

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/319516375>

SigmaC : Document analysis based Automatic Concept Map Generation for Enterprises

Technical Report · September 2011

DOI: 10.13140/RG.2.2.29284.88960

CITATIONS

0

READS

493

6 authors, including:



[Nisansa de Silva](#)

University of Moratuwa

62 PUBLICATIONS 348 CITATIONS

[SEE PROFILE](#)



[Amal Shehan Perera](#)

University of Moratuwa

88 PUBLICATIONS 375 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Real time computation of Value at Risk [View project](#)



Spatio-temporal Forecasting of Dengue Outbreaks using Machine Learning and Big Data [View project](#)



SigmaC

Document analysis based Automatic Concept Map Generation for Enterprises

Project Supervisors:

Dr. Shehan Perera

Mr. Nisansa de Silva

Group 04:

K.N.J.Fernando 080108K

H.M.T.C.Herath 080154V

E.L.Karannagoda 080214G

M.W.I.D.Karunarathne 080226U

Date of Submission:

04th September, 2011

Computer Science and Engineering Departement

University of Moratuwa

**This report is submitted in partial fulfillment of the requirements for the award of the degree of
Bachelor of the Science of Engineering at University of Moratuwa, Sri Lanka**

DECLARATION

This thesis is a report of the project work and research carried out in the Department of Computer Science and Engineering, University of Moratuwa, from December, 2011 to August, 2012. Except where references are made to other work, the content of the thesis is original and includes the work done in collaboration as a team. This thesis has not been submitted to any other university.

ABSTRACT

Project Title: SigmaC Document analysis based Automatic Concept Map Generation for Enterprises

Ever growing knowledge bases of enterprises present the demanding challenge of proper organization of information that would enable fast retrieval of related and intended information. Document repositories of enterprises consist of large collections of documents of varying size, format and writing styles. This diversified and unstructured nature of documents restrict the possibilities of developing uniform techniques for extracting important concepts and relationships for summarization, structured representation and fast retrieval. The documented textual content is used as the input for the construction of this concept map. Here a rule based approach is used to extract concepts and relationships among them. Sentence level breakdown let these rules to identify those concepts and relationships. These rules are based on elements in a phrase structure tree of a sentence. For improving accuracy and the relevancy of the extracted concepts and relationships, the special features such as titles, bold and upper case texts are used.

This paper discusses how to overcome these challenges by utilizing high level natural language processing techniques, document preprocessing techniques and developing easily understandable and extractable compact representation of concept maps. Each document in the repository is converted into a concept map representation to capture concepts and relationships among concepts described in the said document. This organization would represent a summary of the document. These individual concept maps are utilized to generate concept maps that represent sections of the repository or the entire document repository. This paper discussed how the statistical techniques are used to calculate certain metrics which are used to facilitate certain requirements of the solution. Principle component analysis is used to ranking the documents by importance. The concept map is visualized using force directed type graph which represent concepts by nodes and relationship by edges.

Keywords:

Natural Language Processing, Concept Map, Concepts/Relationships Extraction

ACKNOWLEDGMENT

First and foremost, we would like to thank to our supervisors of this project, Dr. Shehan Perera and Mr. Nisansa de Silva for the valuable guidance and advice. They motivated us to do the best and spending their valuable time supervising the project SigmaC.

Mr. Nuwan Samarasekara, as the external supervisor of the project, giving the initial project proposal and continuous guidance to clearing out the requirements of the project with his industrial experience, we would like to thank him to spending his valuable time for us while having busy schedules.

Dr. Malaka Walpola and Ms. Pivithuru Wijegunawardhana, as the CS4200 Project Coordinators, not only evaluated the project, but also willingly extended his assistance whenever needed, which essentially fine-tuned our project. We would also like to thank the department and the staff for being a great assistance by providing all necessary lab facilities and guidance.

Our special thanks go to Dr. Chandana Gamage, head of the Department, for all the support extended.

Finally, an honorable mention goes to our families and friends who were always behind us and encouraged us to do our best.

Table of Figures

Figure 1 Network representation of three semantic relations among an illustrative variety of lexical concepts	17
Figure 2 Bipolar Adjective Structure	18
Figure 3 - Co-reference generating algorithm.....	20
Figure 4 Co-reference generating algorithm	20
Figure 5 Structure tree generated by Stanford parser.....	25
Figure 6 Overview of main relation extraction sub-tasks.	32
Figure 7 Relationship Representation on a concept map	34
Figure 8 phrase structure tree generated by the Stanford parser	36
Figure 9 RelEx Relationship Extraction Chart	37
Figure 10 SigmaC Pipe Line.....	38
Figure 11 SigmaC Architecture	40
Figure 12 UseCase Diagram	41
Figure 13 Class Diagram Pre Processing Module.....	42
Figure 14 Class Diagram Concept and Relationship Extraction.....	44
Figure 15 Class Diagram Optimization module.....	46
Figure 16 Class Diagram Visualization Module.....	47
Figure 17 Activity Diagram Concept Map Generation.....	48
Figure 18 Activity Diagram Concept Map Editing.....	49
Figure 19 Traverse <p:sp> node to get txt content and bold text of the node (Part of a method).....	53
Figure 20 Traverse <p:sp> node to get style of the paragraph.....	53
Figure 21 Traverse <w:p> to get style of the paragraph	54
Figure 22 Traverse <w:p> node to get text content (to array) and check whether they are bold.....	55
Figure 23 SigmaC Client GUI	57
Figure 24 SigmaC Client Editble Map.....	58
Figure 25 File Drop Box	58
Figure 26 Input file path array and process resulted file.....	60
Figure 27 Method for getting morphological root	68
Figure 28 Optimizing concepts by using morphological root of concept	
Figure 29 Is_a relationship asserting method	70
Figure 30 Optimize 'is_a' relationships.....	71
Figure 31 part_of relationships assertion method	71
Figure 32 Optimizing part_of relationships	72
Figure 33 Implementation of getHypernyms and getMeronym methods	73
Figure 34 Optimize 'part_of' relationships.....	73
Figure 35 ER diagram for the database.....	77

Contents

DECLARATION	2
ABSTRACT.....	3
ACKNOWLEDGMENT.....	4
Table of Figures	5
1. INTRODUCTION	11
1.1. Motivation and Applications.....	12
2. AIMS AND OBJECTIVES	13
2.1. Aim of SigmaC	13
2.2. Objectives of the Project.....	13
3. LITERATURE REVIEW	14
3.1. Introduction.....	14
3.2. Natural Language Processing.....	14
3.2.1. Word Sense Disambiguation (WSD)	14
3.2.2. WordNet.....	16
3.2.3. Anaphora resolution.....	18
3.2.3.1. Anaphora Resolution Frameworks.....	19
3.2.3.1.1. JavaRAP.....	19
3.2.3.1.2. Reconcile.....	20
3.2.4. Sentence Boundary Detection.....	21
3.2.5. Collocation Identification	21
3.2.6. Parts of Speech Tagging	21
3.3. Concept Extraction.....	23
3.3.1. Concept Identification using POS tag signatures	23
3.3.2. Frequency based Key word extraction.....	24
3.3.2.1. Usage of Phrase structure for concept extraction.....	24
3.3.3. Dependency Relationship based concept extraction	26
3.3.4. Semantic data source based concept extraction	26
3.3.5. Usage of formatting information to support concept identification	27
3.3.6. K-gram based method	27
3.3.6.1. K-gram Algorithms	29
3.4. Relationship Extraction.....	31
3.4.1. Common Relationship Extraction Approaches.....	31
3.4.1.1. The Information Extraction Framework	32

3.4.1.2.	Relation Identification and Characterization	32
3.4.1.3.	Relation Extraction and Adaptation	33
3.4.2.	Ontologies	34
3.4.3.	Pattern-based approach to relation extraction	34
3.4.3.1.	Relations of Interest	34
3.4.3.2.	Patterns	34
3.4.3.3.	Discovery and Pattern Expression	35
3.4.3.4.	Hypernymy Extraction	35
3.4.3.5.	Analyzing the syntagmatic relations in a sentence	35
3.4.4.	Relationship Extraction Frameworks	36
3.4.4.2.	RelEx - Relation extractor	36
4.	DESIGN	38
4.1.	System Overview	38
4.2.	System Architecture of SigmaC	39
4.2.1.	Introduction	39
4.2.2.	Description of SigmaC Architecture	39
4.2.3.	Architecture Diagram of SigmaC	40
4.2.4.	Introduction	40
4.2.5.	Description of SigmaC Architecture	40
4.2.6.	Use Cases	40
4.2.6.1.	Introduction	41
4.2.6.2.	Use Case Diagram	41
4.2.7.	Class Diagrams	42
4.2.7.1.	Pre-Processing Module	42
	42
4.2.7.2.	Class Description	42
4.2.7.4.	Concept & Relationship Extraction module	44
4.2.7.4.1.	<i>Class Diagram</i>	44
4.2.7.4.2.	<i>Class Description</i>	45
4.2.7.5.	Optimization Module	46
4.2.7.5.1.	Class Diagram	46
4.2.7.5.2.	Class Description	46
4.2.7.6.	Visualization Module	47
4.2.7.6.1.	Class diagram	47

4.2.8.	Activity Diagrams	48
4.2.8.1.	General process of generating concept maps	48
4.2.9.	Concept map editing process	49
5.	IMPLEMENTATION	50
5.1.	Pre Processing Stage	50
5.1.1.	Adaptors	50
5.1.1.1.	Open Documents Adaptor.....	50
5.1.1.2.	Microsoft Word Documents adaptor (.doc /.ppt).....	50
5.1.1.3.	Microsoft PowerPoint Open XML Presentation (PPTX) Adaptor.....	52
5.1.1.4.	Microsoft Word Open XML Document (DOCX) Adaptor.....	54
5.1.2.	Intermediate XML generator.....	56
5.1.3.	Client side GUI implementation	57
5.1.4.	Anaphora Resolution.....	60
5.2.	Concept and relationship extraction.....	60
5.2.1.	Data Structures for representing concept map	60
5.2.2.	Patterns.....	63
5.2.2.1.	Pattern loading	64
5.2.3.	Optimizing Module	68
5.2.4.	Concept Ranking.....	74
5.3.	Visualization Server with Database	75
5.3.1.	Database Implementation.....	77
6.	RESULTS AND ANALYSIS	78
6.1.	Result comparison before/after resolving Anaphora.....	78
6.2.	Result comparison before/after optimizing result.	79
Java.....		79
6.3.	Results of Concept Ranking.....	81
7.	DISCUSSION	83
7.1.	Project Management	83
7.3.	Project Outcomes	84
8.	CONCLUSION & FUTURE WORK	85
8.1.	Conclusion	85
8.2.	Future Works	85
9.	SIGMAC EXTENSION-RECRUITMENT HELPER.....	86
9.1.	Introduction of SigmaC Extension of Recruitment Helper	86

9.2.	Objectives of the Recruitment Helper.....	86
9.3.	Literature Review of CVHelper.....	86
9.3.1.	Name Extraction	86
9.3.2.	Sentence Boundary detection.....	87
9.3.3.	Entity Extraction	88
9.4.	Recruitment Helper Design.....	89
9.4.1.	CV Extraction.....	89
9.5.	Email Extractor	89
9.6.	Education Extractor	90
9.7.	Professional Qualification Extractor	91
9.8.	Reference Extractor.....	91
9.9.	Technology Extractor.....	92
9.10.	Sample User Map after extracting Entities.	93
9.11.	Information retrieval through internet	94
9.12.	Future Works of Recruitment Helper.....	95
ABBREVIATIONS & ACRONYMS.....		96
REFERENCES		97

1. INTRODUCTION

Knowledge bases of enterprises consist of large document repositories and they are growing continuously. This large volume of documents covers vast amount of knowledge but the extraction of the relevant knowledge in a timely manner has become a challenging task. Therefore search engines and document summarization system has become a prominent research areas. Some of the above systems are developed based purely on statistical analysis while others employ natural language processing techniques.

SIGMAC is developed with the aim of solving above issues by utilizing more human understandable concept maps for knowledge representation. Usage of natural language processing techniques allows SIGMAC to overcome limitation of pure statistical analysis and SIGMAC also focus on utilizing text formatting information and sematic data sources to optimize the results obtained from syntactic analysis.

Outputs obtained from the system are useful in several different ways. Concept map representation of document repository allow concept level browsing of the knowledge base and exploring more interesting relationship between concepts. Concept map representation of individual documents effectively produces summaries of those documents allowing users to get an understanding of the document without going through the entire document. Furthermore the system can be used to generate ontological data source by providing a suitable set of documents for analysis.

1.1. Motivation and Applications

Ever growing knowledge bases of enterprises presents the demanding challenge of proper organization of information that would enable the fast retrieval of related and intended information. Failing to do so, would cause enterprises to confront challenge of survival in the industry than ever before, because of situations such as those in the following list.

- Inability to provide fast access to information
- Waste of labor for information retrieval
- Difficulty of identifying the relevant importance of concepts in the context of the enterprise
- Difficulty of integrating new members due to lack of organized learning resources.
- Analyzing the evolution of concepts with the evolution of the enterprise.

A business enterprise has gathered large reservoirs of valuable information during the evolution of their enterprise. But often it is very difficult to extract the information related to concepts and relationships among those concepts. Organizing that information in an easily accessible manner would enable enterprises to,

- identify the patterns of concepts that has varied over their evolution
- Serve as an access point to the knowledge base of the enterprise.
- Minimize the time and effort required for the discovery of relevant information related to a particular concept.
- Speedup the integration of new members of the enterprise by means of providing a well-organized learning resource.
- Identify the relative importance of concepts.

Widely used current approaches in enterprises are to search for information. This approach would not enable enterprises to fully leverage the true benefits of the above mentioned aspects. Using concept maps which is a widely accepted method of organizing information would enable enterprises to truly experience the value proper organization of information.

2. AIMS AND OBJECTIVES

2.1. Aim of SigmaC

Aim of the SigmaC is introducing a novel way to retrieve the written knowledge by creating concept map using accurate concepts and relationship extraction methods.

2.2. Objectives of the Project

- **Organizing enterprise knowledge base in convenient manner**

Internal personal in an enterprise uses this knowledge base documentations frequently and the knowledge base is becoming bigger and bigger. By this research project we are looking for a convenient way for browsing and extracting relevant information using the idea of concept maps.

- **Support all types of knowledge base documents**

Currently there are many document types that stores enterprise knowledge base. This project is expected to support common document types that gather the knowledge base of an enterprise.

- **Construct a word graph which would have relationships among words**

The suggested system would crawl through the documents in the organization and construct a word graph which would have relationships among words (is a, part of, etc.)

- **Some of the relationships can be weighted**

Weightings to determine the strength of the relationship

- **Editable and self-adjustable concept map**

The map should be editable (add, remove relationships) and the map should self-adjust with the edits. (Other edges also to be recomputed based on the edits).

- **Provide an intuitive UI**

Show the graph of words and a mechanism of browsing through the documents of the organization by clicking on the words in the graph. It would be a novel way of browsing the knowledge base of the organization.

3. LITERATURE REVIEW

3.1. Introduction

This chapter provides a detailed description of the literature review done for SIGMAC project. Furthermore candidate techniques, methodologies and tools identified to support for solving each of the sub problems that needs to be solved for concept map generation from unstructured text is addressed in this document.

These sub problems can be interrelated. These are only high level core problems. Subsequent section describing each of these sub problems would address further issues in each of these areas. Identified sub problems for SIGMAC project as follows.

- Natural Language Processing
- Concept Extraction
- Relationship Extraction
- Visual Representation and Browsing
- Document preprocessing

Document collection could contain different types of documents. Formatting methods used in each type could differ significantly. Therefore each of these documents needs to be converted to an intermediate textual representation. This is done at the Document preprocessing stage. Specific details about document preprocessing would be provided in the relevant section.

Some of the concept and relationship extraction techniques require natural language processing abilities. The level of language processing differs based on the technique, but higher level processing would require the methods used in the lower level processing. Therefore the section on Natural Language Processing would cover the all information about the NLP techniques used in other sections of this document.

Sections on concept extraction and relationship extraction discuss the available techniques and tools studied in the literature review. The section on Visual Representation and Browsing contains discussions about the candidate tools and methods studied during the literature review.

3.2. Natural Language Processing

3.2.1. Word Sense Disambiguation (WSD)

The task of Word Sense Disambiguation is to identify the correct sense of the word in context. Words can have different senses. Some words have multiple meanings. This is called as polysemy and sometimes two completely different words are spelled same. For example “Can” can be used as model verb (I can run fast) or as container (I delivered a can of coca cola). This is called Homonymy. The different between polysemy and Homonymy is always not clear. So the Word sense disambiguation is the problem of identity the correct sense of the word in a given sentence. LEXAS is a good algorithm that supports accuracy for word sense disambiguation.

3.2.1.1. LEXAS (Lexical Ambiguity Resolving System)

This approach integrates a diverse set of knowledge sources to disambiguate word sense, including part of speech (POS) of neighboring words, morphological forms, the unordered set of surrounding words, local collocations, and verb-object syntactic relation. They tested it in a common dataset including the noun “interest” used by Bruce and Wiebe who got the 78% of accuracy previously [1].

After LEXAS tested, they could get 87.7% accuracy that is higher than accuracy got by Bruce and Wiebe.

For example, let's take a sentence "The interest rates of fixed deposits are growing rapidly". Here the word "interest" is appears as a noun. So LEXAS only consider about the noun meanings of "interest" rather than verb meaning. There might be some morphological form of the selected word. For this example interest might be in forms of "interest", "interests". If we consider fall as an example it might be in forms of "fall", "fallen", "fell", "falls" and "falling". So LEXAS is doing this convention by converting each word in input sentence into its morphological root using morphological analyzer of WORDNET.

3.2.1.1.1. Algorithm

For performing WSD, LEXAS uses training corpus of sentences which are pre tagged with their correct senses[1]. LEXAS builds one exemplar based classifier for each content word w . It operates in two phases. Those are training phase and test phase. In the training phase, LEXAS is given a set S of sentences in the training corpus in which sense-tagged occurrences of w appear. For each training sentence with an occurrence of w , LEXAS extracts the parts of speech (POS) of words surrounding w , the morphological form of w , the words that frequently co-occur with w in the same sentence, and the local collocations containing w . For disambiguating a noun w , the verb which takes the current noun was the object is also identified. This set of values form the features of an example, with one training sentence contributing one training example.

In the test phase LEXAS is given unseen sentences. For a new sentence containing w , LEXAS extracts the new set of features value for new sentence including pats of speech of words surrounding w , the morphological forms of w , the frequently core occurring words surrounding w , the local collocations containing w , and the verb that takes w as an object. These values forms features for the test example. Then compare every test example with the every training example and select the closest match one. To find the closest one firstly what they do is to identify the features of the sentence.

Let $cp(i|k)$ denotes the conditional probability of sense i of w given word k , where

$$cp(i|k) = \frac{N_{i,k}}{N_k}$$

$N_{i,k}$ is the number of sentences in which keyword k co-concurs with w .

N_k is the number of sentences in which keyword k co-concurs with w where w has sense i .

After calculating the conditional probability, LEXAS check whether the keyword k is a feature or not. To select the keyword k as a feature it should satisfy following three conditions.

1. $cp(i|k)$ has to be greater than $M1$ where $M1$ is predefined value.

2. Word k has to occur at least $M2$ times with the word w .
3. For the word w , only a $M3$ number of keywords are selected. This is done by ordering the keywords by the frequency of co-occurrence with the given word w in the sense i .

So the next thing is to find the distance between training and test example. LEXAS uses following definition of distance two symbolic values v_1 and v_2 of a feature f .

$$d(v_1, v_2) = \sum_{i=1}^n \left(\frac{C_{1,i}}{C_1} - \frac{C_{2,i}}{C_2} \right)$$

- $C_{1,i}$ – Number of training examples with value v_1 for feature f for sense i in the training corpus
- C_1 – Number of training examples with value v_1 for feature f for any sense in training corpus
- $C_{2,i}$ – Number of training examples with value v_2 for feature f for sense i in the training corpus
- C_2 – Number of training examples with value v_2 for feature f for any sense in training corpus
- n – Total number of sense in word w

3.2.2. WordNet

WordNet [1] is a large, well known lexical database for English Language developed by Cognitive Science Laboratory of the Princeton University, United States. Words of the English language such as Nouns, Verbs, adjectives, adverbs are grouped in to sets called synsets. Currently there are about 117,000 synsets available in WordNet. Each synset express distinct concept. Thoses synsets are linked various semantic relations. WordNet's is developed in order to use it as a tool for computational linguistics and natural language processing. This can be used as a dictionary or a thesaurus. WordNet Interlinks specific senses of words and it labels semantic relationships. Following are those types of sementic relationships.

Synonymy is a lexical relation between word forms. It is about similarity of meanings. It is convenient to assume that the relation is symmetric: if x is similar to y , then y is equally similar to x .

Definition -two expressions are synonymous in a linguistic context C if the substitution of one for the other in C does not alter the truth value.

WordNet contains eight different senses of the highly polysemous noun “case”

1. {carton, case0, box, @ (a box made of cardboard; opens by flaps on the top)}
2. {case1, bag, @ (a portable bag for carrying small objects)}
3. {case2, pillowcase, pillowslip, slip2, bed linen, @ (a removable and washable cover for a pillow)}
4. {bag1, case3, grip, suitcase, traveling bag ,@ (a portable rectangular traveling bag for carrying clothes)}

5. {cabinet, case4, console, cupboard ,@ (a cupboard with doors and shelves)}
6. {case5, container ,@ (a small portable metal container)}
7. {shell, shell plating, case6, casing1, outside surface,@ (the outer covering or housing of something)}
8. {casing, case7, framework,@ (the enclosing frame around a door or window opening)}

Antonymy is kind of opposite relation to the synonymy. Extact definition is not there in any literature. The antonym of a word x is sometimes not- x , but not always. As an example consider {rise, ascend} and {fall, decent}. Here {rise, fall} are consider as antonym but {rise, decent} are not considered as antonym.

Hyponymy is semantic relation between word meanings.e.g., {maple} is a hyponym of {tree}, and {tree} is a hyponym of {plant}.

Meronymy is a semantic relation—is the part-whole (or HAS A) relation, known to lexical semanticists as meronymy / holonymy. A Concept represented by the synset $\{x, x', \dots\}$ is a meronym of a concept represented by the synset $\{y, y', \dots\}$ if native speakers of English accept sentences constructed from such frames as A y has an x (as a part) or An x is a part of y . These relations are transitive and asymmetrical.

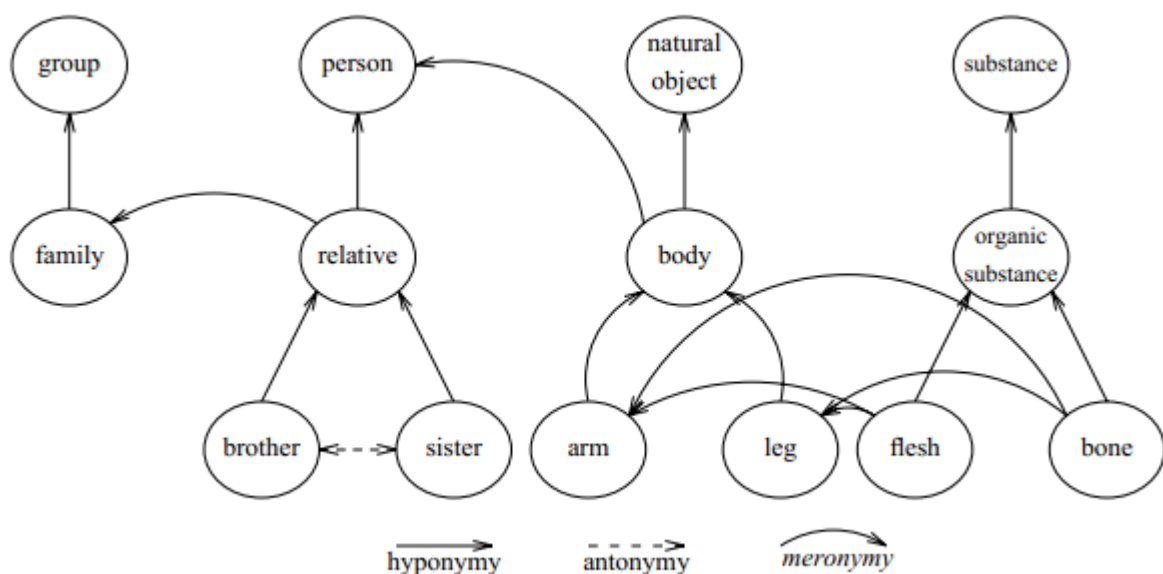


Figure 1 Network representation of three semantic relations among an illustrative variety of lexical concepts

Morphological Relations is a class of lexical relations between word forms. A word can express as varies morphological forms. As a example the verb ‘fall’ can be mentioned in a text as fall, fell, fallen etc. WordNet facilitate all these morphological forms by using morphological relations.

Adjectives

In WordNet there are two classes of adjectives: descriptive and relational. Descriptive adjectives describe the nouns and typically bipolar. Relational adjectives are assumed to be stylistic variants of modifying nouns.

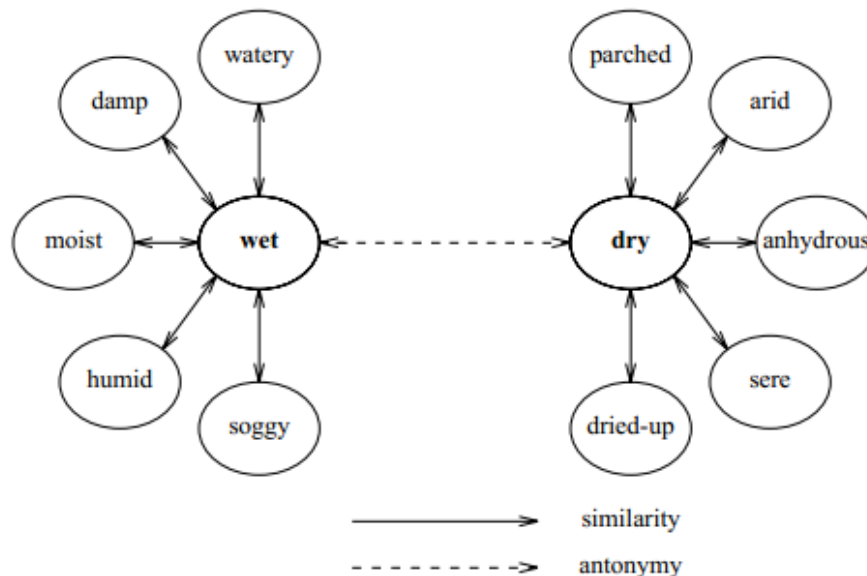


Figure 2 Bipolar Adjective Structure

3.2.3. Anaphora resolution

Anaphora resolution is a complicated problem in Natural Language Processing (NLP). There are various definitions in anaphora and one of that is based on the notion of cohesion. That means Anaphora is cohesion which point back to some previous item. The reference is called an anaphor and the entity which it refers is its antecedent. The process of determining antecedent of an anaphor is called Anaphora Resolution.

Anaphora resolution can be described as resolving what a pronoun, or a noun phrase refers to. This is also called as co-reference resolution. This can be clearly explained using an example.

Java is a programming language. It is a general-purpose, concurrent, class-based, object-oriented language.

Here human brain can identify that 'It' means 'Java' but in computations there should be an algorithm to identify them. In that case relevant words are tagged for resolve the anaphora.

It => Anaphor

Java (N) => Antecedent

If there is a noun phrase (NP) instead of noun whole noun phrase is the antecedent. (Not only noun part)

Co-reference => Both 'Java' and 'It' refer to the same real world entity.

So now anaphora resolution can be explained as determining the antecedent of an anaphor. Co-references can exist as a chain when more than one of preceding entities have the same referent. In co-reference resolution it has to do with all co-reference chains. There are three major types of anaphora.

- **Pronominal anaphora** – Mostly used type of anaphora. This can be resolved by using anaphoric pronoun. (E.g. Above mentioned example)
- **Definite noun phrase anaphora** – Here antecedent refer by a noun phrase of same or close concept. (E.g. **Java is a programming language. The language is a general-purpose, concurrent, class-based, object-oriented language**)
- **One-anaphora** – Anaphoric expression identified using noun phrase ‘one’. (E.g. The small *ones* went in the side pocket.)

Also anaphora can be categorized in following way.

- **Intra-sentential anaphora** – Both anaphor and antecedent which referenced anaphor are in the same sentence.
- **Inter-sentential anaphora** – Referred antecedent which is in a different sentence. Anaphor and antecedent are in two different sentences.

3.2.3.1. Anaphora Resolution Frameworks

3.2.3.1.1. JavaRAP

Resolving anaphora in JavaRAP consist of four steps. They are,

1. It uses parser to generate a parse tree using text with sentence delimitation.
2. Use the parse tree to extract two lists. They are a noun phrases list with all noun phrases and a list of third person pronouns and reflexive pronouns.
3. Then identify the anaphor by considering each item in a subset of the noun phrases (But currently JavaRAP consider only within three sentences from where the anaphor exist). Then antecedent-anaphor pairs are identified by using the anaphor binding algorithm or the syntactic filter, according to the type of anaphor (lexical or a third person pronoun).
4. Antecedents which can't pairs with anaphor by using above step are ranked by their salience weights (Way to measure closeness between antecedent and anaphor) and the antecedent with highest weight is selected as the actual antecedent. Here the antecedent closer to the anaphor is given higher priority.

3.2.3.1.2. Reconcile

Reconcile used two famous criteria which used to define set of noun phrases in co-reference relation. They are Message Understanding Conference (MUC) and Automatic Content Extraction (ACE). The name Co-reference Element (CE) stands for any element in co-reference relation. Reconcile consists of both MUC and ACE specification for resolving co-reference. MUC and ACE define noun phrases as nouns, pronouns and noun phrase. Other than that MUC define: Nested named entities (e.g. “Ceylon” in “Bank of Ceylon”), Relative pronouns, Gerunds, Nested nouns (e.g. “union” in “union members”). ACE define,

- Relative pronouns and gerunds are not co-reference elements (excludes all nested nouns)
- Co-reference elements shouldn't belong to one of seven semantic classes. (Person, organization, geo-political entity, location, facility, vehicle, and weapon)

```
procedure ExtractCorefElements(D, NPs, NEs){
  A text document D
  NPs := all noun phrases in D.
  NEs := the set of named entities in D.

  /* Extract base NPs -- those that do not contain embedded clauses
  and prepositions and are not nested */
  initialize bNPs := NPs
  foreach noun phrase np ∈ bNPs do
    if np contains a preposition or embedded clause:
      Remove np from bNPs
    if np is the child of another np in bNPs:
      Remove np from bNPs

  /* Consolidate NPs and NEs */
  foreach noun phrase np ∈ bNPs do
    oNEs is the set of those NEs that overlap np in span
    foreach named entity ne ∈ oNEs do
      if neither np covers ne nor ne covers np:
        Expand np's bytespan so that it subsumes ne.
        Remove ne from NEs
      else if ne is a proper substring of np and the two have the same head:
        remove ne to NEs

  initialize resultNPs := bNPs
  /* Add in named entities that are not part of the NP set. */
  Add ne to resultNPs

  /* Add in nested noun phrases that are not already in the set. */
  foreach noun phrase np ∈ resultNPs do
    if np contains a nested NP from NPs -- nnp ∉ resultNPs
    and resultNPs does not contain a NP with the same head:
      Add nnp to NPs
  do the same for nested nouns (non-NPs) when applicable
  return NPs
}
```

Figure 4 Co-reference generating algorithm

3.2.4.Sentence Boundary Detection

Sentence boundary detection is of high importance higher levels of linguistic processing methods such as POS tagging and chunking. Simplest method of identifying the sentence boundary would be to split a text body in to sentences using the period (“.”). Although this strategy works in many situations it cannot be taken as an accurate measure due to the ambiguity of the period in a sentence. For example a period may occur in an abbreviation, end of a sentence or in mentioning the names with initials. Different methods for identifying sentence boundaries are suggested by researchers. The approach suggested by David D. Palmer and Marti A. on their paper on Adaptive Sentence Boundary Disambiguation[2] is described below.

Stream of text is divided in to tokens. A lexicon containing the frequency of each word occurring in possible part of speech tags are maintained. POS tags are further sub divided in to 18 generic categories. Category frequency of a word is taken as the sum of frequencies of the word in POS tags which was mapped in to the given category. Then the category frequencies are converted to probabilities by dividing the category frequency by the frequency of occurrence of the word. An array containing the category frequencies and two flags to indicating whether word begins with a capital letter, word is followed by a punctuation mark, is created for each word. Then for ‘k’ number of words surrounding an instance of end of sentence, their descriptor arrays are fed to a neural network. Input layer is connected to a hidden layer with j units which are activated by sigmoidal squashing function. The network output a value between 0 and 1. Disambiguation is done based on two threshold values. If the value returned by the network is above the larger threshold value, the input instance is identified as a sentence boundary, if the value is below the lower threshold value; the instance is not a sentence boundary. Value between two thresholds indicate ambiguous input instance.

3.2.5.Collocation Identification

Collocations represent multi word expression which are conventional phrases used in communication. Three different categories were proposed by Frank Smadja[3]. Those are mentioned below.

- Predicative Relations
- Rigid Noun Phrases
- Phrasal Templates

Concepts does not necessarily be single words, they might be constructed using multiple words. Therefore identification of these collocations is an important part in concept extraction. The most important category of collocations in concept extraction is Rigid Noun Phrases. These are defined as “uninterrupted sequences of words”. Examples of these are “Database Management Systems”, “Stock Market”, “Object Oriented Programming”. Collocations of this category identification can be done using POS tag signatures. Pure statistical analysis methods for identifying collocations of all three categories were studied in the literature review. Different methods used in collocation identification are also included in the Concept Extraction section of the document.

3.2.6.Parts of Speech Tagging

POS tagging is the process of labeling words in a sentence with appropriate POS tags. These POS tags are possible syntactic categories that a word can belong to in its use in a particular sentence. This POS tagged sentences can be used in further level of parsing to generate phrase structure trees for sentences and also in concept identification.

Different methods for POS tagging are being used in various POS taggers. Most popular method of POS tagging is developed using the Markov model. Tagging process behaves according to the following two conditions. These are essentially Markov properties that are required to model a process as a Markov process.

- POS tag of a word does not depend on the previous POS tag of the previous word (Markov Property).
- The probabilities are time invariant (probability does not change as the sentence is processed).

The maximum likelihood of getting a tag A after a particular tag B calculated using relative frequency of different tags followed by particular tag.

$$P\left(\frac{A}{B}\right) = \frac{\text{frequency of occurrence of A after B}}{\text{frequency of occurrence of B}}$$

Set of states in the model corresponds to the set of tags. Probability of getting a particular tag for a given word can directly be computed using the training corpus. That is, the probability of emission of a word from a particular state at a certain position of a word sequence. Let W be the emitted word from state (tag) T. then,

$$P\left(\frac{W}{T}\right) = \frac{\text{Number of occurrences of W tagged by T}}{\text{number of occurrence of T in that position}}$$

Then the most suitable tag sequence for given set of word sequence in a sentence can be calculated as follows.

Let $W_{1,n}$ be the word sequence and let $T_{1,n}$ be a possible tag sequence. From the Bayes theorem,

$$P(T_{1,n}|W_{1,n}) = \frac{P(W_{1,n}|T_{1,n}) * P(T_{1,n})}{P(W_{1,n})}$$

From Bayesian statistics we can ignore $P(W_{1,n})$ in finding the maximum of $P(T_{1,n}|W_{1,n})$.

Then

$$\arg_{t_{1,n}} \max(P(T_{1,n}|W_{1,n})) = \arg_{t_{1,n}} \max(P(W_{1,n}|T_{1,n}) * P(T_{1,n}))$$

This can be reduced to

$$P(W_{1,n}|T_{1,n}) * P(T_{1,n}) = \prod_{i=1}^n [P(W_i|T_i) * P(T_i|T_{i-1})]$$

Assumptions of words being independent are made in above calculations. Therefore the optimal tag sequence for a sentence can be determined as follows

Then

$$\arg_{t_{1,n}} \max \left(P(W_{1,n} | T_{1,n}) * P(T_{1,n}) \right) = \prod_1^n [P(W_i | T_i) * P(T_i | T_{i-1})]$$

But this method only can identify the tags for words that are in the training corpus. Different methods for tagging unfound words in the training corpus have also been developed by researchers. Stanford POS tagger[4] is one of the widely used tagger.

3.3. Concept Extraction

Concept extraction from documents or individual sentences is a heavily researched area. Several methodologies have been developed by researchers to achieve this task. Each of these methods has their advantages and drawbacks. Some of the techniques depend solely on statistical measures while other techniques utilize semantic information of words from semantic databases to identify concepts. Techniques based on the syntactic structure of the sentences and hybrid techniques which involve syntactic analysis of sentences, use of semantic information and applying statistical information has also been developed. Following section of the document describes the identified methods for concept extraction during the literature review.

3.3.1. Concept Identification using POS tag signatures

This technique uses POS tagged sentences as its input and tries to identify candidate concepts that can be used to represent document. The main use of this technique is to identify fixed collocations, but this can be employed to identify concepts as well. Candidate concept extraction is done based on the heuristic that concepts in a sentence should follow particular set of POS tag signatures. When the document is given, first step is to parse the sentences using a POS tagger. Then the POS tagged sentences are inspected to check predefined set of POS tag patterns. Pattern matching is based on greedy approach where largest match is taken in to consideration. If matches are found those words or fixed collocations are taken as candidate concepts. After identifying candidate concepts frequency of occurrence of candidate concepts in the document are taken filter out less frequent concepts and the remaining candidate concepts which have higher frequency is taken to represent the document.

Various set of POS tag patterns has been proposed by different researchers. Some of the example patterns suggested are listed below.

POS tag patterns suggested by Paul Buitelaar, Thomas Eigner in their research about topic extraction from scientific documents[5] are as follows, other patterns can be found in the VioletaSeretans[6] paper of usage of syntactic patterns for collocation identification.

- ***JJ NN – semantic tagging***
- ***JJ NNS - anaphoric expressions***
- ***NN NN - ontology adaptation***
- ***JJ NN NNS - modelling similarity measures***
- ***NN JJ JJ NN NN - domain specific semantic lexicon construction***

Set of patterns identified by Justeson and Katz are listed below. These patterns were taken from Foundations of Statistical Natural Language Processing[7] book.

- *A N - linear function*
- *NN - regression coefficients*
- *A A N - Gaussian random variable*
- *A N N - cumulative distribution function*
- *N A N - mean squared error*
- *N N N - class probability function*
- *N P N - degrees of freedom*

Let's take the sentence "java applications can run on any operating system" as an example to illustrate the process. POS tagged sentence using Stanford POS tagger would look like the following.

java/NN applications/NNS can/MD run/VB on/IN any/DT operating/NN system/NN ./.

Concepts mentioned in the sentence are "java applications" which has the tag pattern NN NNS and "Operating System" which has the tag pattern NN NN. First set of patterns mentioned above would only identify the "Operating System" as the concept but the second pattern set would identify both "java applications" and "Operating System" as concepts.

There are some drawbacks in this approach. Set of concepts identified can contain some words or collocations which have the same signature but are not concepts described in the document. Accuracy of this improved by the frequency filtering but still it would contain some of the unexpected concepts depending on the source of the text corpora. Furthermore this method does not consider about different senses for words. Similar concepts represented by words would get counted for two different concepts thus resulting in lower frequency of occurrence. This is a common problem for all the methods that are not supported by semantic information about words.

3.3.2.Frequency based Key word extraction

In this model terms with highest frequency of occurrence is taken as candidate keywords to represent the document[7]. This model improve its result by filtering out terms which are common in all documents such as the, a, an etc. This list is called the stop word list. In order to identify multi term keywords, technique uses n-gram models. For key words with two terms, co-occurrence of two words is taken as a measure of relationship between words. Set of bigrams with high frequencies are taken as candidate keywords.

This technique suffers from the fact that extracted key words might not necessarily represent concept discussed in the document. It can include multi terms such as "he said", "last year", in its results. POS tag pattern based filtering method can provide better results than this technique. Concept identification without considering semantic information can also affect the results in cases where synonyms and homonyms of words play a major role in document.

3.3.2.1. Usage of Phrase structure for concept extraction

Phrase structure trees of sentences in a document are used to identify concepts[8]. These phrase structure trees are generated using syntactic parsing of sentences using language grammar parsers. Sentences are given a tree structure so that noun phrases, verb phrases etc. of sentence can be

traversed using tree traversing algorithms or patterns in the tree can be matched using applying pattern matching on the tree itself.

First, sentences in the document are parsed using language grammar parse and then noun phrases of sentences are taken as candidate concepts. Then the selected set of concepts is filtered out using frequency filtering where candidate concepts with heights frequency of occurrence are retained to represent document. The idea behind this method is that, concept are likely to be noun phrases in a sentence rather than verb phrases. Here the sub trees rooted under verb phrase nodes could also contain noun phrases. These phrases are also included in finding candidate concepts to represent the given document.

Let's take the sentence "Java is a programming language" as an example. The phrase structure tree generated by the Stanford Parser would look like that of the following figure.

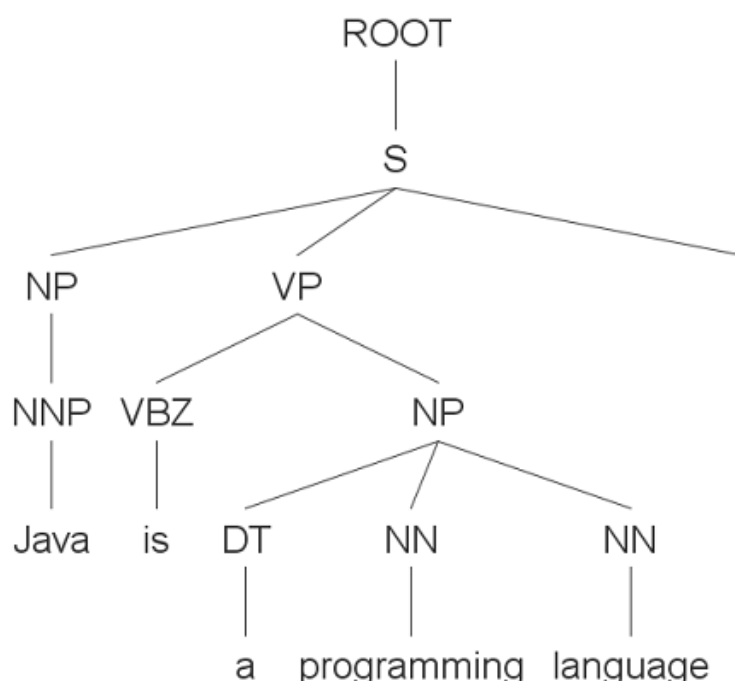


Figure 5 Structure tree generated by Stanford parser

Here the noun phrases of the sentence are denoted by NP. Two noun phrases for this sentence are "Java" and "a programming language". As it can be seen in the second noun phrase unwanted determiner is attached to the noun phrase. It can be removed using the strategy described below.

Extension to this method is to use further filtering of candidate concepts using POS tag patterns. This will allow us to filter out unwanted components from the noun phrases.

Semantically blind frequency analysis is another major problem in this method as in other syntax based or frequency based methods. Phrase structure itself can help in identifying difference in meaning therefore considering positioning of noun phrases in the phrase structure is suggested as a possible improvement. But how this can be employed is still under study in the research community.

3.3.3. Dependency Relationship based concept extraction

Further processing of phrase structure trees of sentences are done to generate dependency relationships between words. This parsing provide detailed information such as direct object, subject etc. First the sentences in the document are processed to extract dependency relationships. Then the subjects and objects of a particular verb are taken as candidate concepts. Further frequency filtering is used to filter out low frequency concepts.

Following shows the dependency relationships identified by RelEx dependency relationship extractor for the sentence “java is a programming language.”

```
_obj(be,language)  
_subj(be,java)  
tense(be,present)  
inflection-TAG(be,.v)  
pos(be,verb)  
pos(.,punctuation)  
_nn(language,programming)  
inflection-TAG(language,s)  
pos(language,noun)  
noun_number(language,singular)  
inflection-TAG(programming,n-u)  
pos(programming,noun)  
noun_number(programming,uncountable)  
inflection-TAG(java,.n)  
pos(java,noun)  
noun_number(java,uncountable)  
pos(a, det)
```

This technique would extract “language” and “java” as concepts from the sentence. Details of the dependency relationships are discussed under the section describing the RelEx dependency relationship extractor.

One drawback of this method is that, the concept identification is dependent on the dependency relationship generation which is still not fully capable of identifying multi term concepts. Therefore further level is parsing is used to identify multi term concepts instead of just extracting subjects and objects in dependency relationships. Accurate selections of dependency relationships are required for concept identification. Inappropriate selection of relationships to be used would result in exclusion of important concepts or inclusion of unwanted concepts to represent document.

3.3.4. Semantic data source based concept extraction

This method requires access to a database with semantic information of words. This is one of the methods utilized by Boris Gelfand in Automatic Concept Extraction from plain text[9]. Synonyms and homonyms are taken in to account when identifying concepts. Synonym and homonym identification is done using the contextual information in occurrence of words. That is, the set of words in the sentence where candidate word occurs are taken in to account to identify synonyms and homonyms. Concept relationships are also derived from semantic source using concept hierarchies

available in the semantic database. Frequency of occurrence is used to identify key concepts to represent the document. This method gives more accurate set of concepts to represent document.

The accuracy of this technique is heavily dependent on the semantic data source. Comprehensive semantic data sources of all possible words are not feasible or they are impossible. Another problem with this technique is that the words can have different senses depending on the domain and the position in which the word appears in the sentence. Covering all this base would be a complex task. Therefore this can lead to abandoning concepts which are not present in the semantic database. A domain dependent semantic data source can be helpful in concept extraction from documents of a particular domain. Expertise of the domain knowledge and linguistic knowledge is required to create such a database. Maintenance of such information need to be done in order to support the evolution of phrase used in the selected domain.

3.3.5. Usage of formatting information to support concept identification

Most of the documents are not in plain text format. Therefore those documents contain some formatting information about the content of the document. This formatting information such as font size, bold text, can be used to support concept identification process. These support information can be extracted in the preprocessing stage of documents prior to linguistic analysis of sentences. Topics in each segment of the document can be identified using font-sizes. The heuristic behind this process is that the topic of a segment would have larger font-size. Identified topical segment does not necessarily represent concepts. Therefore this information would only be used for supporting concept extraction process. Accurate linguistic processing to improve the information in topical segments is also not possible because, most of the topical segments do not form grammatically correct sentences.

The major difficulty of this method arises due to the need of document parsers for each document type that need to be analyzed. Each document type would need to be handled independently since there is no universal parsing mechanism to identify the formatting information in each of the document types.

3.3.6. K-gram based method

In this method k-gram is defined as an ordered sequence of k words occurring one after the other in text. For example “Sanath Jayasuriya is a good cricketer” has three grams: “Sanath Jayasuriya is”, “Jayasuriya is a”, “is a good”, “a good cricketer” and has four grams: “Sanath Jayasuriya is a”, “Jayasuriya is a good”, “is a good cricketer”. In this method it is assumed that all the Wikipedia titles are concepts.

For a given dataset C to find the concepts what the preprocessing step does is to extract all the k-grams with their frequencies. Since very few concepts have more than four words this algorithm considers only the $k \leq 4$ grams[3].

Claim 1

If a given k-gram $a_1a_2 \dots a_k$ ($k > 2$) is a concept, then it is not true that both of the following k-1-grams are concepts: $a_1a_2 \dots a_{k-1}$ and $a_2a_3 \dots a_k$. If a 2-gram a_1a_2 is a concept, then at least one of a_1 and a_2 are concepts.

k	Total	Both k-1-grams	%age	\geq One k-1-gram	%age
2	2245301	1250613	55.69	2147305	95.63
3	1357852	105515	7.77	688324	50.69
4	681447	12150	1.78	202763	29.75
5	350481	1822	0.51	64630	18.44
6	166827	525	0.31	22075	13.23

Table 1 -Claim Justification

This claim has been justified from analyzing Wikipedia articles[3].

From analyzing the justification table we can see that the claim is not 100% accurate. So there is a possibility to miss some concepts, but the possibility is too small. At the first extract the k-grams which can be considered as concepts. Rest of the k-grams can be discarded. To find the k-grams which are to be considered as concepts, this method use pre-defined set of k-grams that are concepts. From the above claim it can be discard either k-gram or some k-1 gram that the k gram contains. In order to extract concepts from above filtered words, following indicators are used.

- **First Indicator**

In a concept frequency of occurrence is high. Frequency of occurrence of concept a is also called as support S_a .

- **Second Indicator**

What we select as the concept should be better than the sub and super concepts. In this algorithm it has been defined three threshold values to satisfy second indicator.

If $a=t_1t_2t_3\dots t_k$, then $b=t_1t_2\dots t_{k-1}$ is the prefix k-1 gram and $c=t_2t_3\dots t_k$ is the suffix k-1 gram. Following confidence level should be exceed the threshold values for satisfy the second indicator.

- Pre-conf : $C_{1a} = \frac{S_a}{S_b}$
- Post-conf : $C_{2a} = \frac{S_a}{S_c}$
- Rel – Conf(a) = $\frac{\min\text{-conf}(a)}{\max(\max\text{-conf}(b), \min\text{-conf}(c))}$

- **Third Indicator**

This indicates that only portion of sentences that convey a single meaning or idea.

3.3.6.1. K-gram Algorithms

Algorithm 1 takes k-grams with annotated frequencies and proceeds to extract concepts in bottom up manner. Algorithm 1 calls to algorithm 2 to check whether the given k-gram satisfies the above indicators 1 and 2.

To satisfy the second indicator, it should be exceeded some threshold values. Those are listed below

Parameter	k=1	k=2	k=3	k=4
Support-threshold	100	35	60	60
Min-conf threshold	-	0.035	0.1	0.15
Rel-conf threshold	-	-	0.2	0.25
Pre-conf threshold	-	0.1	0.15	0.2

for each k-gram[3].

Table 2 Threshold parameter values

Algorithm 1 *conceptExtraction*: Extract k -grams from a set of k -grams and their frequencies

Require: $l \leftarrow$ set of k -grams with frequencies S_a

```

1:  $C \leftarrow \emptyset$  {Set of Candidate Concepts}
2: for all  $k = 1$  to  $n$  do
3:    $D \leftarrow \emptyset$  {Set of Sub-concepts to discard}
4:   for all  $a \in l$ ,  $a$  is a  $k$ -gram do
5:      $(D, Answer) = \text{candidateConceptCheck}(a, C, D)$ 
6:     if  $Answer == \text{true}$  then
7:        $C \leftarrow C \cup \{a\}$ 
8:     end if
9:   end for
10:   $C \leftarrow C - D$ 
11: end for
12:  $F \leftarrow \emptyset$  {Set of Concepts}
13: for all  $a \in C$  do
14:   if  $\text{containsStopWords}(a) == \text{false}$  then
15:      $F \leftarrow F \cup \{a\}$ 
16:   end if
17: end for
18: return  $F$ 

```

Algorithm 2 *candidateConceptCheck*: Check if a k -gram is a candidate concept

Require: $a \leftarrow k\text{-gram}$
Require: $C \leftarrow \text{set of concepts}$
Require: $D \leftarrow \text{discard set}$
1: $k \leftarrow k\text{-gram-size}(a)$;
2: $b \leftarrow k\text{-gram-prefix}(a)$
3: $c \leftarrow k\text{-gram-suffix}(a)$
4: **if** $[S_a > \text{support-threshold}(k)]$ **then**
5: **if** $[k = 1]$ **then**
6: **return** (D, true)
7: **end if**
8: **if** $[\text{min-conf}(a) > \text{min-conf-threshold}(k) \wedge$
 $\text{rel-conf}(a) > \text{rel-conf-threshold}(k)]$ **then**
9: **if** $[k > 2]$ **then**
10: $D \leftarrow D \cup \{b\}; D \leftarrow D \cup \{c\}$
11: **end if**
12: **return** (D, true)
13: **end if**
14: **if** $[b \in C \wedge$
 $\text{post-conf}(a) > \text{dominance-thres} \times \text{pre-conf}(a) \wedge$
 $\text{post-conf}(a) > \text{post-conf-threshold}(k)]$ **then**
15: $D \leftarrow D \cup \{c\}$
16: **return** (D, true)
17: **end if**
18: **if** $[c \in C \wedge$
 $\text{pre-conf}(a) > \text{dominance-thres} \times \text{post-conf}(a) \wedge$
 $\text{pre-conf}(a) > \text{pre-conf-threshold}(k)]$ **then**
19: $D \leftarrow D \cup \{b\}$
20: **return** (D, true)
21: **end if**
22: **end if**
23: **return** (D, false)

3.4. Relationship Extraction

Huge amount of usable electronic data is in the form of unstructured text which cannot be easily understood by machines. Relationship extraction is a task that recodes those unstructured information found on those text, into well-structured format. In our project the domain is computer science and related technologies. Adapting conventional relation extracting frameworks requires significant effort. Here we analyze the current approaches to achieve those goals.

Here we are going to discuss about the relationship among the concepts which extracted from technical documents. In our project requirements the relationships should not only be hierarchical it can and it will not be a tree, more like a linked map of nodes. This map will not be radial hierarchy or tree structure as in mind maps. Nodes are the concepts and the edges linking those nodes will be the relationships. These relationships can be classified and quantified. Edges should represent how close those concepts (proximity).

3.4.1. Common Relationship Extraction Approaches

- Supervised

This includes rule engineering and supervised machine learning techniques. For a new domain, system needs to be built from a scratch.

- Bootstrapping

Using initial seed data system can be build. This Approach is less costly compared to the supervised method in terms of effort and less complex.

- Generic Approaches

It can be easily adapted to a new domain using generic approaches because of the parameters do not need to be modified for new domains or tasks.

From above supervised and bootstrapping approaches are mainly focused on instantiation of ontology while generic approaches address problem of learning relation schemas from data.

Relationship Extraction can be divided in to two main tasks. First one is relationship identification (also known as relation mining) and other one is relationship characterization (also known as relation discovery).

3.4.1.1. The Information Extraction Framework

Goal of the information extraction framework is extracting structured information from text written in natural language. Information extraction framework consists of solutions for different sub problems. Core tasks that can be identified are named entity recognition (NER), relationship extraction (RE) and event extraction (EE).

3.4.1.2. Relation Identification and Characterization

Here the goal is to identify mentions of relations from the text. Following is the definition for relationship mention. [10]

“A relation mention is a predicate ranging over two arguments, where an argument represents concepts, objects or people in the real world and the relation predicate describes the type of stative association or interaction that holds between the things represented by the arguments “

Occurrences of textual references to a relation are referred to here as relation mentions. Likewise, instances of textual references to an entity are referred to as entity mentions. Finally, labels used to describe the class of relations and entities are referred to as relation types and entity types.

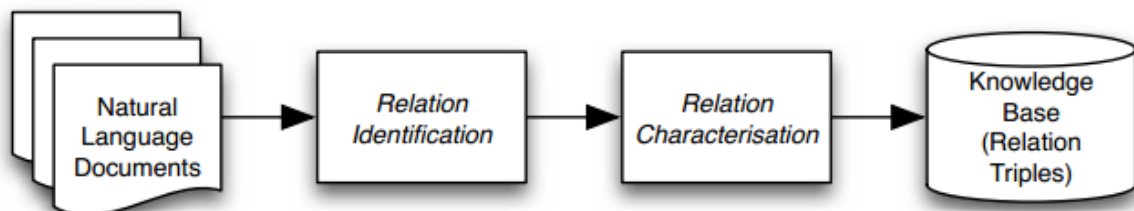


Figure 6 Overview of main relation extraction sub-tasks.

Figure 6 represent pipeline of two main sub tasks of relation identification and relation Characterization. Input is documents containing unstructured text written in natural language. These documents are fed into a relation identification system which identifies pairs of relation forming entity (or concept) mentions. Then the relation characterization system annotates the entity mention pairs with a label which is describing the type of the relationship.

3.4.1.3. Relation Extraction and Adaptation

3.4.1.3.1. Supervised Approaches

Traditional approaches to relation extraction are based on either rule engineering or supervised machine learning. Rule engineering needs an expert in targeted domain in order to develop extraction grammar. A rule should be there to extract each relation on the text. In supervised learning previously tagged or annotated corpus need to fed to the system in order to train the new system. Both concepts or entities and relations mention should have been marked by a human annotator. Most of the time supervised machine learning systems extracts features of entity mention together with surrounding context of a sentence.

3.4.1.3.2. Bootstrapping Approaches

Bootstrapping approaches can be further categories in to three types. In the first type it is required to have pre-tagged training data like in supervised machine approaches, but uses relatively small amount. Then the widely coverage system will do the relation extraction though the iterative process of learning and automatic annotating new data. In the second type it is required to have small amount of entity pairs for each type of specific relation. Then the system will bootstrapped in a iterative process.

1. Identify the sentence segments where the training entities occur together and using those to identify patterns for relationship identification task.
2. Using patterns identified by step 1 to identify new pairs which the intended relation exists.

Third type requires set of documents tagged as relevant or non-relevant to a particular relation extraction task. The difference among relevant and non-relevant is used to assign a weight that balances pattern frequency in relevant documents versus frequency across relevant and non-relevant documents.

3.4.1.3.3. Generic Approaches

Generic approaches are different from above two approaches because no need to have annotated data or parameter tuning for relation extraction in new domain. Generic approaches can be developed with reference to one domain and transfer other domain without losing the accuracy significantly. They can be used for learning the structure of ontologies and learning how to instantiate them. Conrad and Utt (1994)[11] describe an approach to identifying associated pairs of named entities from a large corpus using statistical measures of co-occurrence. Hasegawa et al. (2004)[12] describe an approach to characterizing co-occurring named entity pairs by relation type.

3.4.2. Ontologies

Ontologies are the formal representations of major terms and concepts of domains with their relations. These Ontologies are heavily used in bio medical sub domains with natural language processing (bio-NLP). This should have been an easy task if we have complete open centralized ontology for computer science. For Bio-Medical domain currently there are several ontologies developed for subdomains. (Ex: <http://obofoundry.org/>).

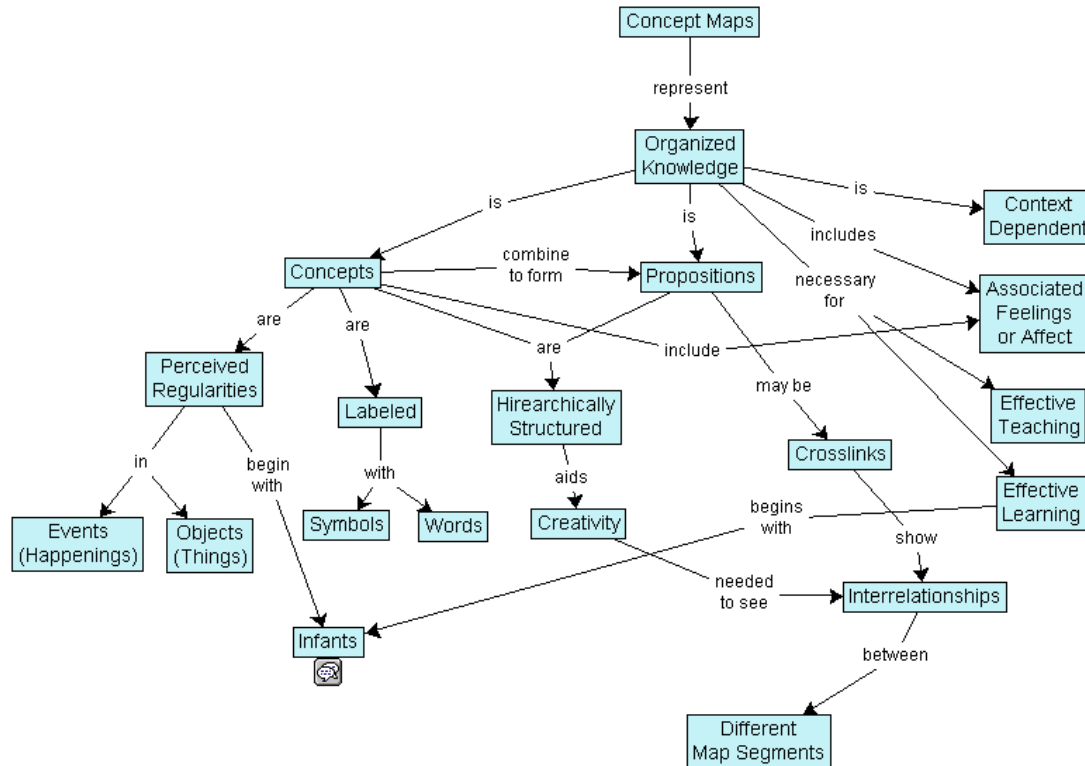


Figure 7 Relationship Representation on a concept map

3.4.3. Pattern-based approach to relation extraction

3.4.3.1. Relations of Interest

Many researchers are mostly interested in relations such as hypernymy (or is-a), meronymy (or part-whole), causality (relationship between the cause and the effect) etc. Other than those common relations, there, researchers have studied identifying many relationships such as located in, book-authored-by, employee-organization, film-director etc.

3.4.3.2. Patterns

Once a relationship was identified, researchers investigate the linguistic patterns that can express that relationship. Then the specific knowledge patterns are used to identify those relationships.

3.4.3.3. Discovery and Pattern Expression

Traditionally they used machine readable dictionaries (MRD) to acquire semantic relations. And Domain specific semantic relationships also can be acquired by using those MRDs. This was convenient when there were small size digital textual data available. But now with the availability of very large textual repositories, data mining techniques and algorithms are used for this task.

Also the internet can be used for finding patterns for specific domain. Since here in our project we are feeding hand crafted corpus which is reliable but sometimes it will be inadequate to find relationship data of two concepts. So as shape up the concept map we need to extra relationship data which can be obtained by crawling related web pages. By using some filtering strategies we can refine those identified relationships.

In the pattern base approach the terminology is used for those patterns is lexico –syntactic patterns. Those lexico syntactic patterns are expressed with combinations of Part of Speech (PoS) tags. For an example a typical hypernymy (or is-a) pattern is expressed as

<NP0> is-a <NP1> which

Here the <NP0> and <NP1> are PoS tags for noun phrases of a sentence.

3.4.3.4. Hypernymy Extraction

There are lot of research are going on hypernymy extraction. All those approaches can be categorized in to following types.

1. Analyzing the syntagmatic relations in a sentence
2. Analyzing the paradigmatic relations in a sentence
3. Document clustering

3.4.3.5. Analyzing the syntagmatic relations in a sentence

These types of methods usually employ set of PoS tag patterns to identify Hypernymy extraction. Popular pattern set is Hearst pattern set [13]

- NP_{hyper} such as { {NP_{hypo},}* (and|or) }NP_{hypo}
- suchNP_{hyper} as {NP_{hypo},}* { (and|or) }NP_{hypo}
- NP_{hypo}{,NP_{hypo}}*{,}or otherNP_{hyper}
- NP_{hypo}{,NP_{hypo}}*{,}and otherNP_{hyper}
- NP_{hyper}{,}including {NP_{hypo},}*{and|or}NP_{hypo}
- NP_{hyper}{,}especially{NP_{hypo},}*{and|or}NP_{hypo}

NP_{hyper}and NP_{hypo}are PoS variables which applied to arbitrary texts.

3.4.4. Relationship Extraction Frameworks

3.4.4.1. *Stanford parser*

Stanford parser[14] is a Java based typed dependency parser. Phrase structure tree can also be extracted at an intermediate step of the parsing. Development of phrase structure is done based on probabilistic context free grammar. These are necessarily context free grammar rules which have associated probabilities. Dependencies identified by the parser are triplets which consist of the name of the relation, governor and the dependent. A total of 53 typed dependencies are identified by the parser. Five different representations of typed dependencies are supported by the parser which is intended to give different level of information and different access methods to efficiently manipulate the output of the parser.

Following figure shows the phrase structure tree generated by the Stanford parser for the sentence “Java knowledge is helpful in Android application development.”

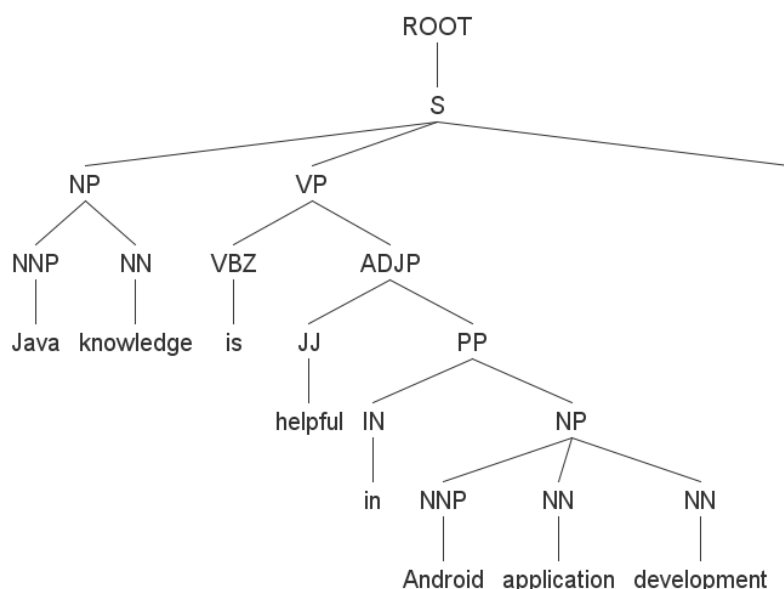


Figure 8 phrase structure tree generated by the Stanford parser

3.4.4.2. **RelEx - Relation extractor**

RelEx is a relation extracting tool which was initially developed by mainly targeting bioinformatics field. Getting details about some biological aspect was difficult, because large scaled biomedical publications need to refer for that. Also they need a way to get relations between some interrelated biological aspects e.g. physical or regulatory interactions between genes and proteins. In case of both above reasons they need a structured way to manage their free text resources. RelEx is the methodology used for that by extracting relationship from free text. But now RelEx is a part of

OpenCog's main project. It is creating an open source Artificial General Intelligence framework (A thinking machine).

There are several approaches exist for extracting relationships. Detecting co-occurrence of concepts within the sentence or whole document is one of them. This described, any set of concepts repeatedly occurred together then there should be any relationship between them. But the type and the direction of relation cannot be determined using this approach. Pattern-based extraction is another approach for relationship extraction. Here the extractions are extracted using pre-defined patterns. But this achieves lower recall. But RelEx used dependency parse tree to extract relationships

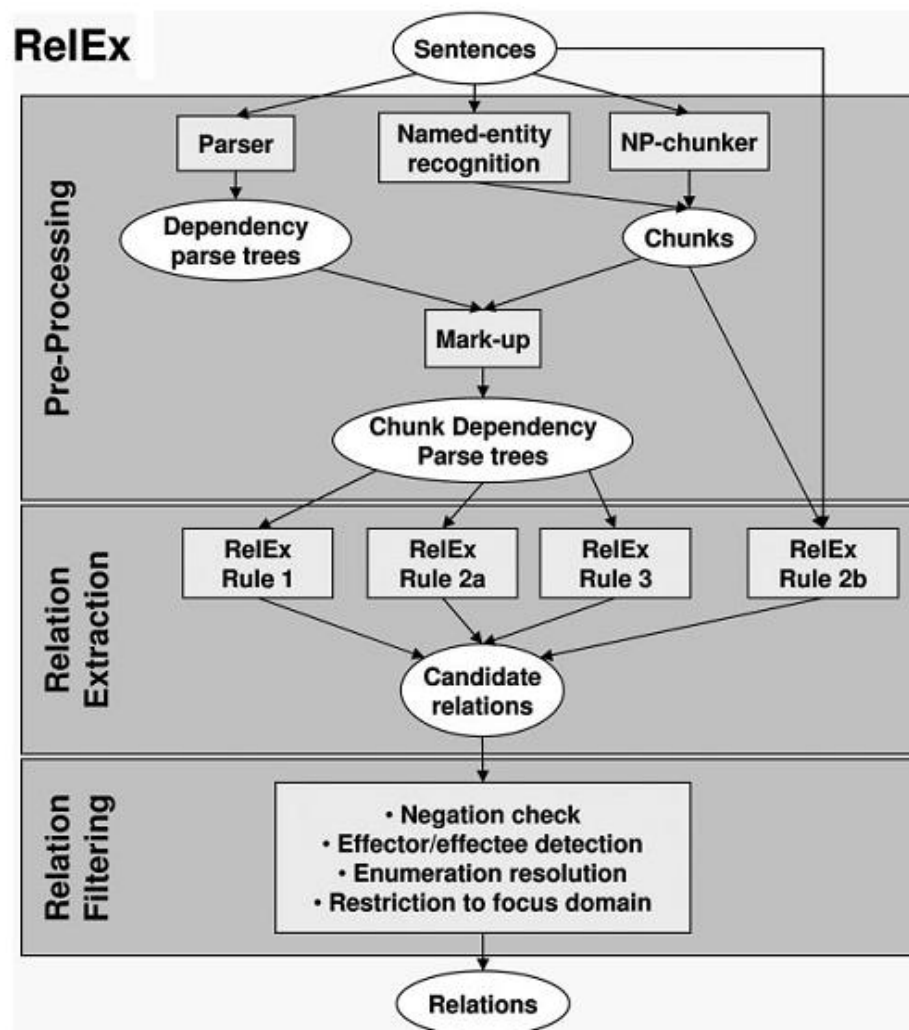


Figure 9 RelEx Relationship Extraction Chart

4. DESIGN

This chapter provides a comprehensive architectural overview of SIGMAC which is being developed to generate concept maps from enterprise documents, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

4.1. System Overview

SIGMAC is a standalone concept map generation application for browsing and visual summarization of documents and it is composed of five major components.

- Document Preprocessing
- Concept map generation
- Concept map optimization
- Concept map visualization
- Knowledgebase management

These components are arranged in a pipeline to so that each component would feed its output to the next component for further processing. Following diagram illustrates the overall system.

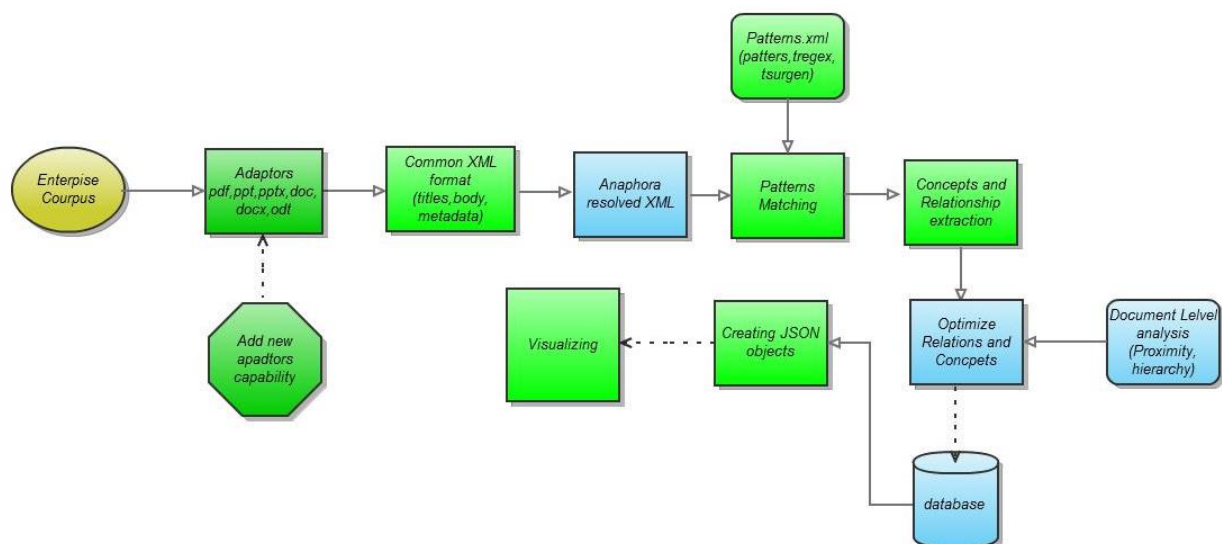


Figure 10 SigmaC Pipe Line

4.2. System Architecture of SigmaC

4.2.1. Introduction

System architecture of a system is the basic structure of system entities which describes the arrangements and configuration of them in order to satisfy the constraints and requirements of the system. Also it describes the interfaces between the entities and how they interact. Knowledge about the architecture is necessary to deliver a system which satisfying the requirements. Since this represents earliest design decisions, it is important to design good initial system architecture. Correct architecture leads to success implementation of the system.

4.2.2. Description of SigmaC Architecture

SigmaC system architecture designed as a layered architecture. It mainly consists of three layers, they are Document pre-processing layer, Core processing layer and Front-end.

Document pre-processing layer is the layer which can consider as the first phase of the system. It has the responsibility of processing documents. This has three major subparts; they are extracting body text and formatting details of document with different document types (doc, docx, ppt, pptx, pdf, odt and etc.), Resolving anaphora and Store them in an intermediate XML file.

Core processing layer is the main processing part of the system and it can be consider as the heart of the system. This contains main business logic of the system; it is extracting concepts and relationships between those concepts. This also a combination of several sub modules, they are Sentence Analyzer, Pattern engine, Concept/Relationship extractor, Optimizer and DB Manager. Concepts and relationships are extracted using phrase structure tree of each sentences and pre defined patterns. Then they are stored in a database.

Front-end is the layer which is communicates with end user of the system. This consists of three major sub modules; they are Access control module, Graph generation module and Request/Response module. Here graph generation module get the concepts and relationships from database and generate the concept map. Also users are facilitating to search concepts and add/edit/delete concepts and relationships. Here all the user requests are process in Request/Response module.

4.2.3. Architecture Diagram of SigmaC

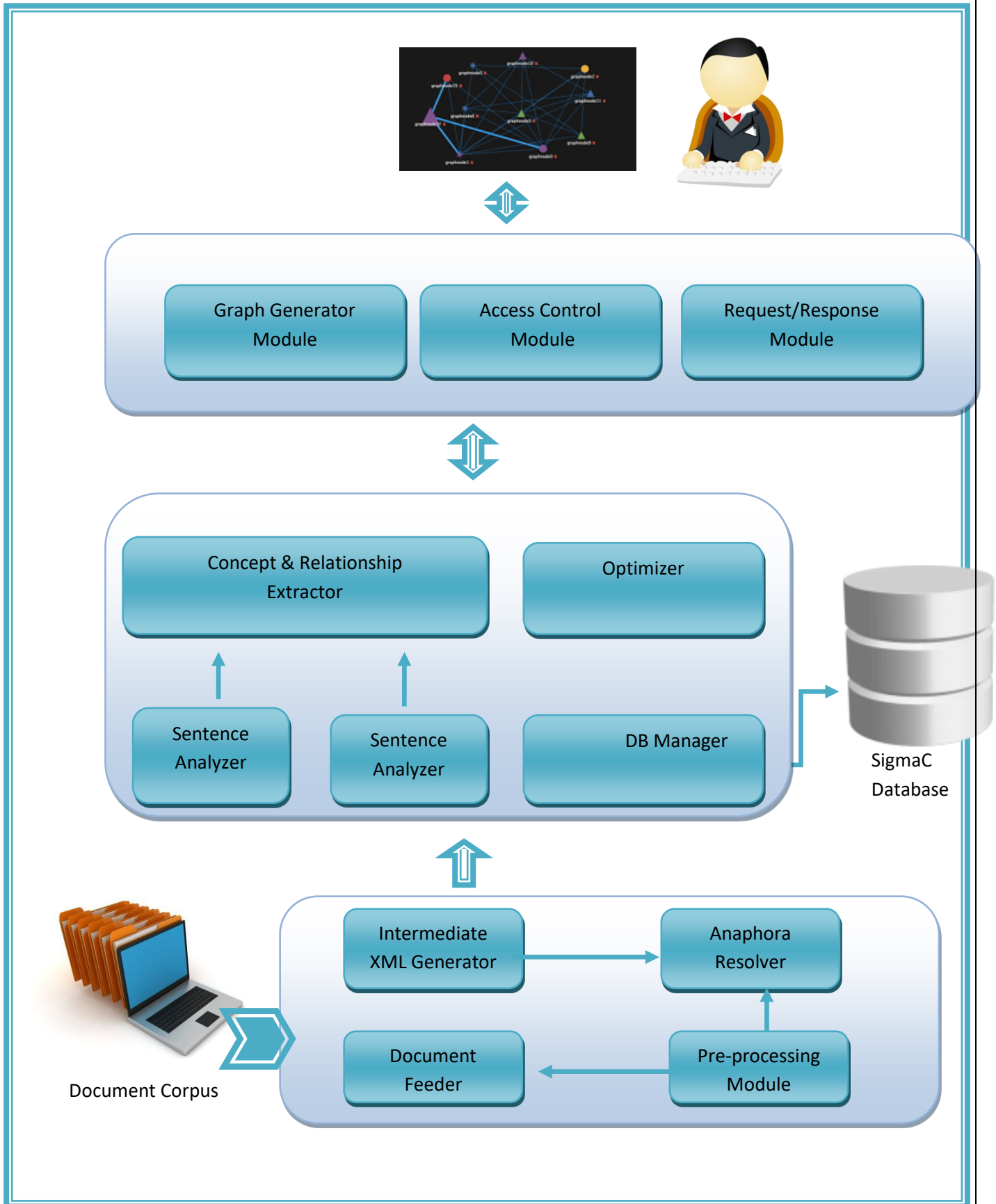


Figure 11 SigmaC Architecture

4.2.6.1. Introduction

These types of diagrams are used for describing functionality of a system. These diagrams contain the external interfaces and main uses of the software system. This view is mandatory when using the 4+1 Views, because all elements of the architecture should be derived from requirements.

4.2.6.2. Use Case Diagram

This software solution can be used by any authorized personal (Authenticating mechanism is not showed in the diagram) to view and retrieve information from the concept map. For the ease of describe the scenarios here the user is further specialized in to document uploader and document seeker.

- **Document Seeker**

User wants to browse through the concept map and choose relevant file. He/she can open any related document under a concept using the graph which represents the concept map.

- **Document Uploader**

Some authorized users can edit the concept graph using the interface. In the concept graph, users should be able to modify, add and delete relationship between two concepts. And User should able to add more documents into the concept map and introduce new repositories.

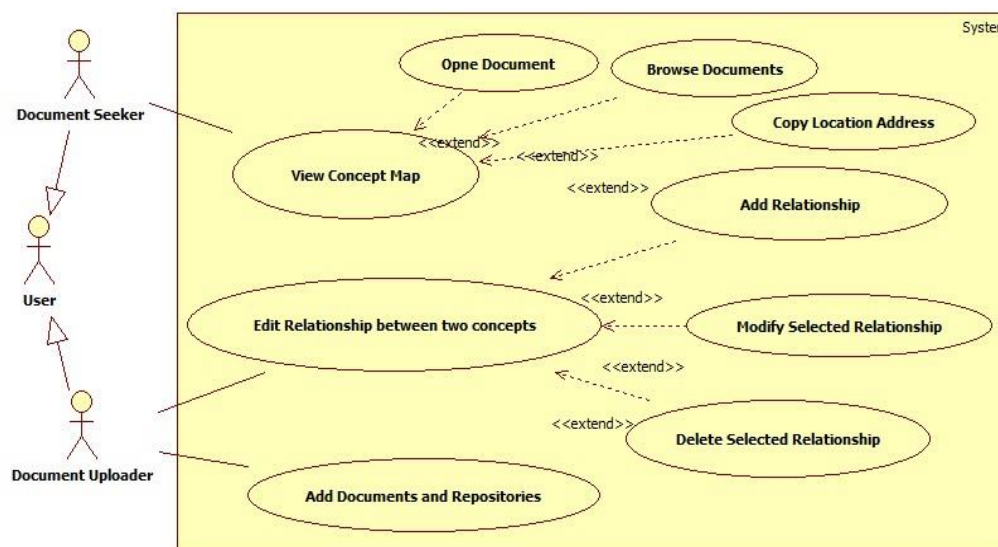


Figure 12 UseCase Diagram

4.2.7. Class Diagrams

4.2.7.1. Pre-Processing Module

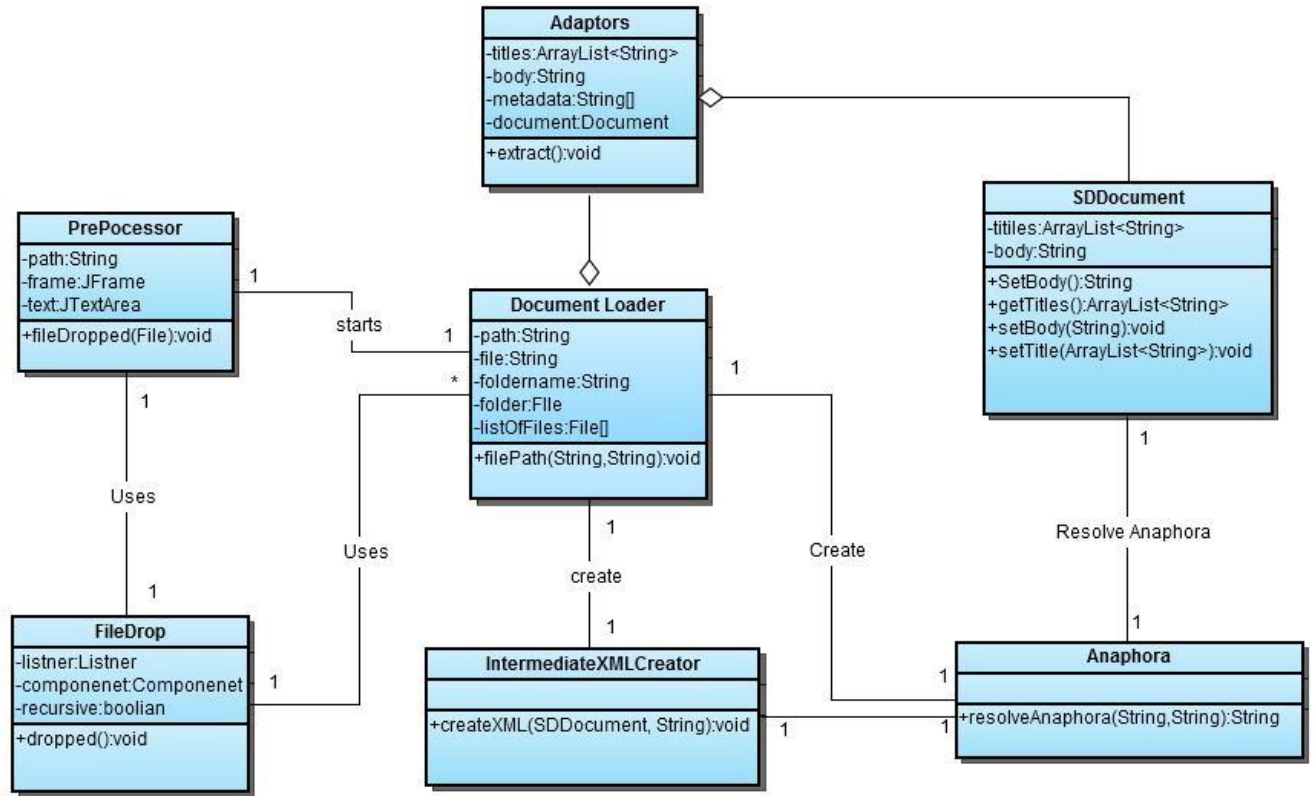


Figure 13 Class Diagram Pre Processing Module

4.2.7.2. Class Description

Class PreProcessor

This class is used to create a Drag and Drop GUI for adding documents to the system.

Class FileDrop

This class is used to provide methods used to implement the Drag and Drop GUI.

Class DocumentLoader

This is class which is controlled the whole pre possessing part. After identifying the actual file paths of the documents which are dragged to the drag and drop interface, it identifies the file format and load the relevant adaptor for text extraction.

Class Adaptors

These classes are the core of pre processing part and there are several classes for several document formats such as pdf, ppt, doc, docx, pptx, open document stands. Each of these adaptors extracts the text and identifies the actual titles of the documents. So the formatting details of each of document types used to identify the titles and the rest of the text which are not titles are considered as the body.

Class SDDocument

This is the intermediate class for setting and getting titles, body and metadata of the documents.

Class Anaphora

This class is used to resolve the anaphora in test body extracted from adaptors.

Class IntermediateXMLCreator

This class is used to create the intermediate XML file from the titles, body and metadata. Used body as the text which is resolved anaphora.

4.2.7.4. Concept & Relationship Extraction module

4.2.7.4.1. Class Diagram

Class Diagram for Concept & Relationship Extraction Module

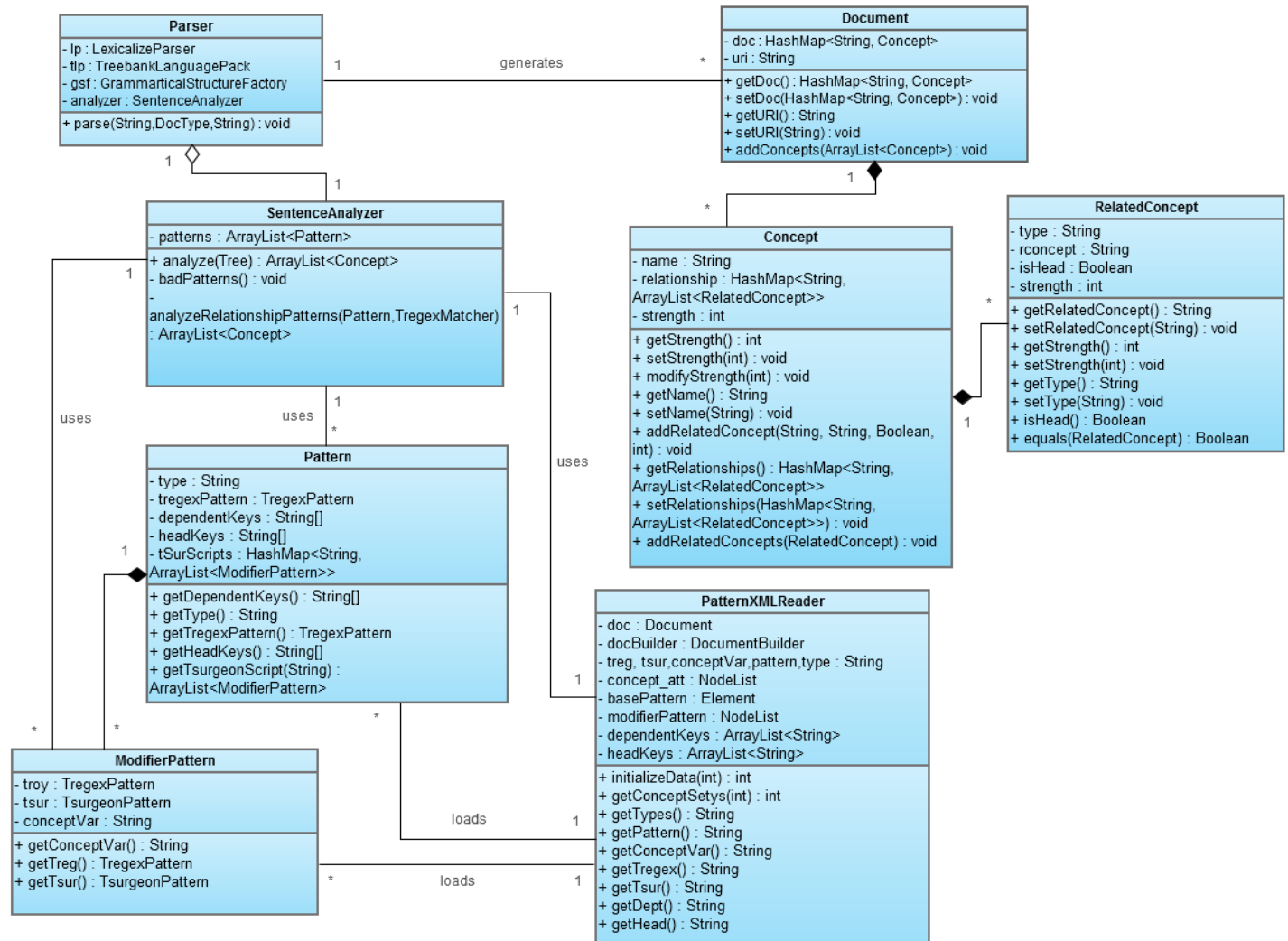


Figure 14 Class Diagram Concept and Relationship Extraction

4.2.7.4.2. *Class Description*

Class Concept

This class represents a concept in the concept map.

Class Document

This class represents the concept map generated by the parser.

Class Parser

This class provides interface between preprocessing module and optimization module. It is also responsible for using sentence analyzer to generate concept map for the document.

Class SentenceAnalyzer

This class is responsible for analyzing each individual sentence in the document and extract concepts and relationships from that sentence..

Class Pattern

This class represents patterns which are applied to individual sentences to identify concepts and relationships.

Class ModifierPattern

This class represents modifications that need to be made to the match segments of the sentence.

Class PatternXMLReader

This class is responsible for loading patterns from the xml document which contains patterns used by sentence analyzer.

Class RelatedConcept

This class represents relationships between concepts in the document.

4.2.7.5. Optimization Module

4.2.7.5.1. Class Diagram

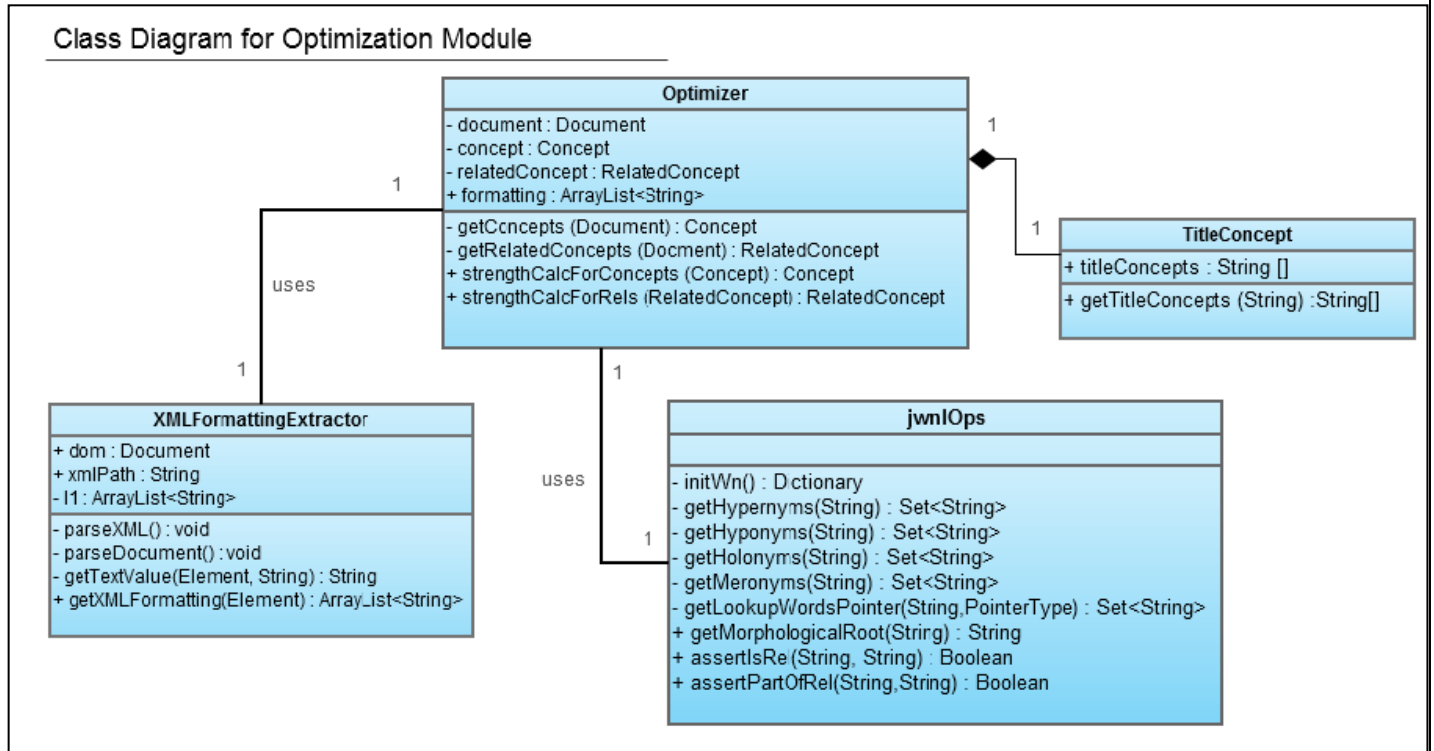


Figure 15 Class Diagram Optimization module

Optimization module is responsible for optimizing concepts and relationship using title text and wordnet library. Here concepts and relationships are weighted and filter to get most relevant set of concepts and relationships which are used to represents the document or corpus.

4.2.7.5.2. Class Description

Optimizer

This is the class which acts as the controller of optimizing module. This class is responsible for getting existing strength values of concepts and relationships and modifying it by using jwnlOps and TitleConcepts classes.

XMLFormattingExtractor

This class is written for traversing the intermediate XML files and getting titles.

jwnlOps

This is the class which use wordnet to get morphological root, assert is_a relationship and asserting part_of relationship of given concept/s.

TitleConcept

This class is responsible for extracting all the concepts of titles by considering collocations.

4.2.7.6. Visualization Module

4.2.7.6.1. Class diagram

Class Diagram

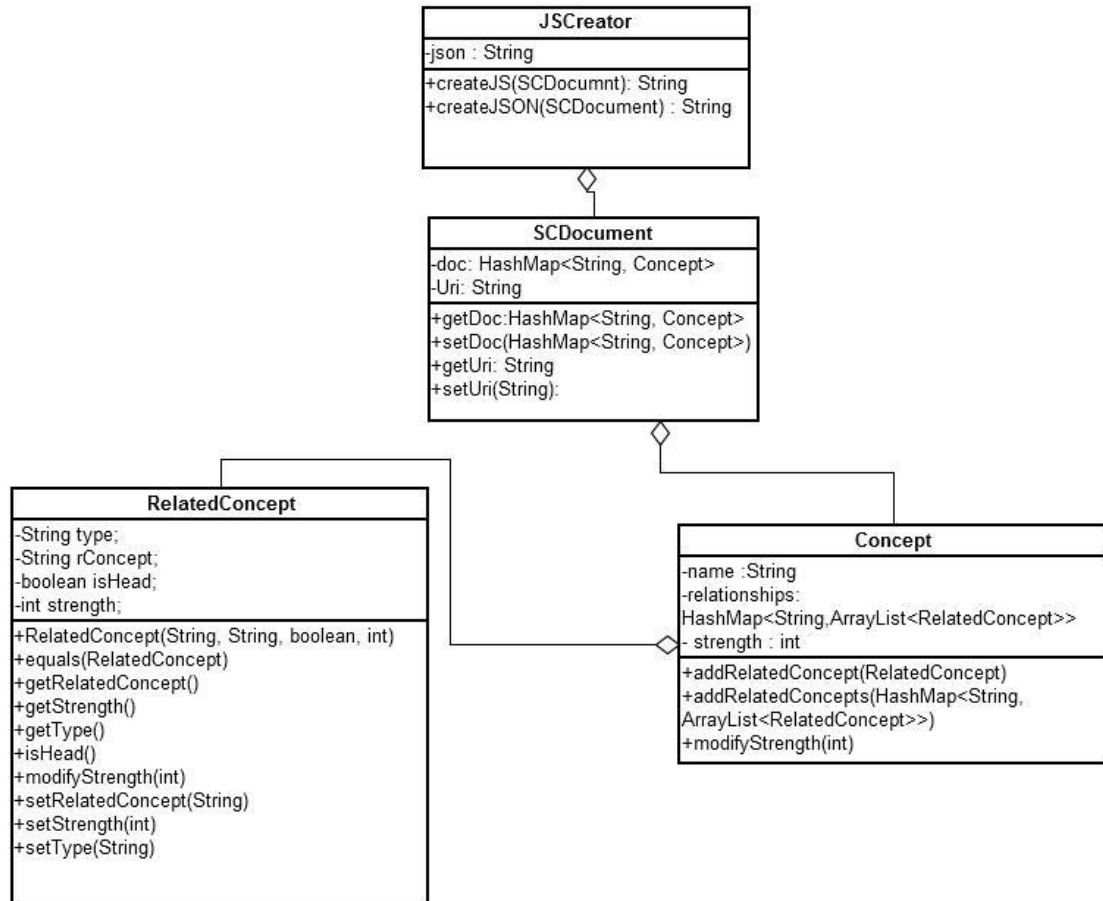


Figure 16 Class Diagram Visualization Module

Here the JSCreator class is act as a generator of JSON and JavaScript snippet for rendering concept map in web interface. createJSON() method will generate required JSON object to visualize the concept and map on the force directed graph in the relevant HTML div.

4.2.8. Activity Diagrams

4.2.8.1. General process of generating concept maps

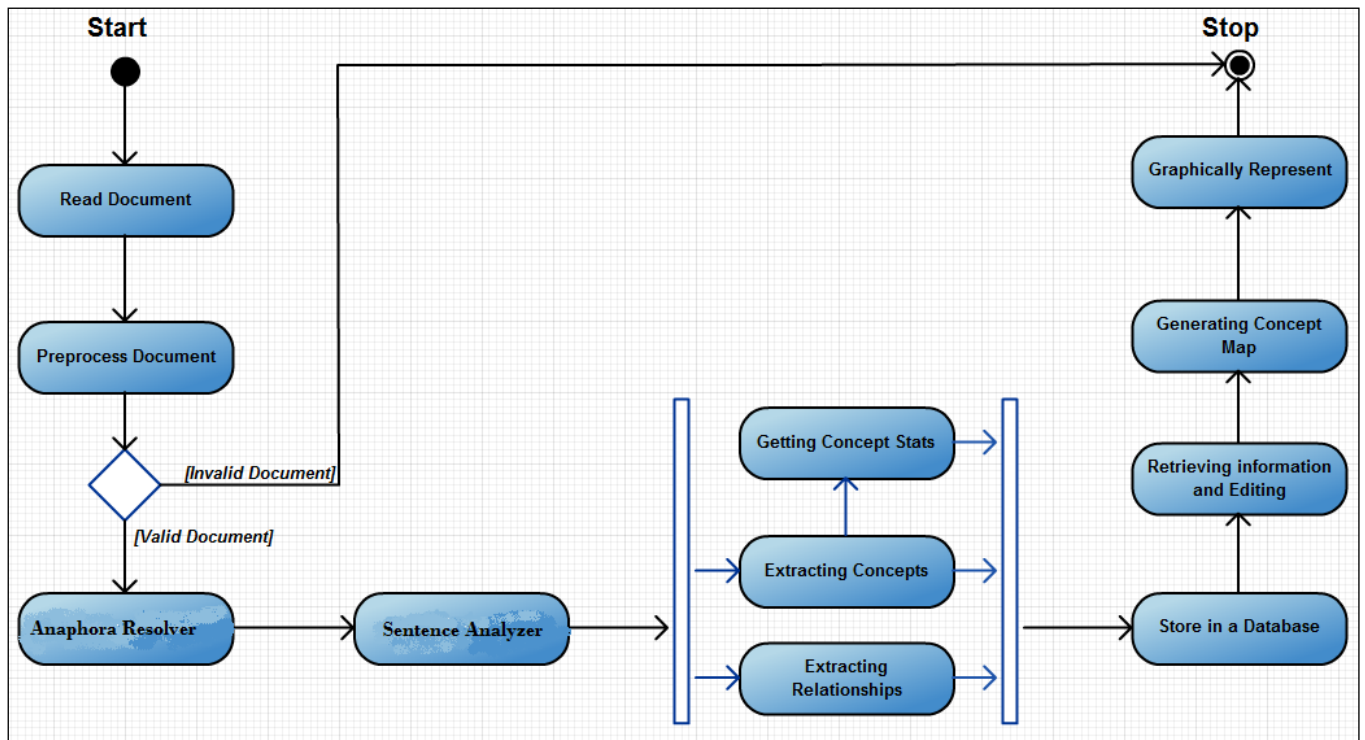


Figure 17 Activity Diagram Concept Map Generation

Above figure is the activity diagram of general process of generating concept maps. As shown in the figure initially the input documents are preprocessed and identify the validity of the document. Then the valid documents are further processed. The content of the document is processed by POS tagging sentences and building Chunks (this is important for identifying collocations).

Then using those data, concepts and their relationships and store them in database. Those information are used to generate concept maps and finally represented them in a graphical representation.

4.2.9. Concept map editing process

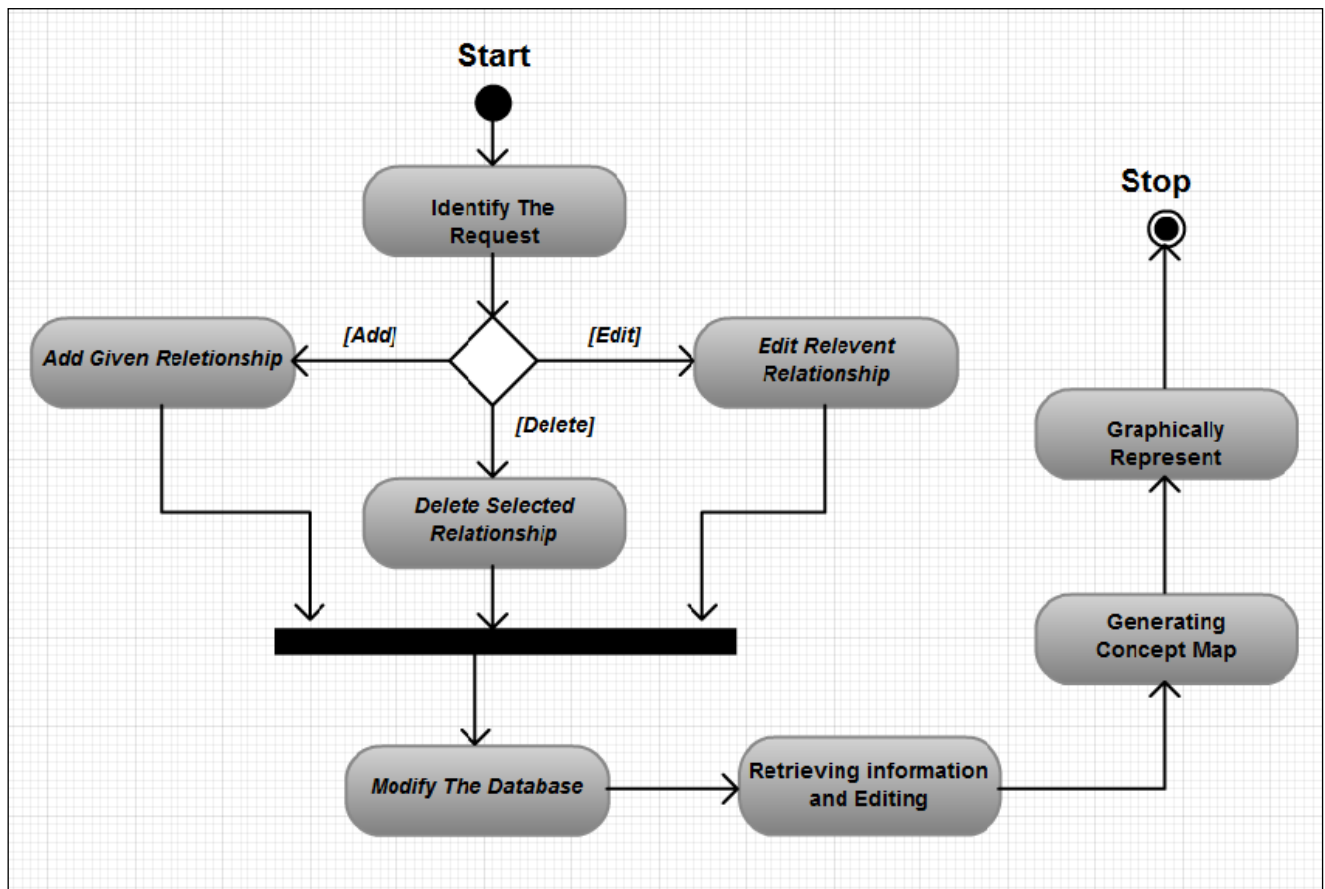


Figure 18 Activity Diagram Concept Map Editing

Since end user of the system has the facility to manually edit the generated concept maps. As shown in above activity diagram user can add, edit or delete the concepts and relationships of the concept map. After saving them those modified concepts and relationships database also updated and user can see the modified concept map.

5. IMPLEMENTATION

5.1. Pre Processing Stage

5.1.1. Adaptors

5.1.1.1. Open Documents Adaptor

This adaptor is used to extract body text and titles in the documents that stored in open document standards such as .ods , . odt etc. An open document is actually a zip file containing several data files in XML format representing the style, content and miscellaneous data. Body text (as paragraphs) and titles (as headlines) can be found in content.xml file under the XML tags `<text:p>` and `<text:h>` respectively. Using java in-built xml parsers like *DocumentBuilder* and *DocumentBuilderFactory* textual data is extracted.

```
NodeList nListParas = docm.getElementsByTagName("text:p");
NodeList nListHeaders = docm.getElementsByTagName("text:h");

// Appending Paragraphs to a StringBuffer
for (int temp = 0; temp < nListParas.getLength(); temp++) {
    Node nNode = nListParas.item(temp);
    bodybuff.append(nNode.getTextContent());
}
// Appending titles to a ArrayList
for (int temp = 0; temp < nListParas.getLength(); temp++) {
    Node nNode = nListHeaders.item(temp);
    if(nNode != null){
        titles.add(new Title(nNode.getTextContent()));
    }
}
```

5.1.1.2. Microsoft Word Documents adaptor (.doc /.ppt)

Doc is a file extension for word processing documents for Microsoft word 97-2003. Microsoft Word versions up to Word 97 used a different format from Microsoft Word 97 and later. In Microsoft Word 2007 and later, the binary file format was replaced as the default format by the Office Open XML format, though Microsoft Word can still produce DOC files. Due to doc files are not supported XML format, it is hard to extract relevant data from a file. So Apache POI library is used to extract information from doc files.

Text extraction can be done easily from using POI library.

```
docFile = new File(path);
FileInputStream fis=new FileInputStream(docFile.getAbsolutePath());
doc=new HWPFDocument(fis);
docExtractor = new WordExtractor(doc);
String all=docExtractor.getText();
```

In final line the string 'all' is assigned to the extracted text. The next thing is to find the titles from the document. There is no way of directly get the titles from a doc file. So in this adaptor, it is assumed that the contexts which has large font size than text common font size, as titles. Next thing is to find the common font size. It can be done using apache POI APIs.

```
public Object getBodyFont(Range overall,HWPFDocument doc,Range range,CharacterRun cr){

    ArrayList listfontsizes = new ArrayList();
    Set setfontsize = new HashSet();

    for(int i=0;i<overall.getEndOffset();i++){
        range = new Range(i, i+1, doc);
        cr = range.getCharacterRun(0);
        listfontsizes.add(cr.getFontsize());
        setfontsize.add(cr.getFontsize());
    }

    Object aArray[]=setfontsize.toArray();
    int bodyfontoccurance=0;
    Object BodyFONTsize=null;

    for(int i=0;i<setfontsize.size();i++){
        int occurrences = Collections.frequency(listfontsizes, aArray[i]);

        if(occurrences>bodyfontoccurance){
            bodyfontoccurance=occurrences;
            BodyFONTsize= aArray[i];
        }
    }

    return BodyFONTsize;
}
```

Then the body and titles can be extracted.

PPT is the file extension used in presentation programs in Microsoft Office 2003. Like doc file format ppt is also not supported XML format. Due to that reason direct information extraction is not possible as pptx. Apache POI library is used to extract the information from ppt file. In titles extraction it is assumed the same assumption used in doc file. Body font size calculation can be done using following

```
Map.Entry<Integer, Integer> maxEntry = null;

for (Map.Entry<Integer, Integer> entry : map.entrySet())
{
    if (maxEntry == null || entry.getValue().compareTo(maxEntry.getValue()) > 0){
        maxEntry = entry;
    }
}

return maxEntry.getKey();
```

code segment.

Title extraction can be done using POI APIs.

```

public ArrayList getTittles(Slide[] slide,int bodyfontsize){
    ArrayList<Title> titles = new ArrayList();
    for (int j = 0; j < slide.length; j++) {
        TextRun[] txt = slide[j].getTextRuns();

        for (int k = 0; k < txt.length; k++) {
            String text = txt[k].getText();
            RichTextRun[] richTextRuns = txt[k].getRichTextRuns();

            for (int l = 0; l < richTextRuns.length; l++) {
                if(richTextRuns[l].getFontSize()>bodyfontsize){
                    titles.add(new Title(richTextRuns[l].getText()));
                }
            }
        }
    }
    return titles;
}

```

So the body and titles can be extracted.

5.1.1.3. Microsoft PowerPoint Open XML Presentation (PPTX) Adaptor

This adapter is developed for extracting titles and body text with formatting details (Identify headings type and bold text) from the files with .pptx file extension. PPTX files are created using collection of separate XML files and folders in a compressed zip package. PPTX file has XML documents which contain properties, images, macros, charts, and other media files. In PPTX presentation content of each slide is stored in different XML file as ppt/slides/slide<slide no> and also note of each slide is stored in different XML file as ppt/notesSlides/notesSlide<slide no>. So in order to extract titles and body text of presentation it is required to parse and traverse all XML trees of them. NoteSlides normally contains only body text, so the content of the noteSlides directly append to body text. But slides are needed to process to get formatting details. A slide contains each paragraphs under the XML tag of <p:sp>. In each <p:sp> there are two sub nodes named <p:ph> (contains style of paragraph –ctrTitle, title and subTitle) and <a:t> (contains paragraph text). If any paragraph is bold then the value of attribute named ‘b’ in <a:rPr> sub node is set to one. All <p:sp> nodes with styles such as, ctrTitle, title and subTitle are taken as titles and the rest of paragraphs are combined together to get body text. For the implementation of the adaptor DOM XML parser was used.

```

String style="";
NodeList styleTE = empEl.getElementsByTagName("p:nvSpPr");
if(styleTE != null && styleTE.getLength() > 0) {
    for(int i = 0 ; i < styleTE.getLength();i++) {

        Element st = (Element)styleTE.item(i);

        NodeList styleE = st.getElementsByTagName("p:nvPr");
        if(styleE != null && styleE.getLength() > 0) {
            for(int j = 0 ; j < styleE.getLength();j++) {
                Element se = (Element)styleE.item(i);
                NodeList styleE1 = se.getElementsByTagName("p:ph");
                if(styleE1 != null && styleE1.getLength() > 0) {
                    Element spe = (Element)styleE1.item(0);
                    style=spe.getAttribute("type");
                }
            }
        }
    }
}

```

Figure 20 Traverse <p:sp> node to get style of the paragraph

```

NodeList formattingParaE = empEl.getElementsByTagName("a:p");
String[] allTxt = new String[formattingParaE.getLength()];
if(formattingParaE != null && formattingParaE.getLength() > 0) {
    for(int i = 0 ; i < formattingParaE.getLength();i++) {

        Element fpe = (Element)formattingParaE.item(i);
        NodeList formattingE = fpe.getElementsByTagName("a:r");
        if(formattingE != null && formattingE.getLength() > 0) {
            for(int j = 0 ; j < formattingE.getLength();j++) {

                Element fe = (Element)formattingE.item(j);
                String arText=getTextValue(fe, "a:t");
                allTxtString+= arText;

                NodeList boldE1 = fe.getElementsByTagName("a:rPr");
                if(boldE1 != null && boldE1.getLength() > 0) {
                    Element spe = (Element)boldE1.item(0);
                    String boldS=spe.getAttribute("b");
                    if(boldS.equals("1")){
                        boldString+=arText;
                    }
                }
            }
        }
    }
}

```

Figure 19 Traverse <p:sp> node to get txt content and bold text of the node (Part of a method)

5.1.1.4. Microsoft Word Open XML Document (DOCX) Adaptor

This adaptor is developed for extract titles and body text with formatting details (Identify headings type and bold text) from the files with .docx extension. DOCX files are created using collection of separate XML files and folders in a compressed zip package. DOCX consist of three folders, Word, docProps and _rels. They hold XML files which contain the content, document properties, and relationships between the files. For the requirement of DOCX adaptor only the content of word/document.xml file was sufficient. It contains each paragraphs under the XML tag of <w:p>. In each <w:p> there are two sub nodes named <w:pStyle> (contains style of paragraph –Heading 1,2,3,4) and <w:t> (contains paragraph text). If any paragraph is bold then there is <w:b> sub node contains in <w:p> node. All <w:p> nodes with styles such as, Heading1/2/3/4 are taken as titles and the rest of paragraphs are combined together to get body text. For the implementation of the adaptor DOM XML parser was used. It converts XML file into a JavaScript accessible object (the XML DOM) which contains functions to easily traverse the XML file.

```
String style="";
NodeList styleTE = empEl.getElementsByTagName("w:pPr");
if(styleTE != null && styleTE.getLength() > 0) {
    for(int i = 0 ; i < styleTE.getLength();i++) {

        //get the w:pPr element
        Element st = (Element)styleTE.item(i);

        NodeList styleE = st.getElementsByTagName("w:pStyle");
        if(styleE != null && styleE.getLength() > 0) {
            for(int j = 0 ; j < styleE.getLength();j++) {
                //get the w:pStyle element
                Element se = (Element)styleE.item(i);
                style=se.getAttribute("w:val");
            }
        }
    }
}
```

Figure 21 Traverse <w:p> to get style of the paragraph

```

NodeList formattingE = empEl.getElementsByTagName("w:r");
String[] allTxt = new String[formattingE.getLength()];
String[] isBold = new String[formattingE.getLength()];
if(formattingE != null && formattingE.getLength() > 0) {
    for(int j = 0 ; j < formattingE.getLength();j++) {

        Element fe = (Element)formattingE.item(j);
        allTxt[j]= getTextValue(fe, "w:t");

        NodeList boldE = fe.getElementsByTagName("w:rPr");
        if(boldE != null && boldE.getLength() > 0) {
            Element be = (Element)boldE.item(0);
            NodeList l11= be.getElementsByTagName("w:b");
            if(l11 != null && l11.getLength() > 0) {
                isBold[j]= "Bold";
            }
        }

    }
}

```

Figure 22 Traverse <w:p> node to get text content (to array) and check whether they are bold

5.1.2. Intermediate XML generator

Data which were extracted in previous stage should be stored in files in order to give input to the extraction stage. Storing in XML files were identified as most convenient way because of the simple and efficient nature of the XML files. These files are temporary files that need to feed into next stage.

Sample intermediate XML file

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<document>
  <Url>C:\Users\COMPAQ\Desktop\Visualization of Concept and relationships.docx</Url>
  <LastModified>03/16/2012 9.46AM</LastModified>
  <titles>
    <title scale="5">System Features</title>
    <title scale="5">Future Enchantments</title>
    <title scale="4">Support Multiple Document types</title>
    <title scale="4">Implement as a web interface</title>
    <title scale="4">Advance document browsing mechanism via concept map</title>
    <title scale="4">Visualization of Concepts and relationships</title>
    <title scale="4">Convenient and User friendly interface</title>
    <title scale="4">User Editable self-Adjustable Concept map</title>
    <title scale="4">Fast and accurate search mechanism</title>
    <title scale="4">Identifying multi-term words and acronyms</title>
    <title scale="4">Different roles for different security levels</title>
    <title scale="4">Supporting multiple languages</title>
    <title scale="4">3D concept map visualization</title>
    <title scale="4">Rule learning Engine</title>
    <title scale="4">Extend to analyze internet</title>
  </titles>
  <body>
    There are several document types that stores enterprise knowledge base. This projec
    docx, open document formats) that gather the knowledge base of an enterprise. The s
    The System should be able to access through several hosts. It should use client- se
    architecture with web based client interface.
    The system should visualize all the concepts and relationships. So, concept map is
```

5.1.3. Client side GUI implementation

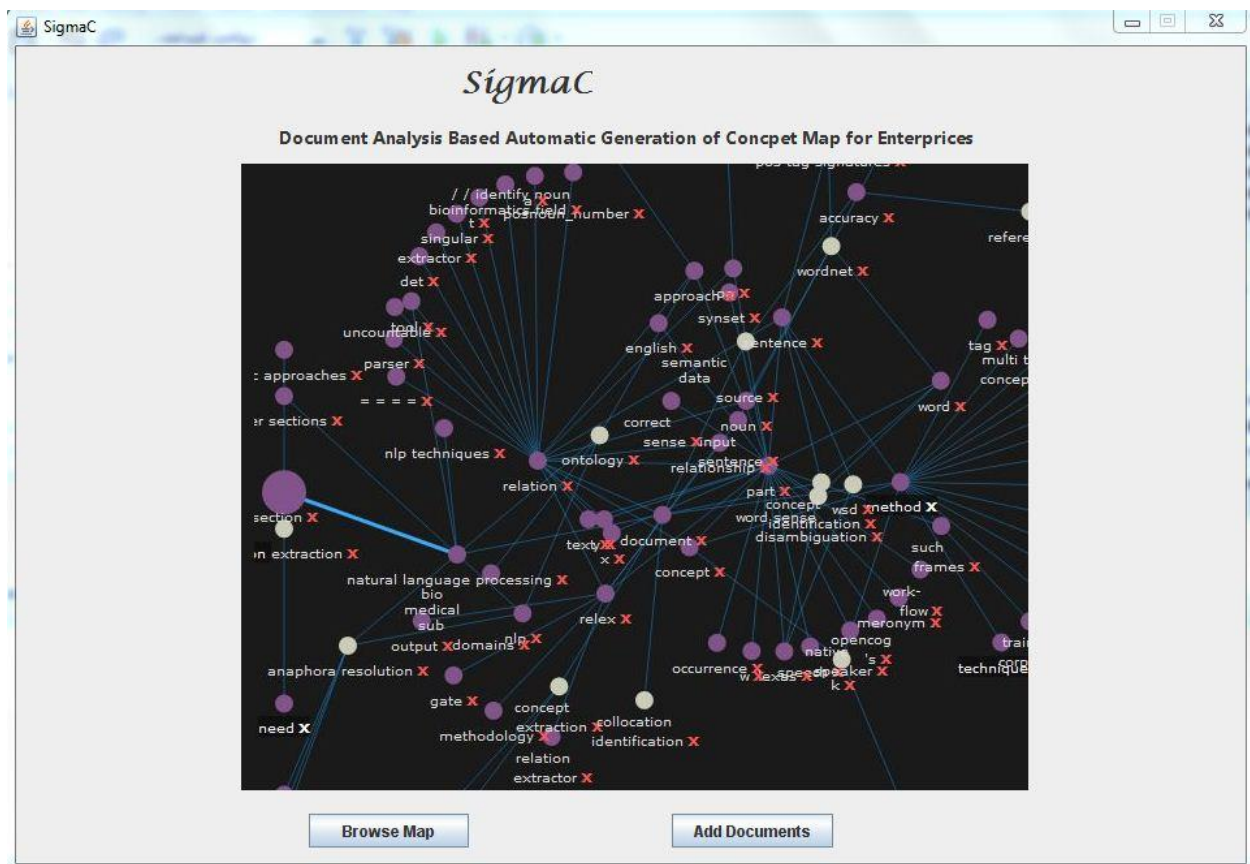


Figure 23 SigmaC Client GUI

This is the first page of clients. Client has two options to select. First one is “Browse Map”. From that client redirect to the default web browser and show the current overall repository concept Map. The other option is to “Add Documents” option and client can add new documents to the system from using that option. This GUI is designed by using java JFrame swing component. Buttons are designed using java JButton swing component. When user click on the “Add Documents” button, the current JFrame is set to invisible and new JFrame is created and (SigmaC File Drop) set it to visible. This JFrame consists of non-editable JTextArea. Users can drag relevant files to top of this text area and it identifies the actual file path of files in drag event.

```
FileDrop fileDrop = new FileDrop(System.out, text, new FileDrop.Listener() {  
  
    public void filesDropped(java.io.File[] files) {  
  
        DragFileProcessor tet=new DragFileProcessor(files,text,mpa);  
        Thread tt=new Thread(tet);  
        tt.start();  
        jButton1.setEnabled(true);  
    }  
});
```

Progress bar is designed using JProgressBar component.

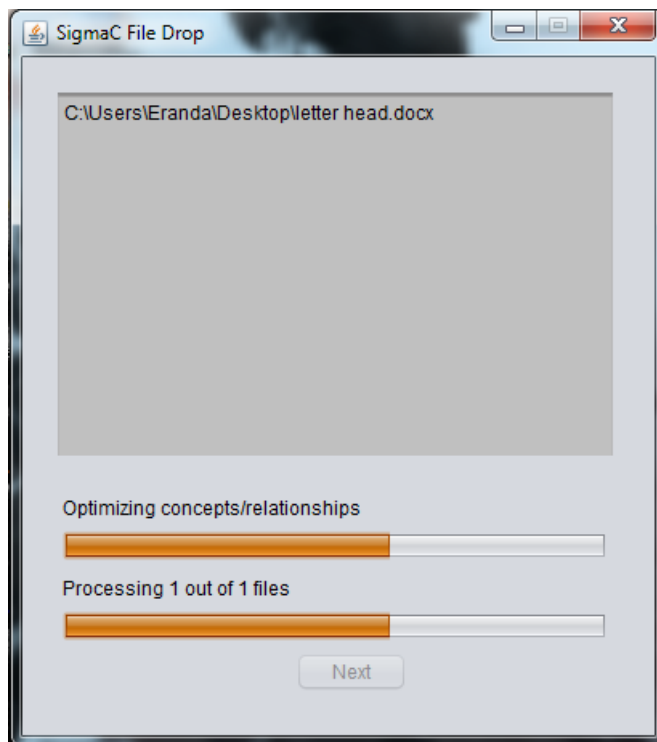


Figure 25 File Drop Box

The “Next” button is set to enable after processing all the files and then Adjust Frame JFrame is created and set to visible.

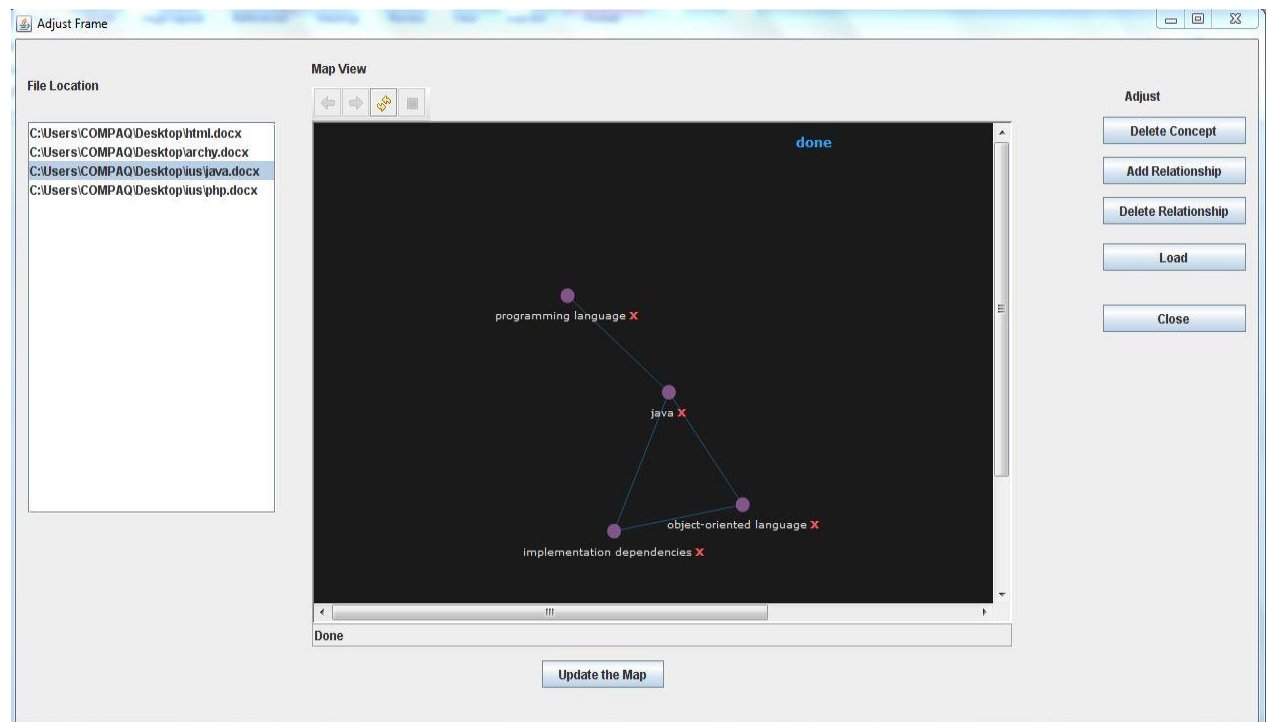


Figure 24 SigmaC Client Editble Map

This frame consists of a JList and a JPanel. HTML widget is included into the JPanel for showing the map. This widget is support JavaScript and when user select the file location relevant map is generated inside widget.

```
super(new BorderLayout());
JPanel webBrowserPanel = new JPanel(new BorderLayout());
final JWebBrowser webBrowser = new JWebBrowser();
webBrowser.setLocationBarVisible(false);
webBrowser.setMenuBarVisible(false);

webBrowser.navigate(path);
webBrowserPanel.add(webBrowser, BorderLayout.CENTER);
add(webBrowserPanel, BorderLayout.CENTER);
webBrowserPanel.setVisible(true);
```

The client should be able to edit the map and save changes. Client can click the cross symbol in concept and delete the concept. It is triggered in JavaScript and writes the deleted concepts into a file. When user clicks on the “Update the Map” button, it reads the concepts from the file and updates the document object.

```
closeButton.onclick = function() {
    ...
    var actualpath=replace2;

    WriteToFile("../src/temp/rnodes.dpi",node.name,actualpath);

    node.setData('alpha', 0, 'end');
    node.eachAdjacency(function(adj) {
        adj.setData('alpha', 0, 'end');
    });
    fd.fx.animate({
        modes: ['node-property:alpha',
               'edge-property:alpha'],
        duration: 500
    });
};
```

5.1.4. Anaphora Resolution

Anaphora resolution is a very important task of the project. Since the content of the document is fed to concept/relationship extractor sentence by sentence. Designing a new anaphora resolution engine is a tough task. So a decision taken to research about existing anaphora resolution engines and select best one for the project. Literature review covered this in detailed. Reconcile anaphora resolution engine is selected for that requirement.

Reconcile is actually a co reference identifier. Input for the engine should be an array of text files. After processing the set of files engine creates files with the name of filename.coref. Since here the system process one file at a time an array contains only one file path.

```
String[] x=new String[1];
x[0]=tempFileDest;
Resolver.main(x);

try{
    // Open the file first
    FileInputStream fstream = new FileInputStream(x[0]+".coref");
    // Get the object of DataInputStream
    DataInputStream in = new DataInputStream(fstream);
    BufferedReader br = new BufferedReader(new InputStreamReader(in));
```

Figure 26 Input file path array and process resulted file

Those coref files contain tagged text. In the result each proper nouns and pronouns are surrounded by <NP></NP> tags. In the tag there are two properties are exist, they are NO and CorefID. If any set of noun phrase with same CorefID refers first proper name with that CorefID. So all the pronouns are replaced by that proper noun and remove NP tag to get anaphora resolved text.

5.2. Concept and relationship extraction

Concept and relationship extraction component implements the core functionality required by the system. This component is responsible for generating concept map for a given document. Preprocessed XML document containing titles and text body is fed as the input and the concept map is provided as the output of the component. Furthermore, this defines the essential data structures to represent concept map for a given document and to represent patterns which are used to extract concepts and relationships.

5.2.1. Data Structures for representing concept map

Three classes are used to build up the data structure of the concept map. They are,

- Concept – represents a concept
- Related Concept – represents a relationship

- Document – represents the concept map built using above two classes

Concept Class

Following are the member variables of the class.

```
private String name;
private HashMap<String,ArrayList<RelatedConcept>> relationships;
private int freequency;
private float importance;
```

Here, name is the actual string representing the concept and frequency is the number of occurrences of the concept. The member variable “importance” provides the rank of the concept in the concept map and its actual value is calculated after the optimization phase. The “relationships” member holds the set of relationships of the concept found in the given document. The key of the map is the related concept and the list hold the set of relationships by means of RelatedConcept objects. The list is used in order to represent scenarios where two concept are related in different ways, for example, two concepts might have an “is-a” relationship and an association between them, this causes the requirement to maintain two different set of statistics to represent two types of relationships.

All these members are exposed through public interfaces and public methods are provided to add relationships while preserving the semantics and statistics. Following is an example of that.

```
public void addRelatedConcept(String name,String type,boolean isHead,int freequency){
    ArrayList<RelatedConcept> list;
    RelatedConcept con=new RelatedConcept(type, name, isHead, freequency);
    boolean match=false;
    if(this.relationships.containsKey(name)){
        list=relationships.get(name);
        for(RelatedConcept c : list){
            if(con.equals(c)){
                c.modifyFreequency(freequency);
                match=true;
                break;
            }
        }
        if(!match){
            list.add(con);
        }
    }else{
        list=new ArrayList<RelatedConcept>();
        list.add(con);
        relationships.put(name, list);
    }
}
```

RelatedConcept Class

This class is used to represent relationship between two concepts and specifically this only refers to a relationship of a particular type. Therefore, in order to represent two types of relationships between two concepts, two instances of the class is required. Following are the member variables of the class.

```
private String type;  
private String rConcept;  
private boolean isHead;  
private int freequency;
```

The member “type” describes the type of the relationship (“is-a”, association) and the member “rConcept” contains the string for the related concept. Direction of the relationship is described using “isHead” attribute and the “frequency” member keeps track of the number of occurrences of the relationship.

All members are exposed via public interfaces and the equality of two instances is checked using the following method.

```
public boolean equals(RelatedConcept rc){  
    if(this.isHead==rc.isHead && this.rConcept.equals(rc.getRelatedConcept()) && this.type.equals(rc.getType())){  
        return true;  
    }else{  
        return false;  
    }  
}
```

Document Class

This class represents a processed document and it contains the concept map to represent the document. Following are the members of the Document class.

```
private HashMap<String,Concept> doc;  
private HashMap<String,Title> title=new HashMap<String, Title>();  
private String uri;  
private String name="";  
private int size=0;
```

The variable “doc” is used to hold the concept map for the document and the “title” member used to store the titles found in the document which are being used in the optimization phase to improve the concept map. Uri and name attributes hold data about the location of the document and the “size” member variable hold a count of the total number of sentences found in the document.

All members are exposed through public interfaces and utility methods are provided to add new concepts to the document while preserving the semantics of the Document class. Following method is used to add concepts to the concept map.


```

public void addConcepts (ArrayList<Concept> concepts) {
    for (Concept c : concepts) {
        if (doc.containsKey(c.getName())) {
            Concept con = doc.get(c.getName());
            doc.remove(c.getName());
            if (con.getFreequency() <= 0) {
                con.modifyFreequency(1);
            } else {
                con.modifyFreequency(c.getFreequency());
            }
            con.addRelatedConcepts(c.getRelationships());
            doc.put(c.getName(), con);
        } else {
            doc.put(c.getName(), c);
        }
    }
}

```

5.2.2. Patterns

Patterns are used to identify concepts and relationships by matching them against phrase structure tree of each sentence in a document. These patterns are developed using Tregex and Tsurgeon. Tregex is used for identification and Tsurgeon is used for modifying the identified concepts in order to remove unwanted content from it.

Patterns are stored in an XML document so that new patterns can be added without modifying the system. Following is an example of the patterns XML document.

```

<relationship type="is_a">
  <pattern>
    <basePattern>
      <![CDATA["S: (NP=c1 $+ (VP=var1 <, (VBZ <: is & $+ (NP=c2 <, (DT=tmp <: a | <: an) & !<< NP & > =var1)))]"]>
    </basePattern>
    <concept name="c2" ishead="false">
      <modifierPattern>
        <tregex><![CDATA[NP <, DT=tmp]]></tregex>
        <tsurgeon>delete tmp</tsurgeon>
      </modifierPattern>
    </concept>
    <concept name="c1" ishead="true">
      <modifierPattern>
        <tregex><![CDATA[NP <, DT=tmp]]></tregex>
        <tsurgeon>delete tmp</tsurgeon>
      </modifierPattern>
    </concept>
  </pattern>
</relationship>

```

Here a base pattern is directly applied to the phrase structure tree of a sentence and that patterns names the tree nodes to be identified as concepts. Each tree node which was identified as a concept by the base patterns has a corresponding modifier pattern to perform necessary modifications to the selected node. Tregex in the modifier pattern is applied on the selected node and it marks the nodes that need to modify and the modifications are stated using Tsurgeon patterns.

Two data structure are used to define these patterns as objects. Those classes are,

- Pattern
- ModifierPattern

Pattern Class

Following are the member variables of the class. These members directly map the elements described in the patterns XML document. Here “headkeys” defines the set of concepts that would be treated as heads in the relationship.

```
private String type;  
private TregexPattern tregexPattern;  
private String[] dependentKeys;  
private String[] headKeys;  
private HashMap<String,ArrayList<ModifierPattern>> tsurScripts;
```

ModifierPattern Class

Following are the member variables of the ModifierPattern class. These members directly map the semantics of the modifier pattern tag of the above mentioned patterns XML document.

```
private TregexPattern treg;  
private TsurgeonPattern tsur;  
private String conceptVar;
```

5.2.2.1. Pattern loading

All the patterns are specified in a XML document. Therefore XML parser is used to load patterns in to object instances. Responsibility of the PatternXMLReader is to create Pattern and ModifierPattern objects from the patterns XML document. This class is used by the sentence analyzer to load patterns.

Parsing

Phrase structure tree generation is done using the Stanford parser. Each sentence of the document is fed in to the parser and patterns are matched against generated phrase structure tree. These are done using two classes. They are,

- Parser
- Sentence Analyzer

Parser

Parser is responsible for extracting text body from XML document and split it in to sentences, parses them to create phrase structure tree and feed it in to the sentence analyzer to analyze each sentence. Parser gathers each concept and relationship returned by the sentence analyzer to build the concept map for the document and return document containing its concept map.

```

public Document parse(String fileName, DocumentPreprocessor.DocType doctype,
    String delimiter) {
    DocumentPreprocessor p=new DocumentPreprocessor(fileName,doctype);
    p.setElementDelimiter(delimiter);
    Document doc=new Document(fileName);
    int sentenceCount=0;
    for (List<HasWord> sentence : p){
        sentenceCount++;
        Tree parse = lp.apply(sentence);
        ArrayList<Concept> cons=analyzer.analyze(parse);
        doc.addConcepts(cons);
    }
    XMLFormattingExtractor xe=new XMLFormattingExtractor();
    doc.setTitleInfo(xe.getXMLFormatting(doc.getUri()));
    doc.setSize(sentenceCount);
    return doc;
}

```

Sentence Analyzer

This class provides the most important functionality of the system. It analyzes phrase structure tree of each individual sentence fed as input and returns set of concepts and relationships that can be identified from the sentence using patterns. All the patterns for identifying relationships are treated in a similar manner while the generic patterns to identify concepts are treated differently.

Associations are defined for each pair of concepts that occur in a single sentence. Furthermore the analyzer avoids the concepts which are isolated. Patterns which are used in the analyzer are loaded from the patterns XML document and they are represented using above mentioned data structures. Data structures which were defined in above sections are heavily used within the sentence analyzer class. Furthermore sentence analyzer uses regular expressions to filter out unusual strings that were identified as concepts due to the grammatical mistakes present in sentences.

Following method implements the algorithm for extracting concepts and relationships from relationship patterns such as those used to identify “is-a”, “part-of” relationships.

```

private void analyseRelationshipPatterns(Pattern p, TregexMatcher matcher) {
    while(matcher.find()){
        ArrayList<String> hd=new ArrayList<String>();
        ArrayList<String> dp=new ArrayList<String>();
        for(String key : p.getHeaderKeys()){
            Tree head=matcher.getNode(key);
            ArrayList<ModifierPattern> ptns=p.getSurgeonScripts(key);
            if(ptns!=null){
                for(ModifierPattern mptn : ptns){
                    head=Tsurgeon.processPattern(mptn.getTreg(), mptn.getTsur(), head);
                }
            }
            if(head.firstChild()==null){
                continue;
            }
            List<LabeledScoredTreeNode> leaves=head.getLeaves();
            String hConcept="";
            for(LabeledScoredTreeNode leave : leaves){
                hConcept+=leave.nodeString()+" ";
            }
            hConcept=filterConcept(hConcept);
            if(cMap.containsKey(hConcept)) {
            }else{
                cMap.put(hConcept, new Concept(hConcept));
            }
            hd.add(hConcept);
        }

        for(String key : p.getDependentKeys()){
            Tree dependent=matcher.getNode(key);
            ArrayList<ModifierPattern> ptns=p.getSurgeonScripts(key);
            if(ptns!=null){
                for(ModifierPattern mptn : ptns){
                    dependent=Tsurgeon.processPattern(mptn.getTreg(), mptn.getTsur(), dependent);
                }
            }
            if(dependent.firstChild()==null){
                continue;
            }
            List<LabeledScoredTreeNode> leaves=dependent.getLeaves();
            String dConcept="";
            for(LabeledScoredTreeNode leave : leaves){
                dConcept+=leave.nodeString()+" ";
            }
            dConcept=filterConcept(dConcept);
            if(cMap.containsKey(dConcept)){
            }else{
                cMap.put(dConcept, new Concept(dConcept));
            }
            dp.add(dConcept);
        }
        for(String h: hd){
            Concept head=cMap.get(h);
            for(String d : dp){
                head.addRelatedConcept(d, p.getType(), false, 1);
            }
        }
        for(String d : dp){
            Concept dependent=cMap.get(d);
            for(String h : hd){
                dependent.addRelatedConcept(h, p.getType(), true, 1);
            }
        }
    }
}

```

Following method implements the algorithm for identifying concepts which may or may not have a direct involvement with the pre specified set of relationships. These are also included in defining associations between concepts.

```

private void analyzeConceptPatterns (Pattern p, TregexMatcher matcher) {
    while (matcher.find()) {
        for (String key : p.getHeadKeys()) {
            Tree match = matcher.getNode (key);
            for (ModifierPattern ptn : p.getSurgeonScripts (key)) {
                match = Tsurgeon.processPattern (ptn.getTreg(), ptn.getTsur(), match);
            }
            Tree t = match.firstChild();
            if (t == null) {
                continue;
            }
            List<LabeledScoredTreeNode> leaves = match.getLeaves();
            String concept = "";
            for (LabeledScoredTreeNode leave : leaves) {
                concept += leave.nodeString() + " ";
            }
            concept = filterConcept (concept);
            if (cMap.containsKey (concept)) {
            } else {
                if (concept.length() > 2) {
                    cMap.put (concept, new Concept (concept));
                }
            }
        }
    }
}

```

5.2.3.Optimizing Module

This is the module which optimizing extracted concepts and relationships between those concepts. The extracted concepts and relationships in the extractor needed to optimize in order to get best representing concepts of the document. In this module there is a property called strength is assigned to both concepts and relationships. The strength is the value which measures the importance of concepts and relationships. Finally those concepts are filtered by defined value to get best representing concepts for the document. Then relationships between those concepts are also selected for generating concept map for the document.

Concepts and their relationships are separately optimized. In the optimization of the concept first step is removing redundancy. In this phase morphological root of each concepts are considered and reduce all morphological forms of concept to its morphological root. Here those concepts are needed to join get optimized concept. As mentioned above a Document object is created to store extracted concepts and relationships from a document. In joining concepts, frequencies of concepts are added and all related concepts also joined to remove any redundancies. Morphological root is taken using WordNet library.

```
public String getMorphologicalRoot(String lexicalForm) throws FileNotFoundException, JWNLEException{
    Dictionary dictionary=initWn();
    String trimLexicalForm=lexicalForm.trim().toLowerCase();
    String morphWord=trimLexicalForm;
    IndexWord baseForms = dictionary.getMorphologicalProcessor().lookupBaseForm(POS.NOUN, trimLexicalForm);
    String[] inWords=trimLexicalForm.split(" \\\\_");
    int noOfInWords=inWords.length;

    if(baseForms!=null){
        String[] outWords=baseForms.getLemma().split(" ");
        int noOfOutWords=outWords.length;
        if(noOfInWords==noOfOutWords){
            morphWord=baseForms.getLemma();
        }
    }
    return morphWord.trim();
}
```

Figure 27 Method for getting morphological root

Above figure shows the method for getting morphological root. It returned morphological root if there is otherwise returning same string. If any set of concepts get same morphological root they combined together to get optimized concept object.

```

HashMap<String,Concept> doc1 = new HashMap<String, Concept>();
for(String con:doc.keySet()){
    String morphRoot;
    morphRoot=wn.getMorphologicalRoot(con);
    if(doc1.containsKey(morphRoot)){
        concept=doc1.get(morphRoot);
        concept.setFreequency(concept.getFreequency()+doc.get(con).getFreequency());
        concept.setName(morphRoot);
        relationships1=new HashMap<String, ArrayList<RelatedConcept>>();

        relationshipJoin(concept.getRelationships());
        relationshipJoin(doc.get(con).getRelationships());
    }
}

```

Figure 28 Optimizing concepts by using morphological root of concept

In joining concept objects a new HashMap is used to add concepts, in adding concept if a concept exists with same name (morphological root of the concept) their related concepts are also merged together. Related concepts are stored in an ArrayList, so both array lists are merged by removing redundancies. Also the related concept name set to its morphological root.

In the optimization phase there is a strength value assigned for each concepts. The strength value is used to set an importance value to a concept (importance value is overall measurement of importance of concept within the document). In setting strength to a concept there are several properties of document are taken in to account. They are,

- Frequency of concept in a document (F_c)
- Size of the document (No of sentences) (S)
- Number of titles in the document (N)
- Scale values of titles which the concept is containing (SV)
- Number of words in concept (NWC)
- Number of words in title which the concept is containing (NWT)

The strength value should be a normalized value, because it can be able to use in comparing concepts in different documents. In order to get normalized frequency value, it divided by the size of the document. If a title contains relevant concept; title scale, what percentage of title covered by the concept and number of title in the document, should be affected to strength value of the concept. Following conditions should be fulfilled to get an accurate value for concept strength.

Concept strength – CS

- By considering concepts properties
 - CS is directly proportional to concept frequency,

$$CS \propto \frac{F_c}{S}$$

- By considering titles properties

- CS is inversely proportional to number of titles in the document,

$$CS \propto \frac{1}{N}$$

- CS is directly proportional to the scale values of titles which the concept is containing and NWC to NWT ratio,

$$CS \propto \sum SV \times \frac{NWC}{NWT}$$

By considering above proportionalities, final equation for calculating concept strength can be obtained as below.

$$CS = k \times \frac{F_c}{S} + \frac{1}{N} \times \sum SV \times \frac{NWC}{NWT}$$

K=0.5 (Constant value)

Relationship optimization is done using WordNet library. Here 'is_a' and 'part_of' relationship optimized by asserting them using WordNet. Also 'aso' (associated) relationships are modified to 'is_a'/'part_of' relationship if any that kind of relationship exists in WordNet lexical database. WordNet library can be used to get hypernyms and hyponyms of concepts to asserting 'is_a' relationships.

- **Hypernyms:** Y is a hypernym of X if X is a Y (programming language is a hypernym of Java)
- **Hyponyms:** Y is a hyponym of X if Y is a X (Java is a hyponym of programming language)

Above hypernym results are used for the assertion method of 'is_a' relationships.

Above method is used to confirm 'is_a' relationships extracted in the extracting module, also 'aso'

```
public boolean assertIsRel(String head,String tail) throws FileNotFoundException, JWNLEException{
    Boolean isConf=false;
    String morHead=getMorphologicalRoot(head);
    String morTail=getMorphologicalRoot(tail);
    String morTailMod=morTail.replaceAll(" ", "_");
    Set<String> isSet=getHypernyms(morHead);

    if(isSet.contains(morTailMod)){
        isConf=true;
    }
    return isConf;
}
```

Figure 29 Is_a relationship asserting method

relationships are modified to 'is_a' relationship if that relationship confirmed by the above method. Head of the relationship also set in the RelatedConcept object. Implementation of getHypernyms method is shown in Figure 7.

```

if(g.getType().equals("aso")){
    if(wn.assertIsRel(con, g.getRelatedConcept())){
        g.setType("is a");
        g.setIsStrength(1);
        g.setStrength(1);
        g.setHead(Boolean.FALSE);
    }else if(wn.assertIsRel(g.getRelatedConcept(), con)){
        g.setType("is a");
        g.setIsStrength(1);
        g.setStrength(1);
        g.setHead(Boolean.TRUE);
    }
}
}else if(g.getType().equals("is a")){
    g.setStrength(1);
    if(wn.assertIsRel(con, g.getRelatedConcept())){
        g.setIsStrength(1);
        g.setStrength(2);
    }else if(wn.assertIsRel(g.getRelatedConcept(), con)){
        g.setIsStrength(1);
        g.setStrength(2);
    }
}

```

Figure 30 Optimize 'is_a' relationships

WordNet library can be used to get holonyms and meronyms of concepts to asserting 'part_of' relationships.

- **Holonym:** Y is a holonym of X if X is a part of Y (computer is a holonym of processor)
- **Meronym:** Y is a meronym of X if Y is a part of X (processor is a meronym of computer)

Above meronym results are used for the assertion method of 'part_of' relationships.

```

public boolean assertPartOfRel(String head,String tail) throws FileNotFoundException, JWNLEException{
    Boolean partConf=false;
    String morHead=getMorphologicalRoot(head);
    String morTail=getMorphologicalRoot(tail);
    Set<String> partSet=getMeronym(morTail);
    String morHeadMod=morHead.replaceAll(" ", "_");

    if(partSet.contains(morHeadMod)){
        partConf=true;
    }
    return partConf;
}

```

Figure 31 part_of relationships assertion method

Above method is used to confirm 'part_of' relationships extracted in the extracting module, also 'aso' relationships are modified to 'part_of' relationship if that relationship confirmed by the above method. Head of the relationship also set in the RelatedConcept object. Implementation of

```

if(g.getType().equals("aso")){
if(wn.assertPartOfRel(con, g.getRelatedConcept())){
    g.setType("part of");
    g.setPartStrength(1);
    g.setStrength(1);
    g.setHead(Boolean.FALSE);
}else if(wn.assertPartOfRel(g.getRelatedConcept(), con)){
    g.setType("part of");
    g.setPartStrength(1);
    g.setStrength(1);
    g.setHead(Boolean.TRUE);
}
}else if(g.getType().equals("part of")){
    g.setStrength(1);
    if(wn.assertPartOfRel(con, g.getRelatedConcept())){
        g.setPartStrength(1);
        g.setStrength(2);
    }else if(wn.assertPartOfRel(g.getRelatedConcept(), con)){
        g.setPartStrength(1);
        g.setStrength(2);
    }
}

```

Figure 32 Optimizing part_of relationships

```

private Set<String> getHypernyms(String lexicalForm) throws FileNotFoundException, JWNLEException{
    return lookupWordsFollowingPointer(lexicalForm, PointerType.HYPERNYM);
}

private Set<String> getMeronym(String lexicalForm) throws FileNotFoundException, JWNLEException{
    return lookupWordsFollowingPointer(lexicalForm, PointerType.PART_MERONYM);
}

private Set<String> lookupWordsFollowingPointer(String lexicalForm, PointerType pointerType) throws
    Set<String> result = new HashSet<String>();
    Dictionary dictionary=initWn();
    IndexWord indexWord = dictionary.getIndexWord(POS.NOUN, lexicalForm);
    if (indexWord == null)
        return result;
    Synset[] synSets = indexWord.getSenses();
    for (Synset synset : synSets){
        if (synset.getPointers(pointerType).length>0){
            PointerTarget[] targets = synset.getTargets(pointerType);
            for (PointerTarget target : targets){
                Word[] words = ((Synset) target).getWords();
                for (Word word : words){
                    result.add(word.getLemma());
                }
            }
        }
    }
    return result;
}

```

Figure 33 Implementation of getHypernyms and getMeronym methods

getMeronyms method is shown in Figure 3.

A strength value is assigned for each relationship to get an idea about the importance of the relationship in the relevant document. In setting strength to a relationship there are several properties of document are taken in to account. They are,

- Frequency of relationship (only for association relationships) in a document (F_r)
- Size of the document (No of sentences) (S)

Relationship strength – RS

➤ For ‘aso’ relationships

- If ‘aso’ relationship modified to ‘is_a’ or ‘part_of’ in optimizing phase then $RS=1$.
- Otherwise RS is directly proportional to concept frequency, - Here RS should be less than 1 (‘is_a’ and ‘part_of’ relationships are more important)

$$RS = e^{-\left[\frac{1}{F_r}\right]}$$

- For 'is_a' and 'part_of' relationships – Here frequency doesn't considered by assuming same 'is_a' or 'part_of' relationship is not repeated in the same document.
 - RS value set to 1 if the relationship identified in concepts/relationships extraction phase – RS=1
 - If the relationship asserted by optimization phase (using WordNet) RS value increased by 1 – RS=2

5.2.4. Concept Ranking

Concept ranking in the concept map is based on the Page Rank algorithm. Document is considered as the web and individual concepts are treated as web pages in the context of SIGMAC for ranking concepts in a given document. Relationships between concepts are modeled as links and relationships other than associations get higher value by spawning multiple links to represent that relationship.

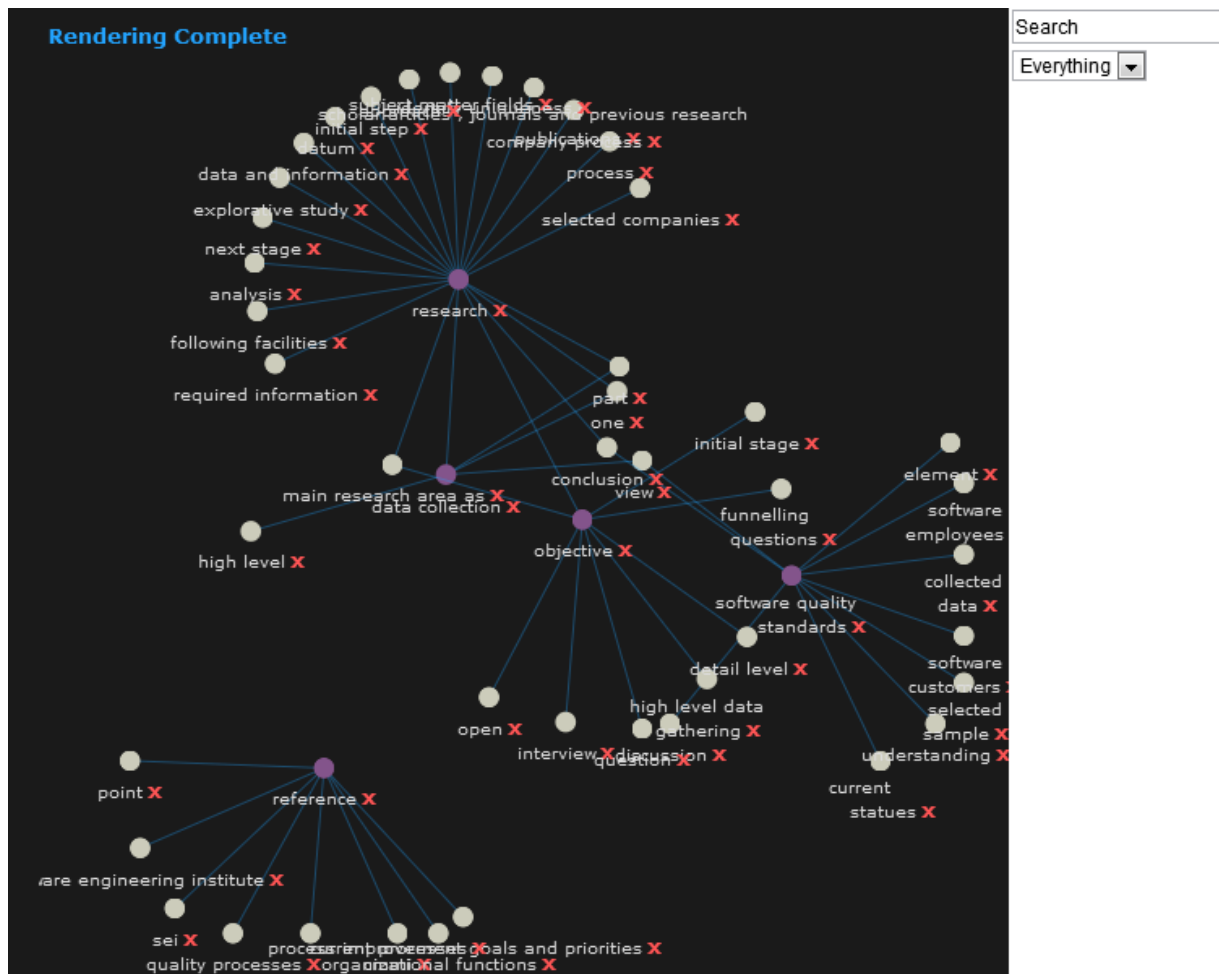
Following figure contains the algorithm used to rank concepts.

```

public void rankConcepts(Document doc) {
    HashMap<String, Concept> cMap = doc.getDoc();
    float totalDocumentValue = doc.getTotalDocumentValue();
    Set<String> keySet = cMap.keySet();
    float total=0.0f;
    int i=0;
    do{
        totalDocumentValue = doc.getTotalDocumentValue();
        i++;
        for(String key : keySet){           //operate on each concept
            //operate on each concept
            Concept concept = cMap.get(key);
            int totalOutLinks=concept.getTotalOutboundCount();           //total number of links going out
            //total number of links going out
            HashMap<String, ArrayList<RelatedConcept>> relationships = concept.getRelationships();
            Set<String> relKeySet = relationships.keySet();
            for(String relKey : relKeySet){           //operate on each related concept
                int outToConcept=concept.getOutboundCountTo(relKey); //links going to concept
                float linkFraction=(float)outToConcept/(float)totalOutLinks; //fraction of links
                float importance=(float)concept.getValue()*linkFraction/totalDocumentValue;
                Concept relatedConcept = cMap.get(relKey);
                relatedConcept.modifyValue(importance);
                total+=importance;
            }
        }
        System.out.println(doc.getTotalDocumentValue()-totalDocumentValue);
    }while(i<100 || (doc.getTotalDocumentValue()-totalDocumentValue)<1.0015);
    updateImportance(doc);
}

```

5.3. Visualization Server with Database



SigmaC concept map was represented via a force directed Graph where the node represent the concepts and the edges are representing the relationships among them. The graph is rendered in the SigmaC dash board by using an open source JavaScript Library called **TheJit**. The data required to render the concept map should be formatted as a JSON data representation structure. Following is the fragment of the JSON object which used to represent one node (Concept) and one edge (relationship). The data that used to compose this JSON object is retrieved by MySQL data base.

```
32  var json = [  
33      {  
34          "id" : "technique",  
35          "name" : "technique",  
36          "data" : {  
37              "$color" : "#83548B",  
38              "$type" : "circle"  
39          },  
40          "adjacencies" : [  
41              {  
42                  "nodeTo" : "method",  
43                  "nodeFrom" : "technique",  
44                  "data" : {}  
45              }  
46          ],  
47      }  
48  ]
```

The SigmaC dashboard is a web application which is very convenient to use in almost all the modern web browsers such as Google Chrome, Mozilla Firefox and Internet Explorer. Java Server Pages, JavaScript and HTML technologies were used while developing this web application. As the application container Apache Tomcat 7.0 was used.

5.3.1.Database Implementation

To store the data which were extracted and uploaded from the client application, a data base has to be established in the server. In order to do this task a relational database was chosen because of the following reasons.

- Multiple queries should execute for retrieving data for draw the concept map
- Server processing for ranking documents for the concepts is needed and that requires relational database model.

Database Schema represented by Entity Relationship Diagram of the Database.

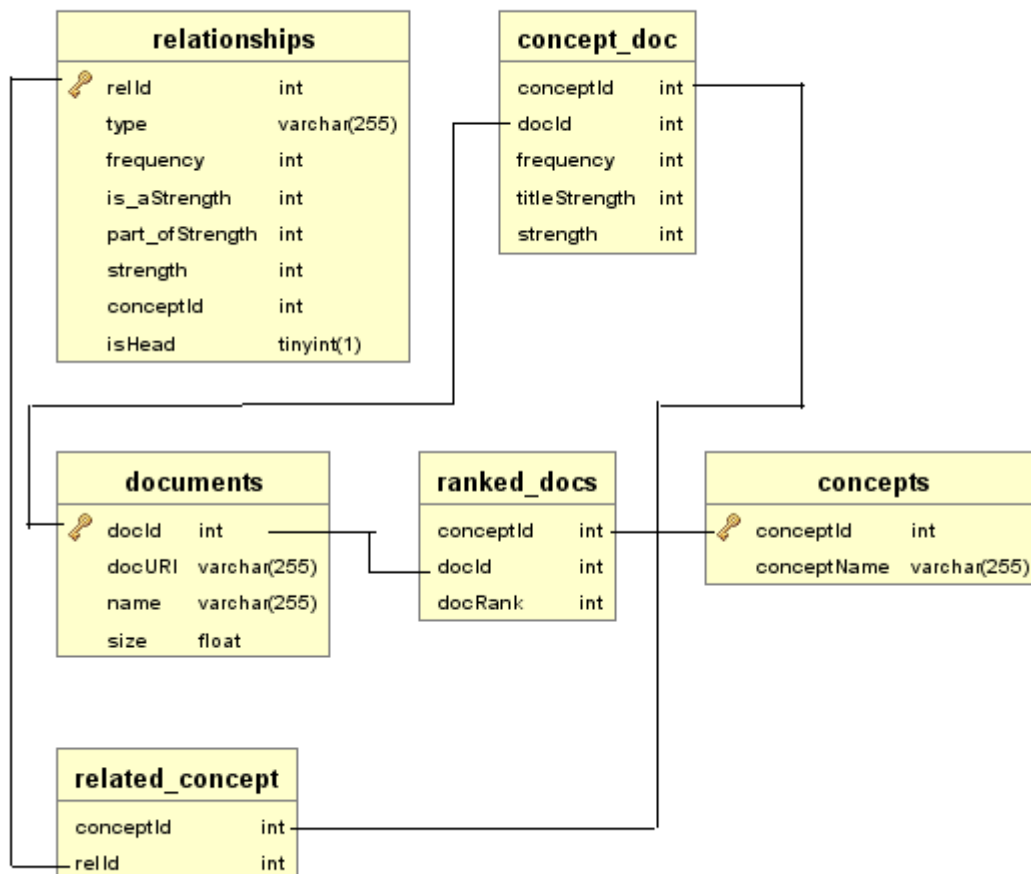


Figure 35 ER diagram for the database

6. RESULTS AND ANALYSIS

6.1. Result comparison before/after resolving Anaphora.

Anaphora resolution is very helpful in gaining better output from the system. Since the input document is preprocessed and fed content sentence by sentences to the extractor. So the extractor doesn't have any knowledge about the pronouns like he, she, it and etc. But anaphora resolver replaces pronouns by its original proper noun, then the frequency of relevant concept (proper noun) correctly measured and it will be helpful for increasing the importance value of important concepts.

Following test sample is used to test above scenario.

Java is a programming language originally developed by James Gosling at Sun Microsystems released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities than either C or C++. Java application is typically compiled to bytecode. It can run on any Java Virtual Machine regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers 'write once, run anywhere', meaning that code that runs on one platform does not need to be recompiled to run on another. Java is as of 2012 one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users.

Frequency values of main concepts before resolving anaphora are,

Concept	Frequency
Java	3
Programming language	2
Java Virtual Machine	1
Java application	1
Sun Microsystems	1

Result after resolving anaphora,

Java is a programming language originally developed by James Gosling at Sun Microsystems released in 1995 as a core component of Sun Microsystems' Java platform. a programming language derives much of its syntax from C and C++++ but has a simpler object model and fewer low-level facilities than either C or C++. Java application is typically compiled to bytecode. Java application can run on any Java Virtual Machine regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. Java is intended to let application developers 'write once, run anywhere', meaning that code that runs on one platform does not need to be recompiled to run on another. Java is as of 2012 one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users.

Concept	Frequency
Java	4
Programming language	3
Java application	2
Java Virtual Machine	1
Sun Microsystems	1

According to the result shown in tables, there is a considerable amount of improvement in frequency and final importance value also increased.

6.2. Result comparison before/after optimizing result.

Optimizing module has two main sub modules; they are concept optimizer and relationship optimizer. In concept optimizer, first remove the redundancies by converting all the concepts using morphological root.

Ex.

- Before optimizing - ‘object-oriented programming languages’ and ‘object-oriented programming language’ (plural/singular) considered as different concepts.
- After optimizing – only ‘object-oriented programming language’ exists and relationships of both above concepts are joined

In the optimization phase a strength value is assigned for each concept.

- Before optimizing – Only frequency value is considered
- After optimizing – Frequency value and title strength value are considered (according to the existence of concept in titles)

Relationship optimization is performed in reducing concepts; here if same relationship between same concepts exists then they are joined by considering type and strength of those relationships. Also relationships are modified with strength value using some assertion methods.

Java

Java is a programming language. PHP, C++ and C# are some of other famous programming languages. Java application can run on any Java Virtual Machine regardless of computer architecture. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. Java is intended to let application developers 'write once, run anywhere', meaning that code that runs on one platform does not need to be recompiled to run on another. Java is as of 2012 one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users.

By processing above sample doc with and without optimization, results can be obtained as follows.

Without optimization

conceptName	frequency
Java	4
client-server web applications	1
C + + and C #	1
general-purpose , concurrent , class-based	1
Java application	1
Java Virtual Machine	1
computer architecture	1
other famous programming languages	1
application developers	1
programming languages	1
platform	1
programming language	1

Here both ‘programming language’ and ‘programming languages’ exist. Only frequency value exists for filtering concepts.

conceptName	frequency	type	RelatedConisHead	RelatedConcept
general-purpose , concurrent , class-based	1	is a	1	Java
programming language	1	is a	1	Java
Java	1	is a	0	general-purpose , concurrent , class-based
Java	1	is a	0	programming language
Java Virtual Machine	1	aso	1	Java application
Java Virtual Machine	1	aso	1	computer architecture
Java application	1	aso	1	Java Virtual Machine
Java application	1	aso	1	computer architecture
general-purpose , concurrent , class-based	1	aso	1	Java
C + + and C #	1	aso	1	other famous programming languages
client-server web applications	1	aso	1	programming languages
client-server web applications	1	aso	1	Java
programming language	1	aso	1	Java
platform	1	aso	1	application developers
platform	1	aso	1	Java
programming languages	1	aso	1	client-server web applications
programming languages	1	aso	1	Java

Also both ‘programming language’ and ‘programming languages’ has relationships with ‘java’

With Optimization

conceptName	frequency	titleStrength	strength
java	4	4	0.571429
programming language	2	0	0.285714
client-server web applications	1	0	0.142857
java application	1	0	0.142857
general-purpose , concurrent , class-based	1	0	0.142857
java virtual machine	1	0	0.142857
computer architecture	1	0	0.142857
other famous programming languages	1	0	0.142857
application developers	1	0	0.142857
c + + and c #	1	0	0.142857
platform	1	0	0.142857
few implementation dependencies	1	0	0.142857

Here redundancies are removed. Both ‘programming language’ and ‘programming languages’ are joined and frequency added together. Here ‘java’ concept is in a title so it has highest importance value then ‘programming language’ because its frequency is 2 other concepts has even more less value as they have less frequencies.

conceptName	frequency	type	RelatedConisHead	RelatedConcept
platform	1	aso	1	application developers
platform	1	aso	1	java
c + + and c #	1	aso	1	other famous programming languages
application developers	1	aso	1	platform
application developers	1	aso	1	java
other famous programming languages	1	aso	1	c + + and c #
computer architecture	1	aso	1	java application
computer architecture	1	aso	1	java virtual machine
java virtual machine	1	aso	1	java application
java virtual machine	1	aso	1	computer architecture
general-purpose , concurrent , class-based	1	aso	1	few implementation dependencies
general-purpose , concurrent , class-based	1	aso	1	java
java application	1	aso	1	computer architecture
java application	1	aso	1	java virtual machine
client-server web applications	1	aso	1	programming language
client-server web applications	1	aso	1	java
programming language	1	aso	1	client-server web applications
programming language	2	aso	1	java

Here relationships of ‘programming language’ and ‘programming languages’ with ‘java’ are joined together and frequencies are added together.

6.3. Results of Concept Ranking

Document containing following three sentences are used for the illustration of the results obtained by the concept ranking.

- Java is a programming language.
- Java was developed by Sun.
- Sun developed a programming language.

The concepts found in the document and there statistics and calculated importance are given in the following table.

Concept	Concept Frequency	Related concepts	importance	Rank
Java	2	Programming language ,	0.41532397	1

		Sun		
Programming language	2	Java, Sun	0.4017329	2
Sun	2	Java, programming language	0.1829432	3

Here all the concepts have the same frequency and each have relationship with other two concepts but “Java” concept gets higher importance due to its relationship with “programming language” being an “is-a” relationship and being the head of that relationships. The concept “programming language” has lower importance than java due to the reason of being the tail of “is-a” relationship with “java”. The concept “sun” has the lowest importance because it only participates in associations with two other concepts.

7. DISCUSSION

7.1. Project Management

Managing time and resources was critical to the SigmaC project in order to meet the functional requirements of the project which were described in the SRS document. Project duration was from end of the December 2011 from end of the August 2012. It was difficult maintain the continuous development of the software solution and communication with all the supervisors due to few interruption with the academic work and other related matters. But with the proper management of the time, work and resources with the instructions from the supervisors the project was able to successfully completed within the allocated time frame.

Following aspects help the project to become a successful one.

- **Development Process**

Initially the development of the SigmaC was planned to use the agile development methodology. However the sprints were extended due to some interruptions. A back log was created according to the functional requirements. Meetings with external and internal supervisors modified some requirements mentioned in the back log.

- **Project Planning**

The main tasks were identified in the discussion meetings and allocated a time frame to complete those. Those tasks were split in to subtasks and distribute the workload equally within the project group members.

- **Team Collaboration**

Frequent communication between team members and also with the supervisors was maintained throughout the period. Both direct and online communication was there to discuss the project related matters. Skype, Google Chat and mailing lists were the online facilities that used during the development period.

- **Project Distribution**

SigmaC codebase and binary distribution files were currently hosted in Google Code as opensource software. More details and manuals were in the project web site

Code Base: <http://code.google.com/p/sigmac/>

Project Web Site <http://sigmac.sldigits.com>

- **Coding Standards**

Commonly accepted object oriented coding standards were maintained.

- **Version Control System**

Sub versioning system was required since the four members were working with the different components which were interdependent. SVN was used to achieve this where the project code was maintained online at <http://code.google.com/p/sigmac/> with frequent commits via Tortoise SVN client.

7.3. Project Outcomes

Following outcomes with regards to the project were identified

1. Novel way to organize enterprise knowledge base using concept maps
2. Accurate Rule based concept and relationship extraction methodology
3. Ontological data source generation
4. Document summaries based on concept maps

Novel way to organize enterprise knowledge base using concept maps

SIGMAC organizes the enterprise knowledge bases based on concept maps which are considered as a more human understandable compact representation. The usage of concept maps provides several advantages, they are,

- Concept level browsing of document
- Identification of relationships between concepts
- Compact conceptual summaries of documents

SIGMAC allows any number of predefined set of relationships to be added to the system via set of rules without any modification to the system. Therefore SIGMAC allows any organization to expand the system to suit their unique requirements.

Accurate Rule based concept and relationship extraction methodology

SIGMAC allows any number of predefined set of relationships to be added to the system via set of rules without any modification to the system. Therefore SIGMAC allows any organization to expand the system to suit their unique requirements.

Rules of the system are defined in such a way that the set of concepts identified is dependent on the type of sentence (type of relationship that the sentence is describing). These types of rules have given us more accurate results from each individual sentence. This is because concepts present are dependent on the relationship being described, but the generic set of rules would not identify these concepts accurately.

Ontological data source generation

Development of ontological can also be accomplished using the SIGMAC. The input document set to the system should accurately cover all aspects of the ontology and the relationships that are not required need to be eliminated from the default set of relationships provided with SIGMAC.

Document summaries based on concept maps

SIGMAC allows users to view concept maps for individual documents. These concept maps effectively provide a conceptual summary of the document. Furthermore users can expand and minimize the summary allowing them to change the level of details present in the document summary. This advantage also comes due to concept map representation used by SIGMAC.

8. CONCLUSION & FUTURE WORK

8.1. Conclusion

The SigmaC project suggest an elegant and novel solution for the enterprise knowledge retrieval by using concept maps which are constructed using advance Natural Language Processing methods and related technologies. Furthermore the project also provides an automated solution for the creation of ontological data sources saving a vast amount of human labor. Concept map based document summaries provided by SigmaC provide machine understandable and human friendly representation of knowledge embedded in documents. Patterns which are based on the type of relationship described by the sentence results in more accurate identification of concepts than those can be identified using patterns which are not based on the type of relationship described by the sentence.

8.2. Future Works

- **Different roles for different security levels**

Since the system supports user editable concept map feature and due to the confidentiality of documents, system should have different access capability levels assigned to individual users. It can be introduced as a future enchantment.

- **Supporting multiple languages**

The System will be developed only to support documents which are written in English language. This can be extended to support other languages.

- **3D concept map visualization**

The System will be developed only to support 2D concept map visualization and it can be extended to support 3D concept map visualization for better browsing experience.

- **Rule learning Engine**

Rules which are identifying concepts and relationships are developed manually. This process can be automated using a rule learning engine.

- **Extend to analyze internet**

The system is developed to analyze enterprise corpuses. This process can be extended to analyze online resources

9. SIGMAC EXTENSION-RECRUITMENT HELPER

9.1. Introduction of SigmaC Extension of Recruitment Helper

Sorting up the CV's according to the relevancy for a particular vacancy in an enterprise is a tedious task for the people who are handling the recruiting aspects. In this project, It's able to solve the above problem by giving the in depth, normalized review of the CV by analyzing the CV, considering the technologies, certification and reference people and binding those using the internet.

9.2. Objectives of the Recruitment Helper

- **Extract CVs and sorting them**

Sorting is done by analyzing each CV, considering the technologies, certification and reference people and binding those using the internet.

9.3. Literature Review of Recruitment Helper

9.3.1.Name Extraction

IBM Research Report- Extracting Names from Natural-Language Text

- **The name extraction procedure**

Nominator forms a candidate name list by scanning the tokenized document and collecting all Sequences of capitalized tokens (or words) and selected prepositions. [15]

The candidate list extracted for the sample document contains:

District of Columbia Bar
D.C. Court of Appeals
ABA
Robert Jordan
Steptoe & Johnson

In a move that would represent a major break with tradition in the legal profession, law firms in this city may become the first in the nation to reward non-lawyers with the cherished title of Partner. The District of Columbia Bar has recommended adoption of a new code of ethics that would permit such a move, reflecting broad changes in the kind of legal services offered by the city's firms. The new ethics code must be approved by the D.C. Court of Appeals before it goes into effect and will apply only to lawyers and law firms that practice here. The professional conduct of lawyers in other jurisdictions is guided by American Bar Association rules or by state bar ethics codes, none of which permit non-lawyers to be partners in law firms.

ABA
Washington
Dubuque

...

Mr. Jordan of Steptoe & Johnson

Each candidate name is then examined to detect if it contains several independent names, and if so, split into as many names as are found in it. This is a challenging task, since the internal Structure of such name combinations can be very complex. In this sample

Mr. Jordan of Steptoe & Johnson is split into

[Mr. Jordan]

[and]

[Steptoe & Johnson]

.
Without recourse to semantics or world knowledge, it does not always have sufficient evidence. In such cases we prefer to err on the conservative side and not split. This explains the presence of "names" such as "American Television & Communications and Houston Industries Inc." or "Dallas's MCorp and First RepublicBank and Houston's First City Bancorp. of Texas" in this results. , "August" or "May" is collected as a candidate name since it is ambiguous between a date and a person name. But if no other name is found to contain it as a substring (e.g., "May Smith"), it is safe to assume that the singleton is a date and delete it from the list. Finally, Nominator groups all variants referring to the same entity in a class, and either chooses one of the variants as the canonical name or constructs a canonical name from components of different variants. This grouping is done together with assigning a type that categorizes the entity referred to by the class, as these two tasks go hand in hand. The result of this step for the sample document is shown below. Each canonical name is followed by its type and then by any additional variants.

District of Columbia Bar >>ORG

D.C. Court of Appeals >>ORG

American Bar Association >>ORG : ABA

Steptoe & Johnson >>ORG

Washington >>PLACE

Dubuque >>PLACE

Robert Jordan >>PERSON : Mr. Jordan

9.3.2.Sentence Boundary detection

Experiments on Sentence Boundary Detection

Memory-based learning, also known as case based and lazy learning, operates by memorizing a set of training examples and categorizing new cases by assigning them the class of the most similar learned example. Here apply this methodology to the sentence boundary detection task by presenting Timbl with examples of word boundaries from a training text, each of which is categorized as either sentence_boundary or no_boundary. Unseen examples are then compared and categorized with the class of the most similar example. These texts are reliably part of speech tagged and sentence boundaries can be easily derived from the corpus. This text was initially altered so as to remove all punctuation and map all characters into upper case. 90% of the corpus, containing 965 sentence breaks, was used as a training corpus with the remainder, which contained 107 sentence breaks, being

held-back as unseen test data. The first stage was to extract some statistics from the training corpus. Here examined the training corpus and computed, for each word in the text, the probability that it started a sentence and the probability that it ended a sentence. In addition, for each part of speech tag we also computed the probability that it is assigned to the first word in a sentence and the probability that it is assigned to the last word. Each word boundary in the corpus was translated to a feature-vector representation consisting of 13 elements, shown in Following Table[16]

Position	Feature
1	Preceding word
2	Probability preceding word ends a sentence
3	Part of speech tag assigned to preceding word
4	Probability that part of speech tag (feature 3) is assigned to last word in a sentence
5	Flag indicating whether preceding word is a stop word
6	Flag indicating whether preceding word is capitalised
7	Following word
8	Probability following word begins a sentence
9	Part of speech tag assigned to following word
10	Probability that part of speech (feature 9) is assigned to first word in a sentence
11	Flag indicating whether following word is a stop word
12	Flag indicating whether following word is capitalised word
13	sentence_boundary or no_boundary

Both precision and recall are quite promising under these conditions. However, this text is different from ASR text in one important way: the text is mixed case. The experimented was repeated with capitalization information removed; that is, features 6 and 12 were removed from the feature-vectors. The results formed this experiment, are shown in the bottom row of Table 3. It can be seen that the recorded performance is far lower when capitulation information is not used, indicating that this is an important feature for the task. These experiments have shown that it is much easier to add sentence boundary information to mixed case test, which is essentially standard text with punctuation removed, than ASR text, even as summing a zero word error rate.

Case information	P	R	F
Applied	78	75	76
Not applied	36	35	35

Figure 36 Results of the sentence boundary detection

9.3.3.Entity Extraction

Alchemy API

AlchemyAPI is capable of identifying people, companies, organizations, cities, geographic features, and other typed entities within your HTML, text, or web-based content. API calls to Alchemy API can be done after registered with Alchemy API. The output is a XML file and contents can be easily extracted from XML.

9.4. Recruitment Helper Design

9.4.1.CV Extraction

9.4.2.Name Extractor

Name is extracted using “Name” tag or the first line of the CV. Following display the java of name extractor.

```
public class NameExtractor {
    String text;
    public NameExtractor(String text){
        this.text=text;
    }

    public String Analyze() throws FileNotFoundException, IOException{
        String name = "";
        String newtext=compressNewLines(text);
        // System.out.println(newtext);
        BufferedReader reader = new BufferedReader(new StringReader(newtext));
        StringBuffer out = new StringBuffer();
        String s = null;
        if(newtext.contains("Name")){
            while((s = reader.readLine())!=null){
                if(s.contains("Name")){
                    String[] split = s.split(":");
                    if(split.length>1){
                        name=split[1];
                    }
                    else{
                        name= CheckName(s, reader);
                    }
                }
            }
        }
        else{
            name= CheckName(s, reader);
        }

        return name;
    }
}
```

9.5. Email Extractor

Email is extracted using regex matching. Java method is as follows.

```
public void Analyze(String text){
    Pattern p = Pattern.compile("[" + "[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*@[A-Za-z0-9-]+(" +
    + "[\\._A-Za-z0-9-]+)*\\.([A-Za-z]{2,})" + ")");
    Matcher m = p.matcher(text);

    while(m.find()) {
        emails.add(m.group());
    }
}
```

9.6. Education Extractor

This class extract whether applicant passed in A/L, O/L or not. It also extracts degree if applicant mentioned in CV. Following displayed part of that method.

```
public void Analyze(HashMap<String, String> cvDetails,String text) throws IOException {
    Set<String> topics=cvDetails.keySet();
    Object[] topicNames=topics.toArray();

    String key=null;
    String relevantContent=null;
    for(int i=0;i<topicNames.length;i++){
        String tagName=topicNames[i].toString();
        if(tagName.toLowerCase().contains("academic") ||
            tagName.toLowerCase().contains("educational")
            ||tagName.toLowerCase().contains("education")){
            key=tagName;
            break;
        }
    }
    if(key!=null){
        relevantContent=cvDetails.get(key);
    }
    else{
        relevantContent=text;
        relevantContent = relevantContent.substring(
            0, relevantContent.length() - relevantContent.length()/4);
    }

    BufferedReader reader = new BufferedReader(new StringReader(relevantContent));
    String s = null;
    while((s = reader.readLine())!=null){
        if(s.toLowerCase().contains(
            "bsc") || s.toLowerCase().contains("diploma")
```

9.7. Professional Qualification Extractor

This class extracts the professional qualifications like SCJP, CCNA which applicant mentioned in CV. It is used pre-defined set of qualification list as a identifier for pattern matcher. So test_pq.txt contains pre-defined qualifications and retrieve those and pass those into pattern matcher to identify relevant professional qualifications. The pattern matcher can identify the qualifications which are not in pre-defined qualification list.

```
public void Analyze(HashMap<String, String> cvDetails,String text,LexicalizedParser tlp) throws IOException {
    this.lp=tlp;
    Set<String> topics=cvDetails.keySet();
    Object[] topicNames=topics.toArray();
    String key=null;
    String relevantContent=null;
    for(int i=0;i<topicNames.length;i++){
        String tagName=topicNames[i].toString();
        if(tagName.toLowerCase().contains("professional qualification")){
            key=tagName;
            break;
        }
    }
    if(key!=null){
        String temp=compressNewLines(cvDetails.get(key));
        FileWriter fw = new FileWriter("test_pq.txt");
        fw.write(temp);
        fw.close();

        DocumentPreprocessor p=new DocumentPreprocessor("test_pq.txt");
        int count=0;
        ArrayList<String> patterns=new ArrayList<String>();
        String regex2="NP<, (NP=course$+ (PP<, (IN$+ NP=name)))";
        //pattern for identification Professional qualifications
        patterns.add(regex2);
        for (List<HasWord> sentence : p){
            Tree parse = lp.apply(sentence);
            for(int i=0;i<patterns.size();i++){
                TregexPattern pattern=TregexPattern.compile(patterns.get(i));
                TregexMatcher matcher=pattern.matcher(parse);

                if(matcher.find()){
                    String courseType="";
                    String courseName="";
                    Tree type=matcher.getNode("course");
                    Tree name=matcher.getNode("name");
                    TregexPattern tr=TregexPattern.compile("NP : JJ=tmp"); //remove the jj part
                    TsurgeonPattern tsur=Tsurgeon.parseOperation("delete tmp");
                    type=Tsurgeon.processPattern(tr,tsur, type);

                    List<LabeledScoredTreeNode> leaves=type.getLeaves();

                    for(LabeledScoredTreeNode leave : leaves){
                        courseType+=leave.nodeString().toLowerCase()+" ";
                    }

                    TregexPattern trName=TregexPattern.compile("NP : DT=tmp"); // remove the DT part
                    TsurgeonPattern tsurName=Tsurgeon.parseOperation("delete tmp");
                    name=Tsurgeon.processPattern(trName,tsurName, name);
                    List<LabeledScoredTreeNode> leavesName=name.getLeaves();
                    for(LabeledScoredTreeNode leave : leavesName){
                        courseName+=leave.nodeString()+" ";
                    }

                    Set CourseSet=ProfQualificatins.keySet();
                    if(CourseSet.contains(courseType)){
                        ArrayList<String> past=ProfQualificatins.get(courseType);
                    }
                }
            }
        }
    }
}
```

9.8. Reference Extractor

Alchemy API is used to identify the reference.

9.9. Technology Extractor

Technology extractor is the heart of Recruitment Helper project and following are the set of techniques used to extract technologies.

- Use pre-defined technology list
- Use Alchemy API to retrieve technologies
- Pattern matching in order to identify more technologies

```
public ArrayList<String> getTechnologies(HashMap<String, String> map, LexicalizedParser lp, String text) throws IOException{
    EntityTest et=new EntityTest();
    et.analyze(text);
    ArrayList<String> Tech=et.getTechnologies();
    ArrayList<String> FieldTerminology=et.getFieldTerminology();
    tempTech=getTempTech();
    tempTech.addAll(Tech);
    FileWriter fw=new FileWriter("technology.tff");//tempory file format
    fw.write(compressNewLines(text));
    fw.close();
    DocumentPreprocessor p=new DocumentPreprocessor("technology.tff");

    String tregex="NP!<< NP";
    for (List<HasWord> sentence : p){
        Tree parse = lp.apply(sentence);

        TregexPattern pattern=TregexPattern.compile(tregex);
        TregexMatcher matcher=pattern.matcher(parse);
        while(matcher.find()){
            Tree temp=matcher.getMatch();
            TregexPattern trName=TregexPattern.compile("NP : DT=tmp"); // remove the DT part
            TsurgeonPattern tsurName=Tsurgeon.parseOperation("delete tmp");
            temp=Tsurgeon.processPattern(trName,tsurName, temp);
            List leaves=temp.getLeaves();

            String name="";
            for(Object leave : leaves){
                name+=leave.toString().replaceAll("-LRB-", " ").replaceAll("-RRB-", " ")+" ";
            }
            for(int k=0;k<tempTech.size();k++){
                if(name.toLowerCase().contains(tempTech.get(k).toLowerCase())){
                    String TechArray[]=name.split(",");
                    for(int l=0;l<TechArray.length;l++){
                        if(FinalTech.contains(TechArray[l].replaceAll("\\.", "").trim().toLowerCase()) || TechArray[l]==""){
                            //Already in list
                        }
                        else if (TechArray[l].replaceAll("\\.", "").trim().length()>2){
                            FinalTech.add(TechArray[l].replaceAll("\\.", "").trim().toLowerCase());
                        }
                    }
                }
            }
        }
    }
}
```

9.10. Sample User Map after extracting Entities.

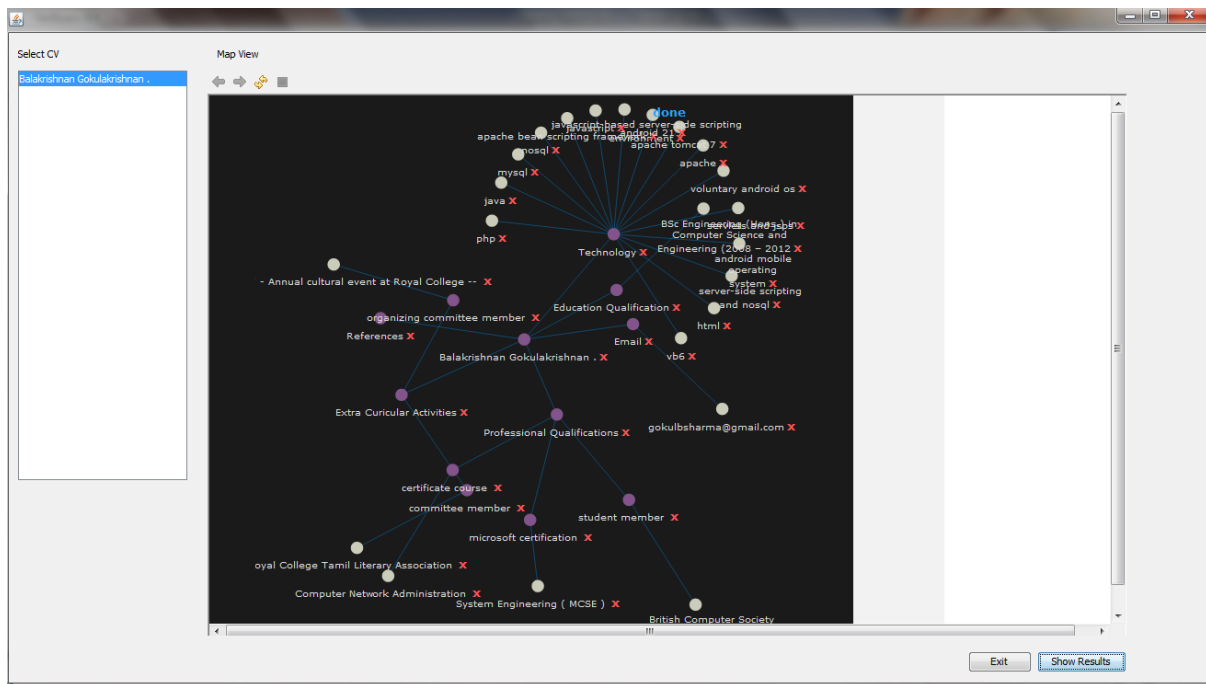


Figure 38

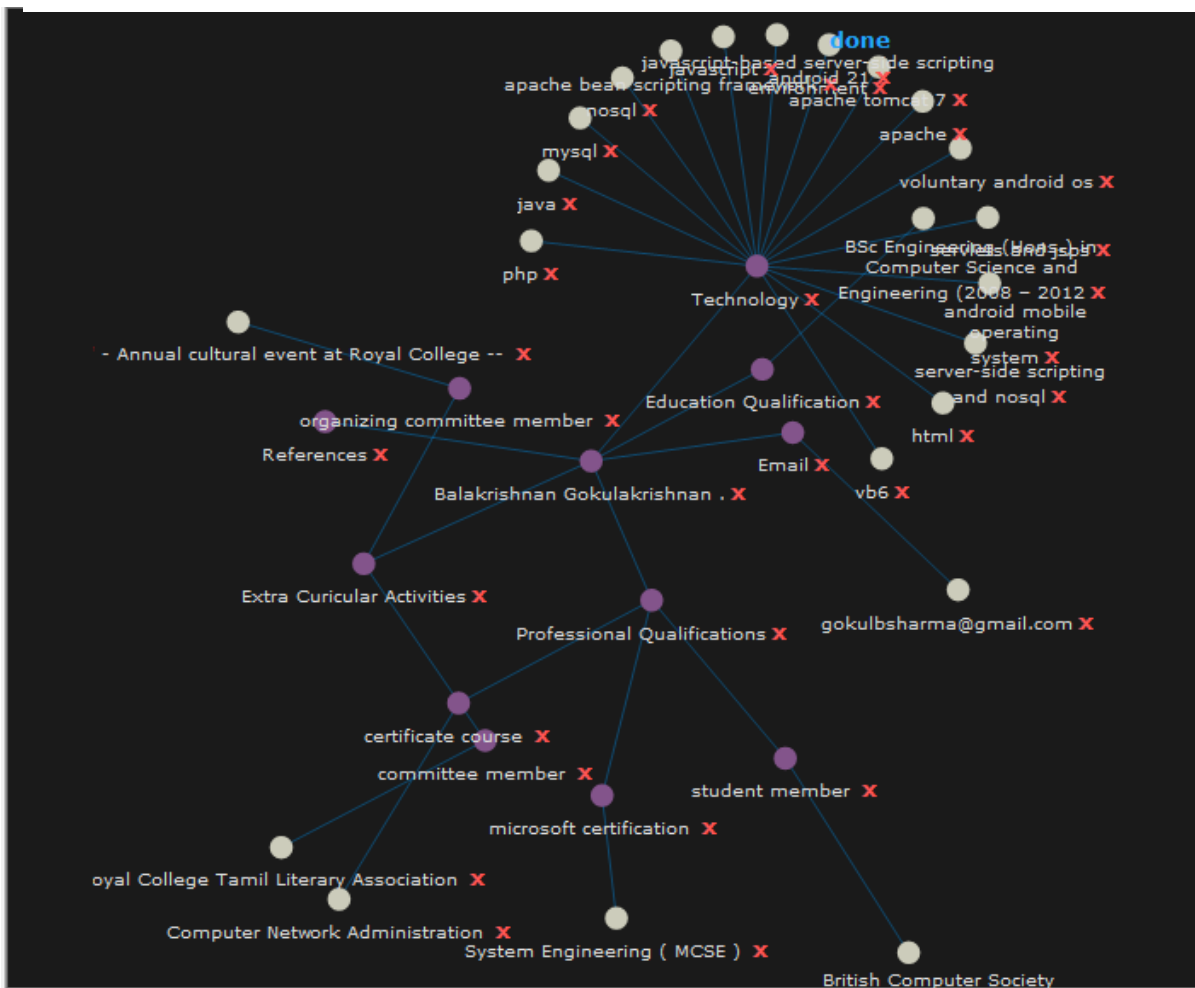


Figure 37

9.11. Information retrieval through internet

After extracting information from CV next process is to identify following information from internet.

- Applicant familiar technologies
- Applicant's Professional Qualifications
- Referees research Areas
- Referees familiar technologies

Google Search, Alchemy API and LinkedIn API use for retrieving above information from internet. There should be a weighted mechanism to identify whether applicant how much knows retrieval technologies.

For an example let's take some applicant is familiar with Java, Hibernate, spring and he has qualifications of SCJP and his referees also familiar with Java. If we try to weight the java technology all of the above should be considered. The reason is, if he knows Hibernate and spring, he should know Java as well. If he followed SCJP he should know java. For doing that we should know spring, hibernate, SCJP are related to Java. This process is also done using internet. Google Search and Alchemy API is used to identify these sentries.

Frequency is used as the weight of the technology and user can defined it as follows.

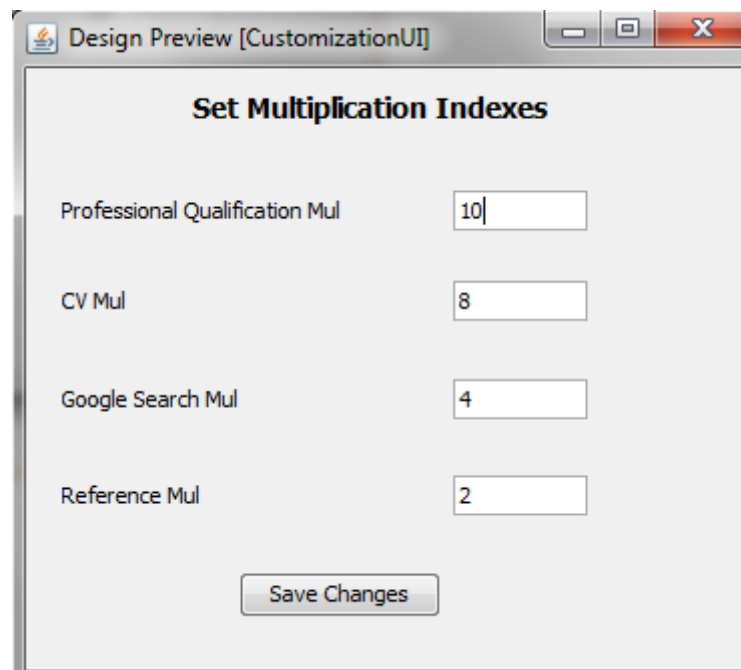


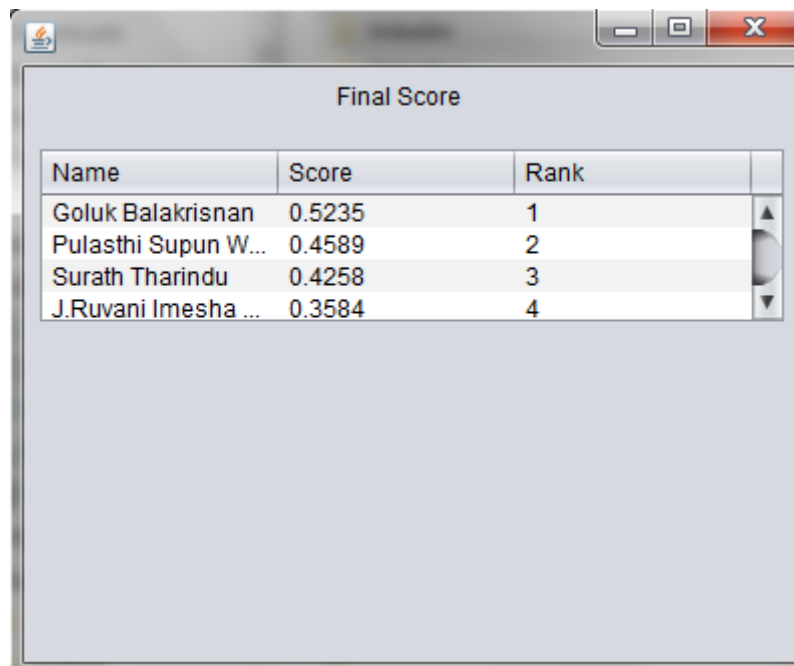
Figure 39

The final weight is multiplexer of each term frequency with relevant weightage.

After that we have set of weighted technology list. Now it should be compared with the SigmaC repository concept map and calculate a score for each CV. In repository concept map each concept

has its own importance value. For mapping is done as multiplier of technologies weight and relevant concept (This should have name same as technology name) importance value and normalized all.

Final Step is to rank CVs based on the score. User can see the final rank by clicking show score button. It popup a window it shows ranks as follows.



The screenshot shows a window titled 'Final Score' with a table containing the following data:

Name	Score	Rank
Goluk Balakrisnan	0.5235	1
Pulasthi Supun W...	0.4589	2
Surath Tharindu	0.4258	3
J.Ruvani Imesha ...	0.3584	4

Figure 40

9.12. Future Works of Recruitment Helper

- Use Education Qualifications like Degree, Exams, GPA for calculating Score
- Rule learning engine for pattern identification for technology extraction

ABBREVIATIONS & ACRONYMS

ACE-Automatic Content Extraction

DBMS – Database Management Systems

GUI – Graphical User Interface

LEXAS -Lexical Ambiguity Resolving System

MUC-Message Understanding Conference

MRD-machine readable dictionaries

NLP- Natural Language Processing

POS- Part of Speech

RelEx - Relationship Extractor – A system developed under OpenCog Project

SigmaC- Sum of Concepts (Greek Letter Sigma)

WSD- Word Sense Disambiguation

WordNet/JWNL - Java WordNet Library

REFERENCES

- [1] G. A. Miller, "WordNet: a lexical database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
- [2] D. D. Palmer and M. A. Hearst, "Adaptive sentence boundary disambiguation," 1994, p. 78.
- [3] F. Smadja, "Xtract: An overview," *Computers and the Humanities*, vol. 26, no. 5, pp. 399–413, Dec. 1992.
- [4] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, Stroudsburg, PA, USA, 2003, pp. 173–180.
- [5] P. Buitelaar and T. Eigner, *Topic Extraction from Scientific Literature for Competency Management*. .
- [6] V. Seretan, "Induction of syntactic collocation patterns from generic syntactic relations," in *Proceedings of the 19th international joint conference on Artificial intelligence*, San Francisco, CA, USA, 2005, pp. 1698–1699.
- [7] C. D. Manning and H. Schuetze, *Foundations of Statistical Natural Language Processing*, 1st ed. The MIT Press, 1999.
- [8] D. Gildea and D. Jurafsky, "Automatic labeling of semantic roles," *Comput. Linguist.*, vol. 28, no. 3, pp. 245–288, Sep. 2002.
- [9] B. Gelfand, M. Wulfekuler, and W. F. Punch, "Automated concept extraction from plain text," in *AAAI 1998 Workshop on Text Categorization*, 1998, pp. 13–17.
- [10] B. Hachey, "Towards generic relation extraction," 2009.
- [11] J. G. Conrad and M. H. Utt, "A system for discovering relationships by feature extraction from text databases," in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, 1994, pp. 260–270.
- [12] T. Hasegawa, S. Sekine, and R. Grishman, "Discovering relations among named entities from large corpora," in *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, 2004, p. 415.
- [13] M. A. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *Proceedings of the 14th conference on Computational linguistics-Volume 2*, 1992, pp. 539–545.
- [14] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, Stroudsburg, PA, USA, 2003, pp. 423–430.