient) of interest, the proper interpretation within a sentence is selected based on influence from the context of the remaining text.

An important application to the above approach is to use context to reduce ambiguity of a sentence. Our particular focus is in event associations that have a direct mapping to our agent-action-patient approach that we use as a backbone representation occurring throughout a text.

Stochastic Petri nets (SPNs) provide a capability to “model very large or complex systems” [89]. Colored stochastic Petri nets associate colors with tokens (also called markings) within places and through transitions, thus providing more concise specifications, as discussed by Peter Haas in chapter 9 of his book [89]. Since colored tokens provide an enumeration distinguishable by color, several tokens can be represented by one dot that can have a variety of colors within a place (circle). This representation significantly reduces the number of places and transitions needed to specify a system. When SPN modeling is limited to a Boolean presence or absence of a one color (i.e., black) token within places and flowing through transitions, the result is a multiplicity of places and transitions, compared to a less complex configuration of colored tokens. Thus, when a system is composed of several, but similar, subsystems, color can significantly simplify the model of a subsystem, thus simplifying the structure of places and transitions within the Petri net model of a system. The color of a token can represent the state of a system, subsystem, or component. Expressing state using colored SPNs (or CSPNs) provides a means of modeling several common states that can be represented throughout the structure of a system. This is especially useful for modeling the functionality of large and complex systems.

Natural language can also be significantly complicated to model, especially when it is generated from large, multiple documents, or information from more than one mode, such as imagery, motion pictures, sensors, speech, etc. Like subsystems, several portions of a document or set of documents may contain similar information. Contents of text can help resolve ambiguity, in our case, event association ambiguity. When the ambiguity is associated with the relationship of nouns to verbs within various sentences, the contents of the same or similar information from other sentences within a text can tighten the
range of meaning within a sentence, making it less ambiguous. When this meaning is about events and association of agents, actions, or patients, their interpretation can be labeled by states within a CSPN model of the text. In today’s media, information can arrive in a variety of modes. From the internet, such forms of information can become vast and complex. Such complication from text (or generated text from various modes) is similar to the complication of large, complex systems that have often been specified by states within SPNs. Representing similar meanings from text as states can be similar to representing subsystem states within large systems. Thus, the usefulness of SPNs, including states specified by colored tokens, is applicable to modeling NL text, especially when the text is large and contains several events. Over the past several years, SPNs, including colored SPNs containing states in its model, have successfully modeled large, complicated systems. The above discussion shows how modeling NL text with SPNs containing a state machine representation using colored tokens should result in an effective modeling technique, especially for large text containing events.

6.3 SPN Synthesis for Reducing Event Association Ambiguity

Event association ambiguity results when the meaning of a sentence, in regards to an event, can have more than one interpretation. Sentence ambiguity may result in reckless writing, such as referring the pronoun “it” to one of a multiple of direct nouns, e.g., weather, tree, etc. However, even with careful writing, ambiguity can often creep into the text due to the natural language’s tendency to be unspecific and leave more than one meaning. Such ambiguity can have undesirable impacts in some fields, for example medical papers.

SPN synthesis was discussed in the previous chapter (paragraph 5.2). In this chapter, more implementation detail is presented, backed with results from the program. After text is parsed and annotated with parts of speech labels (from the Stanford Parser), the program generates a graph representation, compresses it, converts it to a compressed SPN graph, detects when an event association has more than one interpretation within a sentence, generates a few more places and edges to capture each interpretation it detects, and generates extra places and edges to hold state information. The states, represented by colored tokens, are manipulated from weights of minimized vertices resulting from the compression process. These weights have the same effect as probability, thus providing the statistical
functionality of the SPN model. This weight information, calculated by counting vertex reductions during compression, captures the relative importance of certain context that can be used to resolve ambiguity. This weight information is passed to the places containing states. As a result, context from the rest of the text controls the selection of interpretation to include in the SPN model of the text. The interpretation focuses on event association, represented by the agent, action, and patient information within the compressed and analyzed SPN.

Figure 6.2 Shows the SPN Synthesis Process for finding the impact that agents and patients from a sentence have on objects within their prepositional phrases and reducing ambiguity of the sentence. This is accomplished by calculating the number of associations they have in their context, whether it be a paragraph, page, or whole document. From right to left, an agent or patient from a sentence (right) is located in a compressed SPN graph (left). Within two vertices (successors of successors), objects are found. The count of agent to object associations are compared with patient to object associations. The state of the enable place is set for the winner agent or patient to access its participle phrase containing its object, and thus, enabling the winning interpretation of the sentence.

The following list contains steps for calculating the proper interpretation of a sentence from its context. It involves counting event associations between each agent and patient to its object appearing in a compressed SPN representation of its context.
Calculating the amount of event association of agents and patients to objects in context:

For each agent and patient with its propositional object in a sentence with multiple interpretations:

Locate agent (or patient) in the compressed graph (or compressed SPN graph) with the same label (word) as the agent (or patient). The same agents are compressed in SPN graph representing context.

Find all of its successors. (These are actions.)

Find all successors of the above successors (patients or objects).

Locate which ones have the same label as the object of the agent’s (or patient’s) prepositional phrase.

For each object found, increment a counter dedicated to the agent (or patient).

Compare counts of competing agents and patients within the sentence (with multiple interpretations).

Then, the agent or patient with the most counts gets enabled by its place’s state which enables access to its prepositional phrase.

The interpretation with the most counts is enabled (with green) and all other interpretations within the same sentence (or phrase) get disabled (with red). (Colors red and green were selected for clarity.)

An SPN color marking represents a condition (in our case, an interpretation). When set, that color marking is enabled through a transition to the next place.
The diagram in figure 6.3 shows this step by step process in transforming text to an SPN synthesis.

6.4 SPN Synthesis Example to detect and select interpretations from context

Example Text (10 Sentences): I saw Tom with the telescope. The telescope sat on the table. The table and telescope was seen by all of us. We had a great time at the party. It was Tom’s birthday. Tom received several presents from all of us. The telescope was Tom’s favorite present. The telescope appeared expensive. I looked through the telescope. A great time was had by all.
Fig. 6.4e Synthesis: Agent’s Participle Phrase is automatically generated to complete all interpretations. Three Enable Places are added. This is before states are set.

Fig. 6.4f Synthesis: Shows three states set. Agent (I) receives a Green (enabled) state while Patient (Tom) and Action (Saw) gets a Red (Disabled) state.
6.5 Compression Algorithm

The graph compression algorithm results in the compressed graph and compressed SPN. It simplifies both the SPN synthesis of resolving multiple interpretations, discussed in paragraph 6.2, and the SPN synthesis of multiple image file descriptions, discussed in paragraph 6.4. Due to its importance in simplifying the SPN synthesis, details of our compression approach are presented here.

The diagram in figure 6.5 below illustrates the compression process corresponding to the pseudo code representation below in figure 6.6.

Figure 6.5 a. Before adding edges to i

Figure 6.5 b. After adding edges to i

Figure 6.5 Diagram of the compression process, showing vertices i and j, with j having a neighborhood of vertices v. Before removing j, edges from v_i and v_o which are in and out edges of j are added to vertex i. Then, when vertex j is removed, their attached edges v_i and v_o remove automatically. The weight of vertex i is incremented. This above process for combining two vertices is located within two for loops (the outer loop for i and the inner loop for j).

This same compression process in the diagram (figure 6.5) exists for both compressing agent and patient vertices in a graph and for compressing action vertices. The corresponding pseudo code is shown in figure 6.6 below. The only difference between compressing noun (agent and patient) vertices and verb (action) vertices is the two if statements within the pseudo code.
The following pseudo code (mixed with Java code) describes the compression part of our program. This code contains “for” loops, “if” statements, and nested calls to the JUNG2 [83, 84, 85, 86] library to manipulate the graph constructed from previous steps of the program.

```
for vertex i { // iteratively loop through all the vertices “i” of the graph. (we will keep i)
    for vertex j { // iteratively loop through all the vertices “j” of the graph. (we will remove j)
        if ((i and j are not the same vertex id) and (j has not been removed)) { 
            if (i and j are agents or patients) { // Compression here is for agents and patients
                if (labels of i and j are equal) { // i and j contain the same word in their labels.
                    // Get all the predecessor vertices of vertex j and move them to vertex i
                    for (Object v_i: graph.getPredecessors(j)) { Get all predecessors of j
                        for (Object e_i: graph.getInEdge(j)) { // get an incoming edge to j
                            graph.addEdge(e_i, v_i, i); // add this edge e_i between v_i and i
                        }
                    }
                    // Do the same thing with successors of j:
                    // Get all the successor vertices of vertex j and move them to vertex i
                    for (Object v_i: graph.getSuccessors(j)) { Get all successors of j
                        for (Object e_i: graph.getOutEdge(j)) { // get an outgoing edge to j
                            graph.addEdge(e_i, i, v_i); // add this edge e_i between i and v_i
                        }
                    }
                    // Now that all edges to j are transferred to i, remove vertex j.
                    graph.removeVertex(j); // Remove vertex j
                    vertexWeight[i]++; // increment the weight of vertex i since we combined into i
                    vertexRemoved[j] = true; // Mark the array element (removed) associated with j.
                    // We have compressed one vertex j.
                } // Once we loop through all the other vertices j in the graph, we will get another i.
            } // Once we loop through all the vertices i in the graph, we have compressed all the
            // agent and patient vertices that are labeled with the same word to one vertex.
            // The same code is used for compressing actions, except that either all predecessors
            // or all successors of both vertex i and j must have the same label (containing the same
            // word) before they can be compressed. This avoids agents impacting the wrong
            // patients via an action that is over-compressed.
        }
    }
}
```

Figure 6.6 Pseudo code (mixed with Java code) of the compression process.

The above compression code forms chains of agents, actions, and patients with each agent and patient weighted according to how many vertices compressed into the one vertex. This weight can represent probability of this element within the context effecting interpretations associated with the
vertex. Prepositions (such as "with") adjacent to the vertex can hold interpretations, which can be controlled by added places holding colored tokens.

6.6 SPN Synthesis of Multiple Image File Descriptions

The difference between image description files can represent: (1.) changes in location which imply movement of one or more objects in the image or (with respect to the viewer) change of projected view of the objects that appear to the viewer as changes in location, (2.) changes in position of objects relative to themselves, which imply changes in rotation or (with respect to the viewer) changes in the relative angle of view of the object, and (3.) interactions between objects such as the before and after effects of an event such as collision of two or more objects or interaction between a moving object and a stationary one such as a table or floor. If the changes in object location and/or position are the result of movement of the objects, then some time is expected to elapse (based on basic principles of physics). If the differences of images are the result of different projected images of view to a camera or viewing device, then the cause of the differences would be the result of the viewer or viewing device (external to the images) changing position. In any event, the modeling of such changes can be represented by stochastic Petri nets. To model changes in state of the two images, a state machine would be appropriate. However, for integrating such changes with other SPN representations, such as SPN models of the image descriptions and events, then colored Petri nets would be a reasonable choice. Using this same model across interfaces or component parts of a system can help avoid or reduce potential errors in integrating from design through implementation and testing. This is a basic software (and even system) engineering principle, where the most troubling part of development occurs at the integration phase.

Associating changes in object locations and positions, across image description files involves obtaining files of image descriptions in the form of text, converting them to graphs, compressing the graphs, changing the compressed graphs to compressed SPNs, and creating SPN places and transitions with colored tokens to associate similarities and differences between the image descriptions in the two files. Figure 6.7 diagrams this process of associating two different files of image descriptions.
In figure 6.7, an image from picture 1 and 2 are described in text, converted to graphs of objects designated at specific locations and in specific positions included in the description. The program converts the text to a graph, then compresses it, and converts it to a compressed SPN graph. The SPN synthesis process captures objects from the two graphs and incorporates differences by inserting SPN places with colored markings, and transitions. Similarities of objects (produced from compression) are passed to the combined SPN incorporating information from both images. Differences such as whether certain objects are visible or hidden are used to add the places and transitions with colored tokens for state information. The color of each state provides the mechanism for including differences of the two objects into the combined SPN graph. As a result, the information contained in the two separate images is included in the combined and compressed SPN representation. This result could be used with other SPN files generated from text to present an integrated SPN representation of a whole document or multi-documents that include information from pictures.

Figure 6.8 below is an example of associating two image description files. Steps to this approach were discussed above in figure 6.7. Picture 1 and 2 each show an image containing objects. With an image to text translation tool (not available for this research) the image is converted to text. Our program begins with the image description files (figures 6.8 c. and d.) by providing a conversion from text to graph, to compressed graph, to compressed SPN, and SPN synthesis. This adds useful information to event association for NLU that is external to the (parsed) text input of the program. Picture 1 shows a car,
airplane, helicopter and wrench-tool. Picture 2 shows three of the same objects but replaces the forth object, a helicopter with a hammer. Changing the view, changes the image.

Number of objects: 4 objects recognized in the input image.

Type of objects: A yellow car, a brown airplane, a black helicopter and a silver wrench-tool

Location of the objects: The airplane is on the upper left part of the image. Its position is “nose-SE-side”. The helicopter is on the upper right part of the image. Its position is “nose-NE-side”. The wrench-tool is on the upper center part of the image. Its position is “open-edge-NE-side”. The car is on the lower center part of the image. Its position is “nose-W-side”.

Associations of the objects:
The car is 2.1 the length of the airplane, 2.3 the length of the helicopter, 1.3 the length of the wrench-tool;
The car is the biggest in length object;
The airplane is the second in length object;
The helicopter and the wrench-tool have similar length;

Number of objects: 4 objects recognized in the input image.

Type of objects: A yellow car, a brown airplane, a black handle hummer and a silver wrench-tool

Location of the objects: The airplane is on the lower central part of the image. Its position is “nose-NE-side”. The hummer is on the middle-diagonal of the image. Its position is “back-side-top-NE” The wrench-tool is on the right-down part of the image. Its position is “open-edge-top-NW-side”. The car is on the upper left part of the image. Its position is “right-side-nose-NE”.

Associations of the objects:
The hummer is the longest object with 2.1 times the length of the wrench-tool, with 3 times the length of the airplane, with 1.3 the length of the car;
The car is 1.2 the length of the wrench-tool, 1.8 of the airplane;
The car is second in length object;
The airplane is the smallest object;
Figure 6.8 c and d are the text descriptions of the two images. The text description is segmented into headings signifying number, type, location, association of the objects. Text generation is from a tool that is not part of this research. Our research for images starts with the text and generates the graphs, SPNs, and performs the graph compression and the SPN synthesis.

Text is converted to a graph representation as shown in figure 6.6 e and f. Color specifies whether the vertex is an agent (light blue), action (green), or patient (yellow).
The graphs are compressed as shown in figure 6.6 g. and h. The compression algorithm reduces the labels of all duplicate agents and patients to one unique agent and patient (unique words). Similarities and differences in objects, location, and position are more easily calculated from the compressed graph.

Compressed graphs are converted to compressed SPNs as shown in figure 6.6. i. and j. Agents and patients form places and actions form transitions. The actions (transitions) show agent places effecting patient places through action transitions. Compression is activated for agents and patients.

**New Colors:** Transitions (Black) support New Enable Places (Green Circles) to transfer objects from image-1 to image-2.

**Red** Circle is an OUT state in a place disabling a (Pink) Transition which stops transfer of an object from image-1 to image-2.

**Purple** Circle is an IN state in a place generating a new object in image-2 that does not appear in image-1.

Pink transitions support IN and OUT places.

Blue, Green, and Yellow are agents, actions, and patients.
A clearer view of SPN graphs of images 1 and 2 and the combined image results from using a different path of deriving graphs. These SPN graphs come from compressed graphs without any common words, yielding agents, actions, and patients.

**Figure 6.8 l. Image-1 Compressed SPN**

**Figure 6.8 m. Image-1 Compressed SPN**

**Figure 6.8 n. Image-2 Compressed SPN**

Agents, Actions, and Patients (without PP)

<table>
<thead>
<tr>
<th>Differences.OUT.1[20]</th>
<th>= helicopter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differences.OUT.1[41]</td>
<td>= helicopter</td>
</tr>
<tr>
<td>Differences.IN.2[160]</td>
<td>= hammer</td>
</tr>
</tbody>
</table>

Text file output from program shows Similarities (Enable On), Differences (OUT IN). (Reformatted for clarity)

Objects: car, airplane, and wrench-tool transition from image-1 and image 2. OUT stops helicopter from going to image-2. IN generates a new object hammer in image-2
SPN Synthesis (figure 6.8.k) of the two compressed SPN graphs (6.8.i and j) consists of combining the two SPN graphs into one SPN graph by (1.) moving the two graphs into one image, (2.) finding the differences of the two SPN graphs and what part of the SPN graphs they are located, (3.) linking the two SPN agents of patients where their differences occur with an incoming edge, followed by a transition that is enabled by a place with a colored token acting as a state, and the transition followed by an outgoing edge to the corresponding place of the second SPN graph where the difference occurs. The addition of a place with a colored token acting as a state is illustrated in figure 6.6.m. One difference in example in 6.6 consists of the presence of a helicopter in one image and the absence of the helicopter in the other.

A second example of associating two different image description files into an SPN graph in figure 6.9 explains the process in figure 6.7 above. Picture 1 and 2 each show an image containing objects.

<table>
<thead>
<tr>
<th>Figure 6.9 a</th>
<th>Picture-1 Before</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Objects:</strong></td>
<td>Boy with a Bat, Cylinder, Block</td>
</tr>
</tbody>
</table>
| **Description:** | Before: Bat is raised. Cylinder and Block is sitting on floor.  
After: Bat is lowered. Cylinder has rotated counter-clockwise and moved to the upper left. Block has rotated clockwise and moved slightly to the right. Boy is at the same place. |
| **Number of objects:** | 4 objects recognized in the input image |
| **Type of objects:** | boy, bat, cylinder, box |
| **Location of objects:** | The boy is on the lower left part of the image. Its position is “face-front”. The bat is on the lower left part of the image. Its position is “head-raised”. The cylinder is on the lower center part of the image. Its position is “on its side facing the ground”. The box is on the middle right part of the image. Its position is “on its side facing the ground”. |
| **Association of the objects:** | The boy is holding the bat. The bat is in front of the cylinder. The cylinder is near the box. The box is facing the boy and the cylinder.  
The boy is larger than the box. The box is larger than the cylinder. The cylinder is larger than the bat. |

<table>
<thead>
<tr>
<th>Figure 6.9 b</th>
<th>Picture-2 After</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of objects:</strong></td>
<td>4 objects recognized in the input image</td>
</tr>
<tr>
<td><strong>Type of objects:</strong></td>
<td>boy, bat, cylinder, box</td>
</tr>
</tbody>
</table>
| **Location of the objects:** | The boy is on the lower left part of the image. Its position is “face-front”.  
The bat is on the lower left part of the image. Its position is “head-down”.  
The cylinder is on the middle left part of the image. Its position is “tilted-to-left”. The box is on the middle right part of the image. Its position is “tilted-to-right”. |
| **Associations of the objects:** | The boy is holding the bat. The bat is below the cylinder. The cylinder is beside the box. The cylinder is off of the ground. The box is away from the cylinder. The box is off of the ground.  
The boy is larger than the box. The box is larger than the cylinder. The cylinder is larger than the bat. |

**Figure 6.9 c. Text Description before boy with bat hits cylinder that crashes into box (event)**

**Figure 6.9 d. Text Description after boy with bat hits cylinder that crashes into box (event)**
Figure 6.9 e. Graph from description of picture-1.

Figure 6.9 f. Graph from description of picture-2.

Figure 6.9 g. Compressed Graph from picture-1.

Figure 6.9 h. Compressed graph from picture-2.

Figure 6.9 i. SPN graph from picture-1.

Figure 6.9 j. SPN graph from picture-2.
In this example, the four objects in picture 1 are also in picture 2. Therefore, each object is enabled to be present in graphs and SPN graphs representing picture 1 and 2. IN and OUT with their transitions are still present, but none of these objects should be connected to them. If more of the same object exists in one image, then they are both in the similarity list and differences list. Some descriptive words are in list IN.

New Colors: Transitions (Black) support New Enable Places (Green Circles) to transfer objects from image-1 to image-2.

Red Circle is an OUT state in a place disabling a (Pink) Transition which stops transfer of an object from image-1 to image-2.

Purple Circle is an IN state in a place generating a new object in image-2 that does not appear in image-1.

Pink transitions support IN and OUT places.

Figure 6.9 k. Combined SPN graph from picture-1 to picture 2.
7.1 The conclusions from the two surveys are at the end of each survey (in chapters II and III).

7.2 During our survey of NLP and NLU publications and the literature searches that we completed, we found no existing publicized research that used Stochastic Petri Nets (SPN) for NLP or NLU. Thus, we conclude that research in using SPN modeling for NLU for document and event association has never been accomplished. Not only do SPN graphs provide the structural capability of graphs in capturing information, they provide state information that can help resolve event association within the area of NLU research. One benefit, as mentioned in [90], of such research is to provide a new method for resolving sentence ambiguity involving events.

7.3 We have implemented a graph methodology for capturing natural language from text that represents event association in terms of agents performing actions on patients within phrases, sentences, and relatively long text. We have used the JUNG-2 library in finding ways to create the graph visualizations needed to illustrate the underlying functionality of our methodology as it progresses.

7.4 We have implemented, with the aid of the JUNG2 library, a graph to SPN model conversion capability in order to represent what we have represented graphically into an SPN representation.

7.5 We have an approach, initially illustrated by Bourbakis and Mills [90], for capturing all interpretations of a sentence and representing them in a SPN graph. By adding places holding color markings, we can control which interpretations get represented based on what contextual information we find from other synthesis we discover and implement in our research.

7.6 We implemented a working graph program that captures event information from text, implemented a graph to SPN conversion routine, and developed and implemented an approach for representing interpretations of text with SPN models and synthesizing them to resolve event association ambiguity within sentences.

7.7 The table below is a summary of contributions provided by this dissertation and its research:
<table>
<thead>
<tr>
<th>Text to Graph</th>
<th>Graph to SPN</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>My Contribution</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developed a program to extract events from parsed text: Agent-&gt;Action-&gt;Patient</td>
<td>Converted Graph to SPN representation. Modeled NL text with SPN graphs. Detected and Modeled Event Associations with SPN graphs. Synthesized SPN model of NL text to capture effect of NL context on sentence. Applied SPN Transactions to represent actions of events.</td>
<td>Synthesized event associations from context of paragraph to reduce ambiguity of sentence. Synthesized similarities and differences (presence or absence of objects) between image descriptions to control In/Out states of SPN places that model object inclusion or exclusion between images.</td>
</tr>
<tr>
<td>Handle complex sentence structure: Multiple /nested phrases, Multiple subjects, verbs, objects, Active and passive voice. Applied a network visualization library to NL representation.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 7.1 Summary of Research Contributions
REFERENCES

Chapter II:


Chapter III:


Chapter IV


83. Java Universal Network/Graph Framework (JUNG-2.01), [http://jung.sourceforge.net](http://jung.sourceforge.net)


Chapter V


APPENDIX A

Our approach to using the Stanford parser and the JUNG2 graph visualization library:

The Stanford parser:

We used an English version of the NL parser [80, 81, 82], developed by Stanford University, that converts the NL text into a parse tree which our program reads as a string of parentheses, symbols, and words. The parser works for several languages and has an extensive user base in several countries. The English parser is statistically trained by the Penn Tree Bank consisting of millions of sentences.

The JUNG2 library:

We used a graph visualization library called the Java Universal Network/Graph Framework (JUNG-2.01) [83, 84, 85, 86], developed by the University of California at Irvine, to convert words into vertices and convert relations between agents, actions, and patients into edges that produce visualizations of the graphs in both java swing views and image files. The JUNG2 library includes a common collections library [87] that includes transformers and the COLT math library [88].