Spring 2009

Text summarization using concept hierarchy

Xiaomei Huang

TEXT SUMMARIZATION USING CONCEPT HIERARCHY

by

Xiaomei Huang, M.S.

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

May 2009

LOUISIANA TECH UNIVERSITY

THE GRADUATE SCHOOL

11/07/2008

Date

We hereby recommend that the dissertation prepared under our supervision by Xiaomei Huang

entitled Text Summarization Using Concept Hierarchy

be accepted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computational Analysis and Modeling

Supervisor of Dissertation Research

Head of Department

Department

Recommendation concurred in:

Advisory Committee

Approved:

Director of Graduate Studies

Approved:

Dean of the Graduate School

Dean of the College

GS Form 13a
(6/07)

ABSTRACT

This dissertation aims to create new sentences to summarize text documents. In addition to generating new sentences, this project also generates new concepts and extracts key sentences to summarize documents. This project is the first research work that can generate new key concepts and can create new sentences to summarize documents.

Automatic document summarization is the process of creating a condensed version of the document. The condensed version extracts the key contents from the original document. Most related research uses statistical methods that generate a summary based on word distribution in the document. In this dissertation, we create a summary based on concept distributions and concept hierarchies. We use Stanford parser as our syntax parser and ResearchCyc (Cyc) as our knowledge base. Words and phrases of a document are mapped into Cyc concepts. We introduce a unique concept propagation method to generate abstract concepts and use those abstract concepts for the summarization. This method has two advantages over the existing methods. One advantage is the use of multi-level upward propagation to solve the word sense disambiguation problem. The other is that the propagation process provides a method to produce generalized concepts.

In the first part of the project, we generate a summary by extracting key concepts and key sentences from documents. We use Stanford parser to segment a document to

sentences and to parse each sentence to words or phrases tagged with their part-of-speeches. We use Cyc commands to map those words and phrases to their corresponding Cyc concepts and increase the weights of those concepts. To handle word sense disambiguation and to create summarized concepts, we propagate the weight of the concepts upward along the Cyc concept hierarchy. Then, we extract the concepts with some of the highest weights to be the key concepts. To extract key sentences from the document, we weigh each sentence in the document based on the concept weight associated with the sentence. Then, we extract the sentences with some of the highest weights to summarize the document.

In the second part of the project, we generate new sentences to summarize a document based on the generalized concepts. First, we extract the subject, predicate, and object from each sentence. Then, we create compatible matrices based on the compatibility between the subjects, predicates, and objects among sentences. Two terms are considered to be compatible if the following three conditions hold: the two terms are the same concept, one concept is the other concept's immediate super class, or two concepts share the same immediate super class. From the compatible matrices, we build compatible clusters and finally generate new sentences for each compatible cluster. These newly generated sentences serve as a summary for the document.

We have implemented and tested our approaches. The test results show that our approaches are viable and have great potential for future research.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any or all portions of this Dissertation. It is understood that "proper request" consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Dissertation. Further, any portions of the Dissertation used in books, papers, and other works must be appropriately referenced to this Dissertation.

Finally, the author of this Dissertation reserves the right to publish freely, in the literature, at any time, any or all portions of this Dissertation.

Author Xiaonei Muang

Date 05/16/2009

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

First and foremost I want to thank my advisor, Dr. Ben Choi for guiding me through the dissertation. He gave me the freedom to explore, and when I drifted he would always remind me "keep focus."

Second, I would like to thank my committee members Dr. Songming Hou, Dr. Don Liu and Dr. Christian Duncan. I would also like to thank all the members of FIT team, particularly Xingui Tang and Xiaogang Peng. Through the years in FIT group, we have learned from each other and helped each other.

Third, I would like to thank Mr. John De Oliveira from Cycorp for patiently answering my questions.

Fourth, I would like to thank the Center for Entrepreneurship and Information Technology of Louisiana Tech University for their financial support.

Finally, I would like thank my family. They have always had faith in me, loved me and encouraged me through the years. I dedicate this dissertation work to my family.

CHAPTER 1

INTRODUCTION

Automatic document summarization is the process of identifying the most significant information in a document or multiple documents and creating a condensed version of the document. The generated summary should cover the important content of the original documents.

In this information age, human knowledge has been growing in exponential speed. With the current World Wide Web's development, people can upload almost any document to the web. The documents on the web are accessible to anyone who connects to the Internet. This vast amount of information can be both advantageous and disadvantageous. The advantage is that anyone can gain access to this vast amount of information. The disadvantage is that the information is so vast that it becomes such a great difficulty to find the needed information. So there is an increasing demand for automatic document summarization. Automatic summarization can extract or abstract the most important content from the documents and present it to the users. This will greatly save the readers' time and effort in finding useful information.

In the first part of this project, we propose a concept hierarchy-based method for summarizing a document. The process has three major steps: (1) Map words or phrases from a document into Cyc concepts which are defined in ResearchCyc knowledge base. (2) Create general concepts by propagating weights of the concepts upward along the

concept hierarchy of the Cyc ontology and disambiguate senses by the weights of the candidate concepts. (3) Retrieve weight of each word or phrase of each sentence and sum the weight together, normalize the sum by the terms number of the sentence. (4) Sort the sentences according to their weights and retrieve the sentences with some of the highest weights to summarize the document.

In the second part of the project, we generate new sentences based on sentence compatibility. At first, we extract the subject, predicate and object triple from each sentence. Then we create compatible matrices based on the compatibility between the sentences' subjects, predicates and objects. Two terms are considered to be compatible if they have the same concept, or one concept is the other concept's immediate super class, or two concepts are siblings in the concept hierarchy. From the compatible matrices, we build compatible clusters. Then, we generate new sentences for each compatible cluster.

The approaches have been fully implemented and tested. The test results show that the approaches are viable and have great potential for future research.

The contributions of this project are as followings: (1) We are the first to use ResearchCyc as knowledgebase to create new sentences for automatic document summarization. ResearchCyc, started by Dr. Douglas Lenat in 1984, is currently the largest common sense knowledgebase and inference engine in the world. However, due to its lack of documentation, ResearchCyc has not been widely used. (2) We propose a concept weight propagation method to resolve word sense disambiguation problems. By propagating the concept weights upward along the Cyc concept hierarchy, we link the semantically related concepts together. The right concepts are enforced by each other so we can choose the right concept for each word or phrase. (3) We propose a new compatibility method to generate new sentences by finding compatible sentences,

clustering them together, and creating new sentence for each cluster. Figure 1.1 and 1.2 show part 1 and part 2 of the flow chart for our system. The flow chart shows how key concepts, key sentences, and new sentences are created.

Figure 1.3 shows the architecture of our summarization system. First, we use Stanford Parser to parse the document. Based the result from Stanford Parser, we map the words and phrases from the document to the Cyc concept hierarchy. Based on the concept hierarchy representation of the document, we extract key concepts and key sentences from the document to summarize the document, and then create new sentences to summarize the document.

The remainder of this dissertation is structured as follows. In Chapter 2, we introduce the background and related work in the automatic text summarization field. It gives a simple introduction to the Stanford Parser and ResearchCyc Knowledgebase. In Chapter 3, we present the framework of our text summarization system. It provides the implementation steps for generating summaries. In Chapter 4, we provide the experiments and results of creating new concepts, extracting key sentences, and creating new sentences. For the extraction-based summarization method, we compare the results of our text summarizer with Microsoft Autosummarizer system. Being the first for the others, we created our own test cases. In Chapter 5, we give the conclusion and outline the future research.

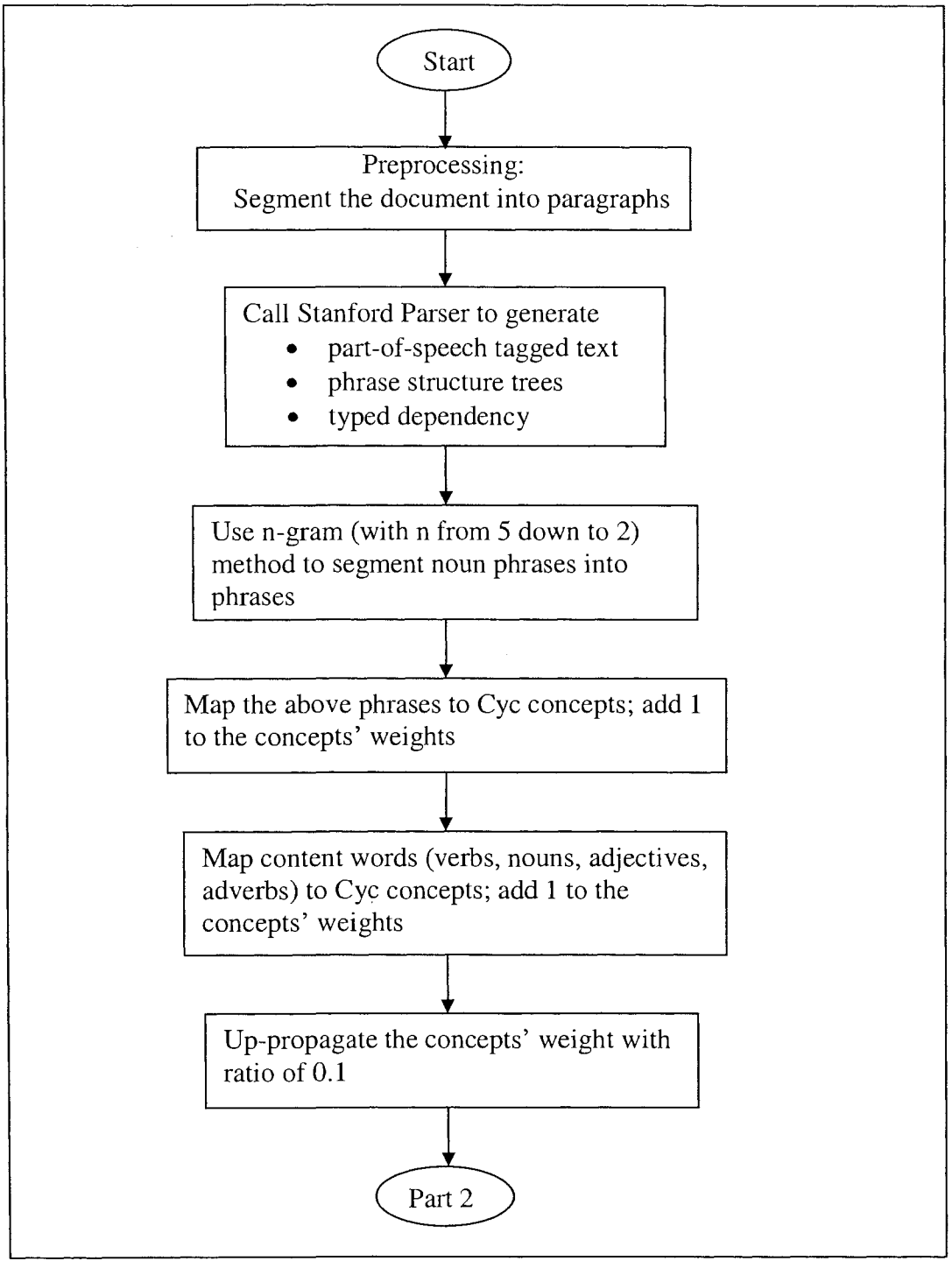


Figure 1.1 Flow Chart of Text Summarization System (Part 1).

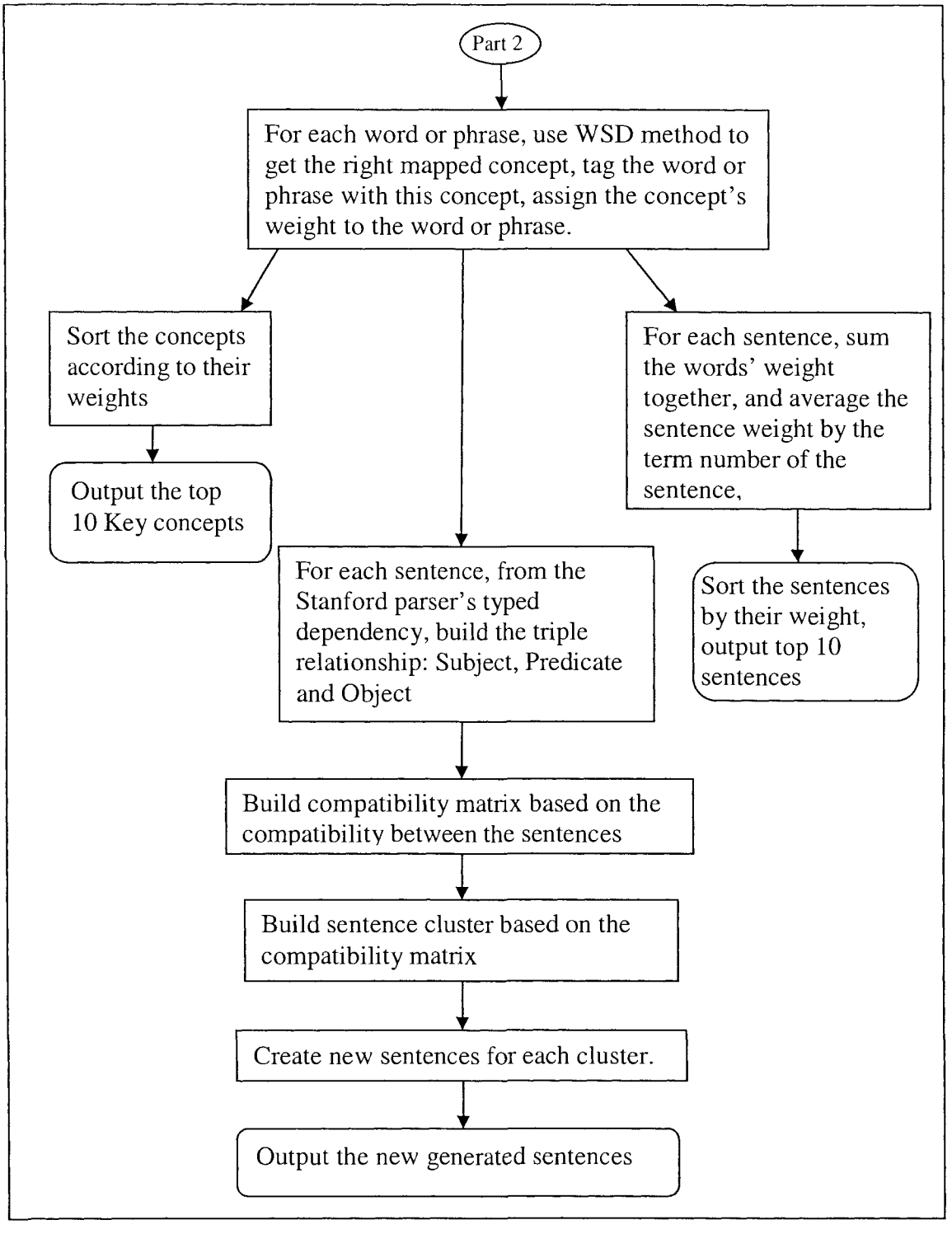


Figure 1.2 Flow Chart of Text Summarization System (Part 2).

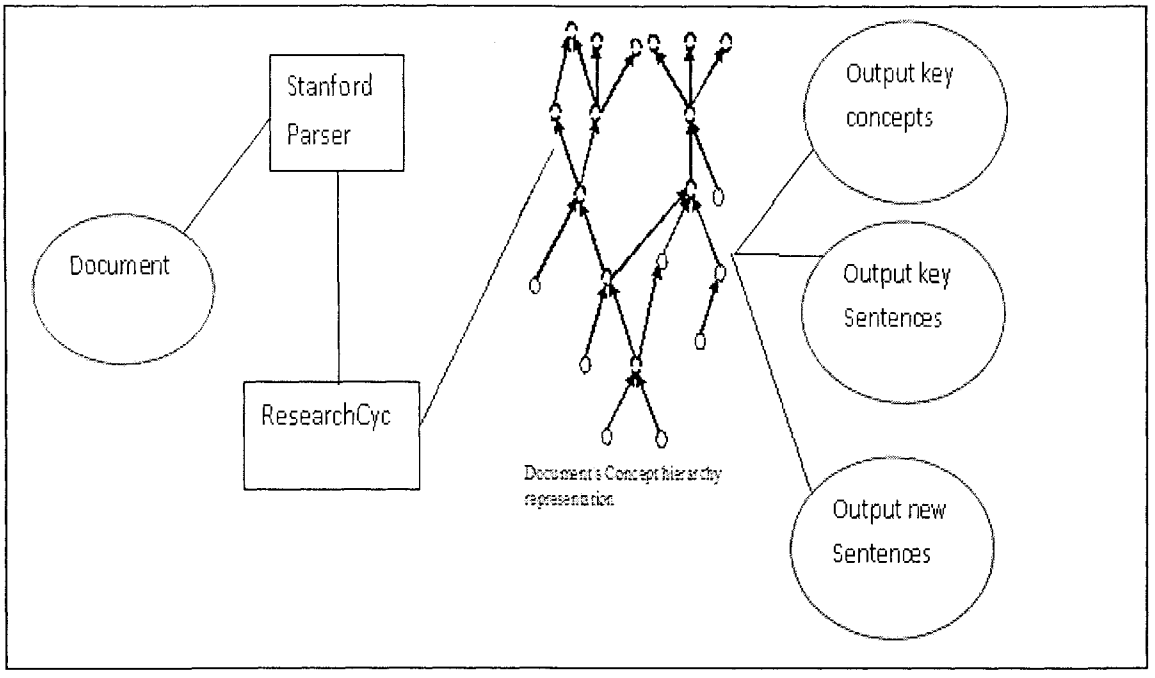


Figure 1.3 Text Summarization Model.

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter describes the related work and provides some background knowledge for document summarization. Since Luhn proposed the first automatic text summarization system based on word frequency, many approaches to text summarization have been developed (Luhn, 1958). Currently, two major text summarization methods are used in the area: keyword-based text summarization and sense-based text summarization. There are some drawbacks for keyword-based text summarization because of the following reasons: one word may have several different semantic meanings, different words can have the same semantic meaning, and a phrase can have totally different meanings from the words' literal meaning (Lin, 1998). Thus, the text summarization results of keyword-based method are not very accurate. Therefore, many researchers started using sense-based text summarization. Most of them use WordNet as the lexical database and try to map words and phrases to Wordnet senses and create summary based on the sense distribution. One drawback of WordNet is its fine-grained sense distinctions. It is often beyond what may be needed in many natural language processing applications. When trying to map a word to its corresponding sense in WordNet, it is very difficult to disambiguate between the senses.

2.1 The Genres of Text Summarization

There are different ways to categorize text summarization methods. According to summaries' focus, summaries can be either generic or user-focused. Generic summaries are generated based on the content of the document. It reflects the author's view, aiming at a broad reader group. User-focused summaries are generated based on a reader's query or a reader's profile. It reflects the reader's interest. According to the input source size, text summarization can be categorized as single-document summarization and multi-document summarization. Single-document summarization is based on only one input document. Multi-document summarization has multiple input documents that need to fuse together the information from each input document and present to the reader a summary which covers all the most important contents from the documents. According to the sentences presented in the generated summary, text summarization can be categorized as extract-based or abstract-based summary. Extract-based summary consists entirely of sentences that occurred in the original document. Abstract-based summary involves rephrasing the sentences or fusing various concepts of the document into a more generalized concept. Automatic text summarization can be categorized as indicative summary and informative summary. "Indicative abstracts allow a searcher to screen a body of literature to decide which documents deserve more detailed attention." (Edmundson 1969). Informative abstract is meant to contain all the pertinent information. It can serve as a surrogate as the original document.

2.2 Related Approaches

In the long history of research in text summarization, there are many approaches to automatic text summarization. Most of the approaches are extraction based. Sentence extraction is based on the following assumptions:

1. The most important concepts of the text are represented by the most frequently occurred words. The sentences with most number of frequently occurred words are important sentences (Luhn 1958).
2. The title conveys the content of the document and section headings convey the content of the section. Thus, sentences consist of title words and section heading words are important sentences (Edmundson 68).
3. Usually when a writer starts a paragraph, she/he starts the paragraph with the topic sentence. It is the main idea of the whole paragraph. The rest of the paragraph explains, develops or supports with evidence the topic sentence's main idea. Sometimes a topic sentence comes at the end of a paragraph. Thus, the first sentence and last sentence of each paragraph are considered important (Baxendal 1958).
4. Cue words and phrases are good indicator of importance of sentences. Sentences which include words or phrases like "importantly", "significantly", and "in conclusion" are considered important. Sentences which include words or phrases like "hardly," "it is impossible that" are considered not important (Edmundson 1968).
5. The important words or phrases are semantically connected by hyponymy and synonymy. Through the semantically related terms' lexical chaining, one can grasp the central topic of the document. After building the lexical chain and

getting the representation for the original text, build a summary from that representation. Picking the concepts represented by strong lexical chains gives a better indication of the central topic of the text. Choose the sentence which has the first member presented in a strong chain (Barzilay 1999).

6. Sentences and/or clauses can be classified into nucleus and satellite according to rhetorical structure theory (Mann 1988). Nuclei are the sentences that express the essential contents; satellites are the sentences which present additional information to the essential part. Nuclei are considered more important than satellite (Marcu 1999).
7. Based on paragraph similarity, if a paragraph is more strongly connected to other paragraphs, it is very possible it carries the central topic of the document. Choose the paragraphs with strong connections for summarization (Salton 1997).

2.2.1 Frequency Based Method

The first automatic text summarization system was developed by H. P. Luhn (1958). In his algorithm, the key sentences were extracted based on concentration of high-score words. First, words like pronouns, articles, and prepositions, which do not have the discriminatory power, are removed. Next, consolidate words that are spelled in the same way at their beginning: this is done by comparing the pair of words from their beginning letter by letter, from the point the two compared letters are not equal, count the number of letters left for both words, and if the total number of letters is equal to or less than 6, the two words are considered of similar notions, then they are aggregated together. This consolidating method was later replaced by using the stemming method (Frakes 1992). The frequencies of these aggregated words are computed, and low-frequency words are removed. Sentences are then graded by the density of the significant

words. Each sentence is segmented into groups bracketed by significant words which are not separated by more than four non-significant words apart. Each group is scored by the square of the number of significant words divided by the total number of words within the group. The sentence score is the highest scored group's score. Extract sentences with highest score to represent the document.

2.2.2 Surface Feature Based Method

Edmundson introduced three featured cue phrases, title of the document, and sentence location into text summarization (Edmundson 1969). He combined these features with the keyword feature in his algorithm.

Edmundson compiled the cue phrases dictionary from 100 heterogeneous documents and further divided the cue phrases into bonus phrases like "in summary" and "significantly" which are positively related. These phrases will be given a positive weight. Stigma phrases such as "hardly" and "impossible," which are negatively related, will be given negative weight. Null phrases (prepositions, pronouns), which are irrelevant, will be given zero weight.

Choosing title and section headings as features are based on the hypothesis. The title chosen by the author is the most representative of the content; the section headings are most representative of the following section. The title's words and heading words will give higher weight.

Choosing location is based on Edmundson's (1969) claim: "sentences occurring under certain headings are positively relevant; and topic sentences tend to occur very early or very late in the document." So he gave these sentences higher weight.

The formula (Edmundson 1969) for computing the final weight of sentences is

$$\alpha_1 C + \alpha_2 K + \alpha_3 T + \alpha_4 L, \quad (2.1)$$

where, C is a constant, K is a keyword, T is title, L is location, and α are weights. After testing, he found the combination of cue phrases, title and location gave the best results. Although these early methods are simple and easy to implement, they worked fairly well. These early researchers lay down the foundations for the future advanced text summarization.

2.2.3 Hybrid Approaches

Surface-level approach uses various features such as thematic features, title, cue phrase, and location in determining salience of information for summarization. The paper by Hovy and Lin (1998) is very typical of this approach. They describe a three-step process: topic identification, concept interpretation, and summary generation.

The topic identification step is to identify topic to get the central idea from the input text. The second step is to fuse together two or more topics to form a more general one. The final step is to use a sentence planner to generate a coherent text based on the output from the first two steps. Thus, the summarization is based on the following equation (Hovy 1998).

"Summarization=topic identification + interpretation + generation." (2.2)

The goal of topic identification is to filter the input to retain only the most important, central topics. Typically, topic identification can be achieved using various complementary techniques, including statistical method (term frequency), methods based on sentence position, and cue words. For each method, compute the weight for each sentence, and then use decision tree or neural network method to extract the sentence to form the potential abstract sentence set. Statistical method is used to calculate the

frequency for each word or phrase which is assigned as the weight for the word or phrase. The more often a phase occurs in a text, the more representative the word to the text. After this, compute the weight for each sentence. The weight of each sentence can be computed by adding up the word or phrase weight of the sentence and then normalize by the length of the sentence.

Cue words method is based on people's experience on reading. Usually, phrases such as "in conclusion" and "most importantly" can be good indicators of important content. During processing, give each keyword in the sentence containing the phrases with cue words a higher score. Then, it computes the sentence weight for each sentence.

Position-based method is based on peoples' experience too. When people write, they usually follow style rules. For instance, people like to give the text a title which represents the content of the text. They write the first sentence to represent the central idea of the paragraph. Thus, a method based on the position of the sentence is developed. Optimal Position Policy (OPP) is defined as a list that indicates in what ordinal positions in the text high-topic-bearing sentences occur (Hovy 1998). For example, the first sentence of the second paragraph is important.

Each separate topic identification module assigns a score to each sentence. Create a combination function to combine all these scores in some way to generate the best result. There are several ways to generate a coefficient for the combination function. The first one is to determine the coefficients by manual experimentation. The result is not optimal. The second one is to use the neural network technique to generate the coefficients. For example, use multilayer Perceptron with the back propagation algorithm to generate the weight matrix like Figure 2.1.

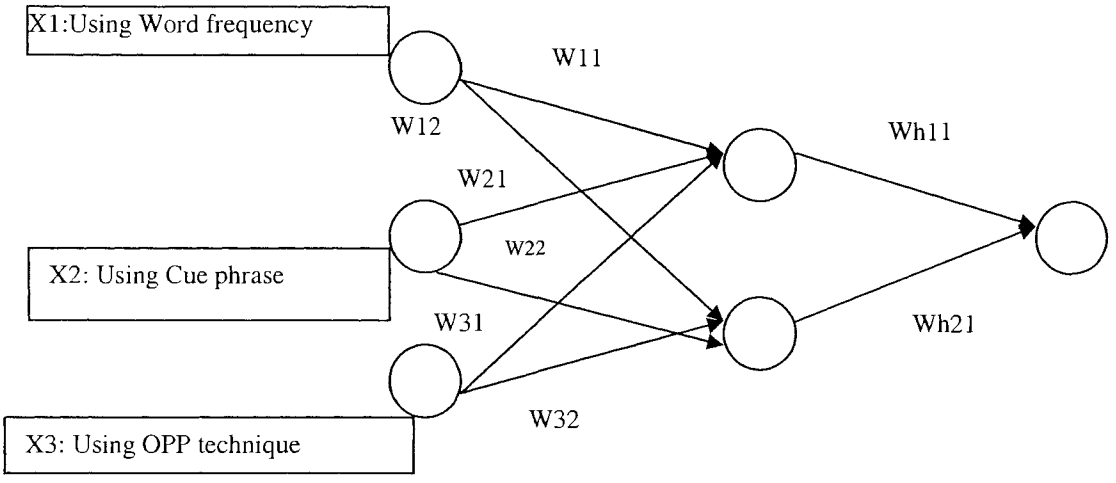


Figure 2.1 Neural Network Architecture for Optimization.

Train the Perceptrons with data to get the weight matrices. The weight matrices will be the coefficient for the combination function. Use this combination function to compute the weight for each sentence and then extract the highest weighted sentences.

2.2.4 Lexical Chains Based Method

Lexical chain was defined as a sequence of semantic related words spanning a topical unit of text (Morris 1991). For example, the words "peach" and "fruit" are related by means of super-ordinate. Lexical chains were first proposed to determine lexical cohesion among terms in a text by Morris and Hirst. "Lexical cohesion is the result of chains of related words that contribute to the continuity of lexical meaning" (Morris 1991). Lexical chain method has been widely used in topic detection from web-based breaking news (Stokes 2000), malapropism detection (St. Onge 1995) and text filtering (Li 2004).

Using the lexical chaining method for text summarization was first implemented by Barzilay and Elhadad (1997). A text is fully analyzed to find all chains of words, and

the strongest chains of words are claimed to represent the theme of a text. In this method, only nouns are candidates for constructing lexical chains. Each lexical chain consists of the candidate word; its repetitions and words can be related through WordNet relationship synonymy and hyponymy.

Lexical chain scoring metrics:

- Extra strong: between a word and its repetitions;
- Strong: between words which are directly linked by a WordNet relation (synonymy and hyponymy);
- Medium strong: between the words whose path linked by WordNet relationship is longer than one.

Summarization proceeds in three steps: construct lexical chains, identify strong chains, and extract significant sentences.

(1) A procedure for constructing lexical chains follows three steps:

- Select a set of candidate words;
- For each candidate word, find an appropriate chain relying on a relatedness criterion among members of the chains;
- If it is found, insert the word in the chain and update it accordingly.

For example: laptop→computer→machine.

(2) Strong chains identification procedure:

Picking the concepts represented by strong lexical chains gives a better indication of the central topic of the text.

The strength of each lexical chain is computed from the following rules:

- Length: The number of chain members;

- "Homogeneity index: 1 minus the number of distinct occurrences divided by the length;"
- "Score (Chain) = Length*Homogeneity Index" (Barzilay 1999).

The chains whose scores are greater than threshold are considered as strong chains.

"Score (Chain)> Average (Scores) + 2*Standard Deviation (Scores)" (Barzilay 1999).

(3) Significant sentence extraction procedure:

Once the strong chains have been selected, extract full sentences from the original text based on chain distribution. For each chain in the summary representation, choose the first sentence that contains the first chain member.

2.2.5 Discourse Tree Based Method

Daniel Marcu (1999) proposed the discourse tree method. This method parses the sentences from the original text into NUCLEUS and SATELLITE based on rhetorical relation (Mann & Thompson 1988). "The distinction between nuclei and satellites comes from the empirical observation that the nucleus expresses what is more essential to the writer's purpose than the satellite; and that the nucleus of rhetorical relation is comprehensible independent of the satellite, but not vice" (Marcu 1999). For example, "Although it's not raining, I still carry an umbrella." "Although it's not raining" is the satellite, and "I still carry an umbrella" is the nucleus. Use rhetorical paring algorithm (Marcu 1997) to parse each sentence to a discourse tree. Nuclei are considered more salient than satellite clauses. After getting the discourse tree, rank sentences according to their salience scores and extract the sentences with higher scores.

2.2.6 Graph Based Method

Apply information retrieval methods at the document level: the processing units are paragraphs within a document (Salton et al. 1994; Mitra et al. 1997; Buckley & Cardie 1997; Salton et al., 1997). Gerard Salton (1994) first presented an extractive system based on the intra-document links between passages of a document. Each paragraph P_i is represented by a vector of weighted terms of the form $D_i = (d_{i1}, d_{i2}, ...d_{it})$, where d_{ik} represents the weight for term T_k which occurred in paragraph P_i. The term weights are computed by the number of occurrences of the term in the current document (its term frequency) and the whole document collection (inverse document frequency) (Salton & McGill 1983).

Pair-wise vector similarities are computed as the inner product between vector elements (Gerard Salton 1997). Create a link between two paragraphs if their similarity is higher than a threshold. The paragraph relationship map is shown in Figure 2.2. The nodes with a large number of associated links are considered as very important paragraphs of conveying the content of the document. They use depth-first method to extract paragraphs. Start by extracting the paragraph with the highest number of associations. Among the paragraph's associated paragraphs; select those paragraphs that follow the current paragraph in text order as candidates for the next step of extraction. Among the above candidate paragraphs, choose the paragraph which is most similar to the current paragraph. By doing this, the abrupt transition of subject matter is eliminated. Repeat the above steps until enough sentences are extracted.

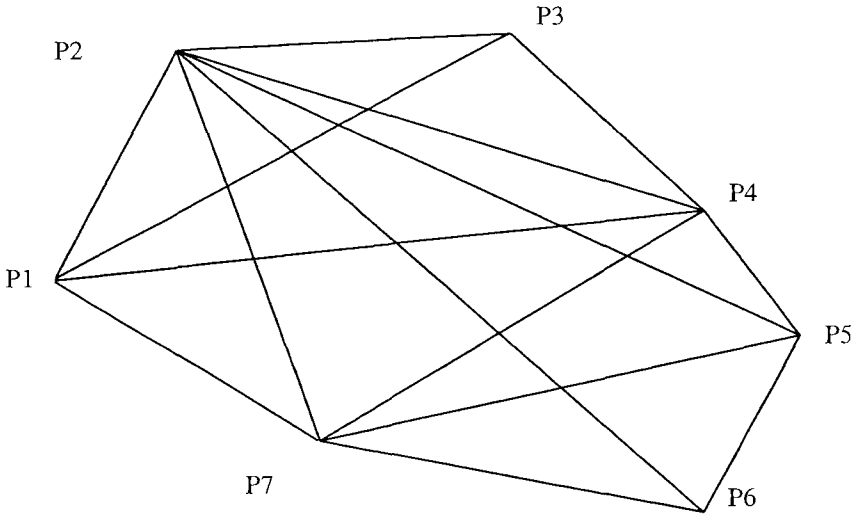


Figure 2.2 Paragraph Relationship Map.

2.3 Natural Language Processing Tools

In our text summarization project, we build summary based on the meaning instead of physical words counting. We need identify the meaning for each word and phrase in the document. To do this, we need find right meaning for each word or phrase. So we use ResearchCyc concepts as the meaning representation. We use Stanford parser to parse sentence to its phrase structure tree. In the phrase structure tree, each phrase is tagged with its phrase type, such as noun phrase, verb phrase, and preposition phrase. Each word is tagged with its part-of-speech. In ResearchCyc, there is a natural language processing system, which can map phrases and words to their Corresponding Cyc concepts.

2.3.1 Stanford Parser Tools

Stanford Parser is a syntactic parser of multi-languages including Chinese, English, German, Arabic, Italian and Bulgarian, based on Probabilistic Context-Free

Grammars (PCFGs) (Booth &Thomson 1973) (Baker 1979). Given a sentence, the parser will generate three kinds of output:

- Part-of-speech tagging (Penn tree-bank style tag) for each word within the sentence (Santorini 1990). The Penn Tree tag set is shown in Table 2.1 (Santorini, 1990).
- Phrase tree structure (noun phrases, verb phrases) for the entire Sentence
- Typed Dependencies (subject relationship, object relationship) between the words within a sentence. The Stanford grammatical relation hierarchy contains 48 grammatical relations. The basic relations of the hierarchy are very similar to that in (Carroll et al. 1999). Stanford parser adds a number of extensions and refinements to it. Typed dependency grammatical relation hierarchy of Stanford Parser is shown in Figure 2.3 (Marneffe, 2006).

To better explain Stanford parser, we use Stanford parser to parse a sentence and list the results. For the sentence:

"Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas."

The part-of-speech tagging output is as follows:

Bills/NNS on/IN ports/NNS and/CC immigration/NN were/VBD submitted/VBN by/IN Senator/NNP Brownback/NNP ,/, Republican/NNP of/IN Kansas/NNP ./

"Bills" and "ports" are tagged with "NNS," which means noun plural; "on," "by" and "of" are tagged with "IN," which means they are prepositions; "and" is tagged with "CC," which means it is a coordinating conjunction; "immigration" is tagged with "NN," which means it is a singular noun or a mass noun. "Senator," "Brownback" and

"Republican" are tagged with "NNP," which means they are proper nouns; "were" is tagged with "VBD," which means it is past tense verb; "submitted" is tagged with "vbn," which means it is a past participle form of a verb.

Table 2.1 PennTree Tag Set.

POS Tag	Description
AUX(G)	auxiliary **be, have**
CC	coordinating conjunction
CD	cardinal number
DT	determiner
IN	preposition/subordinating
JJ	adjective
JJR	adjective, comparative
JJS	adjective, superlative
NN	noun, singular or mass
NNS	noun plural
NNP	proper noun, singular
NNPS	proper noun, plural
RB	adverb
RBR	adverb, comparative
RBS	adverb, superlative
TO	to
VB	verb, base form
VBD	verb, past tense
VBG	verb, gerund/present
VBN	verb, past participle
VBP	verb, sing. present, non-
VBZ	verb, 3rd person sing.

dep - dependent
- aux - auxiliary
  - auxpass - passive auxiliary
  - cop - copula
- conj - conjunct
- cc - coordination
- arg - argument
  - subj - subject
    - nsubj - nominal subject
      - nsubjpass - passive nominal subject
    - csubj - clausal subject
  - comp - complement
    - obj - object
      - dobj - direct object
      - iobj - indirect object
      - pobj - object of preposition
    - attr - attributive
    - ccomp - clausal complement with internal subject
    - xcomp - clausal complement with external subject
    - compl - complementizer
    - mark - marker (word introducing an advcl)
    - rel - relative (word introducing a rcmod)
    - acomp - adjectival complement
  - agent - agent
- ref - referent
- expl - expletive (expletive there)
- mod - modifier
  - advcl - adverbial clause modifier
  - purpcl - purpose clause modifier
  - tmod - temporal modifier
  - rcmod - relative clause modifier
  - amod - adjectival modifier
  - infmod - infinitival modifier
  - partmod - participial modifier
  - num - numeric modifier
  - number - element of compound number
  - appos - appositional modifier
  - nn - noun compound modifier
  - abbrev - abbreviation modifier
  - advmod - adverbial modifier
  - neg - negation modifier
  - poss - possession modifier
  - possessive - possessive modifier ('s)
  - prt - phrasal verb particle
  - det - determiner
  - prep - prepositional modifier
- sdep - semantic dependent
  - xsubj - controlling subject

Figure 2.3 The Grammatical Relation Hierarchy.

The phrase structure tree output is as follows: the branches are phrases such as verb phrases, noun phrases, preposition phrases; the leaves are the single words. These phrases will be mapped to correct concepts in Cyc knowledgebase.

(ROOT
(S
(NP
(NP (NNS Bills))
(PP (IN on)
(NP (NNS ports)
(CC and)
(NN immigration))))
(VP (VBD were)
(VP (VBN submitted)
(PP (IN by)
(NP
(NP (NNP Senator) (NNP Brownback))
(, ,)
(NP
(NP (NNP Republican))
(PP (IN of)
(NP (NNP Kansas)))))))
(. .)))

Typed Dependencies Output is as follows: The term outside the parenthesis represents the type of dependency between individual words. The two terms inside the parenthesis are the individual words being constrained by the dependency type. For example, *"nsubjpass(submitted-7, Bills-1)"* means *"Bills"* is a passive nominal subject, *"submitted"* is the predicate. We will extract (predicate, subject) dependency and (predicate, object) dependency from the typed dependency relationships of the sentence. We will create a new tuple (subject, predicate, object) from the above relationships and create new sentences from the tuples based on their compatibility.

nsubjpass(submitted-7, Bills-1)
prep(Bills-1, on-2)
pobj(on-2, ports-3)
cc(ports-3, and-4)
conj(ports-3, immigration-5)

auxpass(submitted-7, were-6)
prep(submitted-7, by-8)
nn(Brownback-10, Senator-9)
pobj(by-8, Brownback-10)
appos(Brownback-10, Republican-12)
prep(Republican-12, of-13)
pobj(of-13, Kansas-14)

2.3.2 ResearchCyc Knowledgebase

ResearchCyc is the world's largest and most complete general common sense knowledge base and commonsense reasoning engine. The Cyc project was started by Dr. Douglas Lenat in 1984. The goal of Cyc project is to build a general purpose intelligent system. Feed Cyc with common sense knowledge, when there is enough common sense in Cyc, Cyc should be able to learn and reason by itself. The language used for knowledge representation in Cyc is CycL, which is a logic-based language. English words and phrases are encoded as concepts. Facts and rules in everyday life are encoded as assertions. Cyc inference engine draws conclusions by deductions from those assertions. Cyc includes 300,000+ concepts. There are about 26,000 relations to interrelate, constrain, and define the concepts. These relations and concepts are used to construct the facts and rules (nearly 3,000,000) in everyday life. The concepts are organized into a hierarchy. The Cyc upper ontology is shown in Figure 2.4.

There is a natural language processing system (called Cyc-NL) built into Cyc. The Cyc-NL system includes a lexicon, a morphology system, a parser system and a natural language generation system.

- The lexicon includes information about word to part-of-speech correspondences and word to Cyc concept correspondence. For example, assertion "(#$singular #$Dog-TheWord "dog")" relates the word "dog" to part-of-speech #$singular;

assertion "(#$denotation #$Bat-TheWord #$SimpleNoun 0 #$Bat-Mammal)" relates the lexical word #$Bat-TheWord to Cyc concept #$Bat-Mammal.

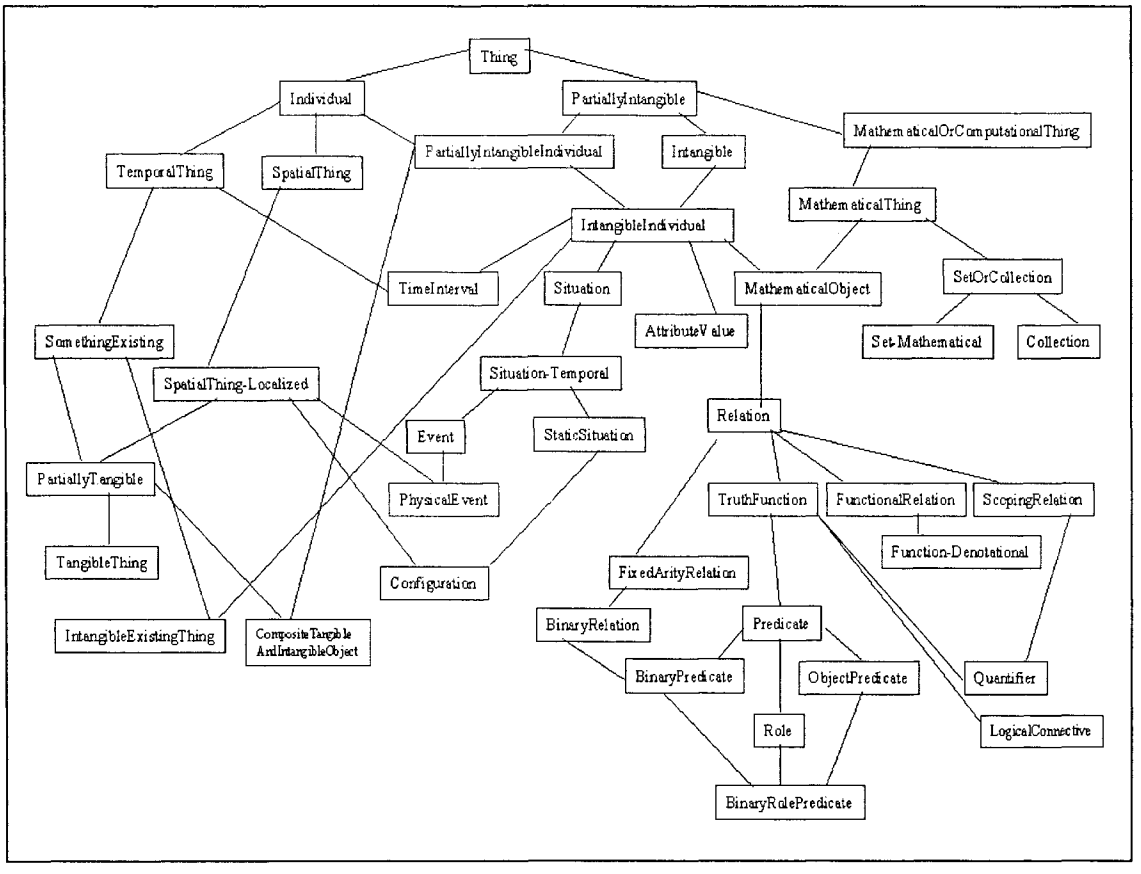


Figure 2.4 Cyc Upper Ontology.

- The morphology system can transform an inflected word to its root form and transform root words to their reflected form according to users' needs. For examples, morphology system can transform the plural form of word "dogs" to its root form "dog"; transform root form of verb "sing" to its gerund form "singing".
- The parser system can parse a natural language word or phrase to Cyc concepts.
- The natural language generation system can transform Cyc concepts, rules and sentences into natural language. For example, Cyc sentence *"(forAll ?PERSON1*

(implies (isa ?PERSON1 Person) (thereExists ?PERSON (and (isa ?PERSON2 Person) (loves ?PERSON1 ?PERSON2)))))" will be transformed to natural language sentence *"for every ?PERSON2, it is the case that for every ?PERSON1, if ?PERSON1 is someone, then ?PERSON2 is someone and ?PERSON1 loves ?PERSON2."*

To demonstrate how does Cyc work, we attach a browser display of concept #$Dog as in Figure 2.5. There are a total of 528 assertions which define #$Dog concept in Cyc and relate #Dog to other concepts in Cyc. For example, #Dog is related to its super-ordinates by #$genls. In #$DomesticBreedsVocabularyMt, its super-ordinate is #$DomesticatedAnimal. In #$UniversalVocabularyMt, its super-ordinates are #$CanisGenus and CanineGenus. Some common sense knowledge such as "dogs make bark sound" is asserted by predicate #$AnimalTypeMakesSoundType. In following figures (Figure 2.6, Figure 2.7, and Figure 2.8), we attach a more detail list of predicates which can relate #$Dog to other concepts in Cyc are attached.

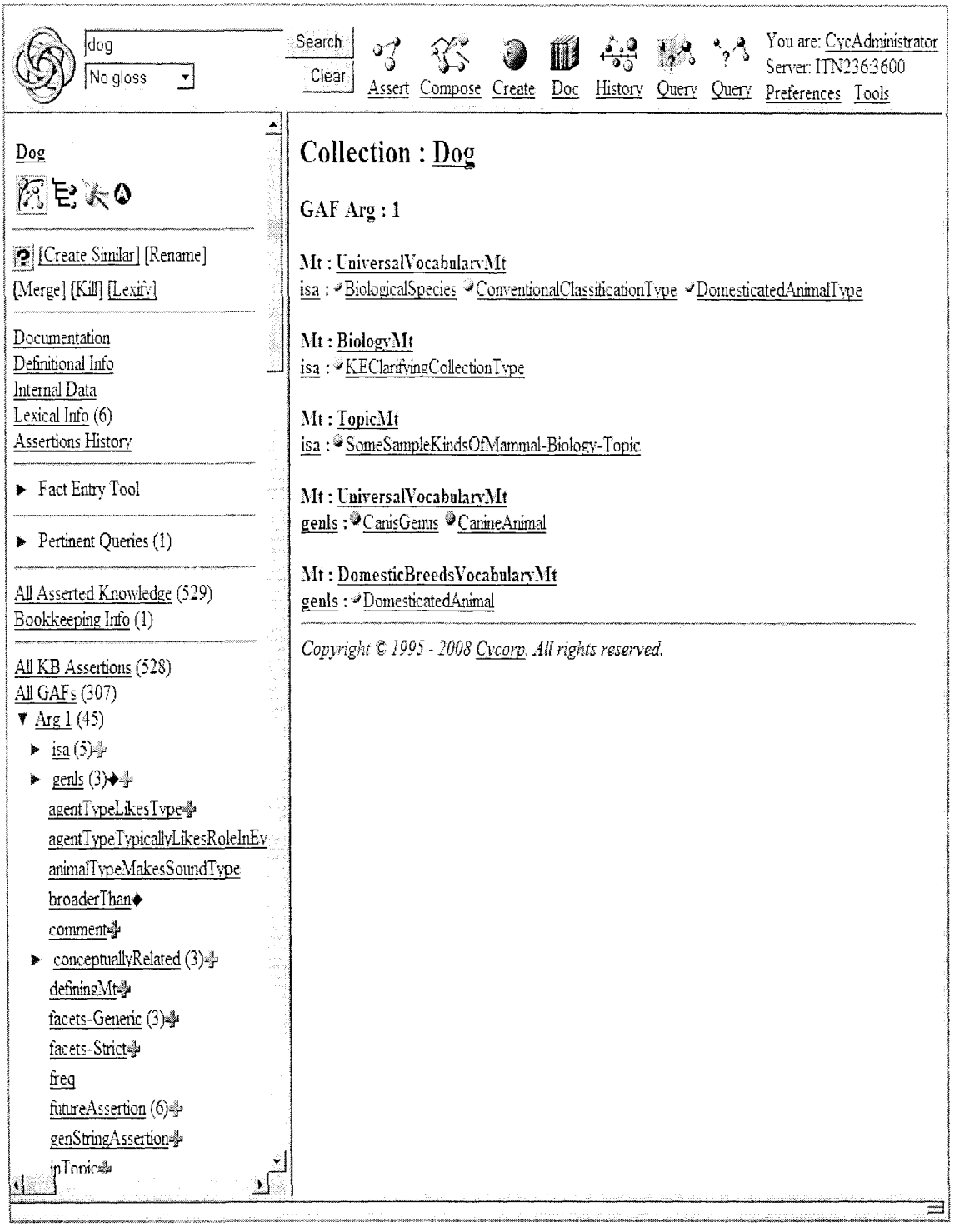


Figure 2.5 The Brower Display of Concept #$Dog.

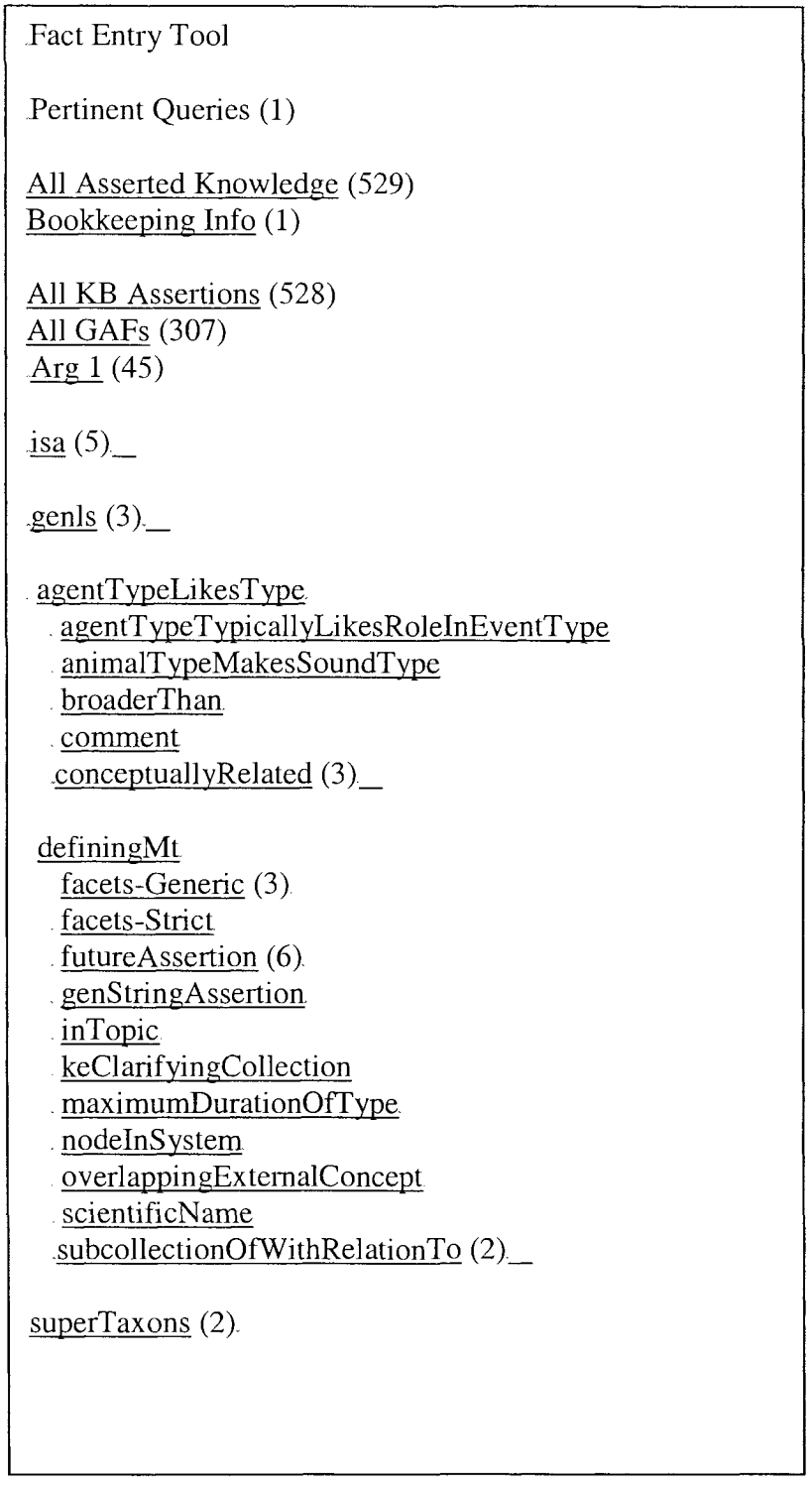


Figure 2.6 Assertions Associated with #$Dog (Part 1).

Figure 2.7 Assertions Associated with #$Dog (Part 2).

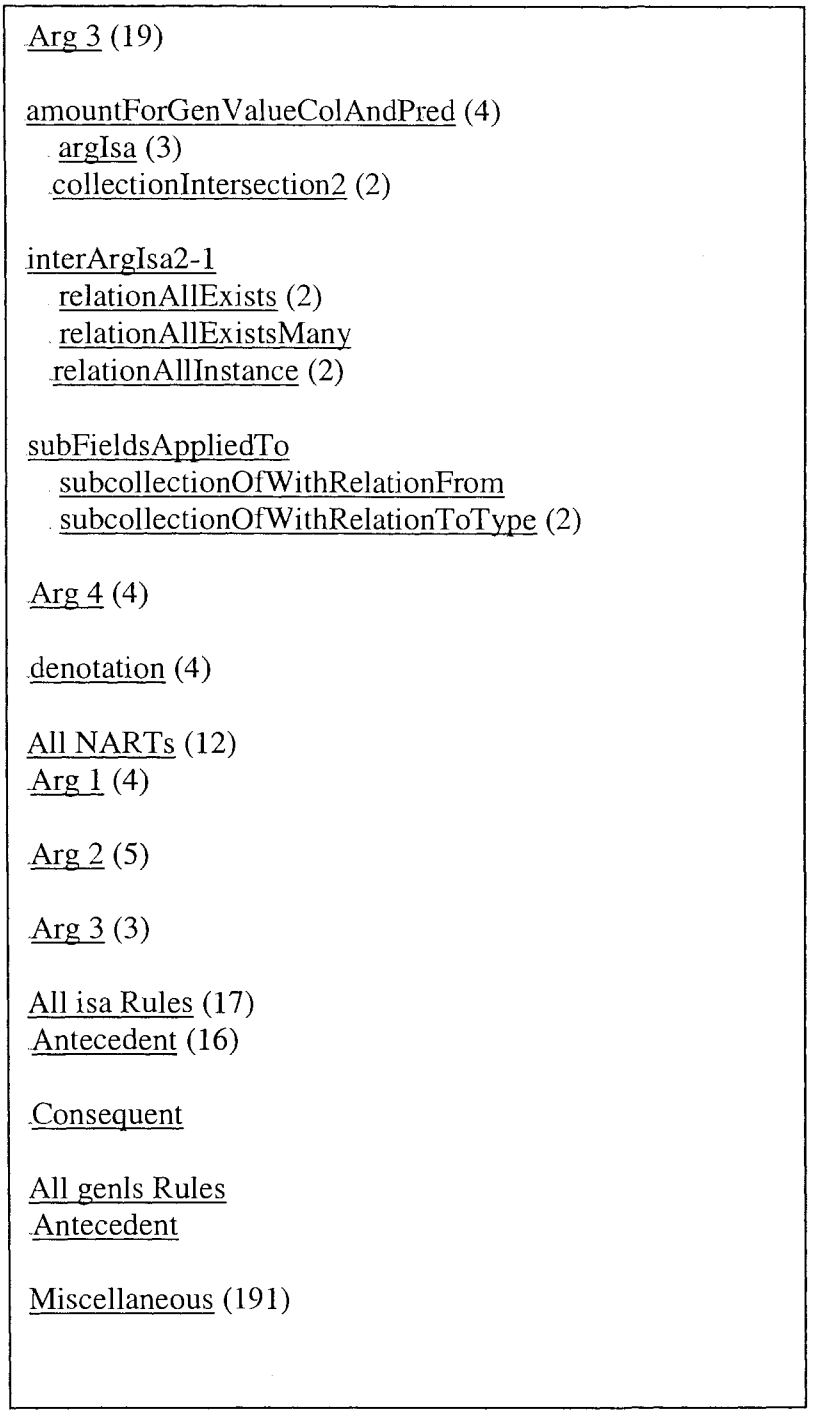


Figure 2.8 Assertions Associated with #$Dog (Part 3).

CHAPTER 3

TEXT SUMMARIZATION USING CONCEPT WEIGHT PROPAGATION METHOD

This chapter describes our methods for document summarization to generate key concepts, extract key sentences, and create new sentences. We use ResearchCyc to map the word to its corresponding concepts. We choose ResearchCyc for the following reasons. Compared to Wordnet, ResearchCyc is coarse grained in terms of sense distinction, thus it is easier to make sense disambiguation with ResearchCyc. ResearchCyc has 300,000 constants and more than 3,000,000 assertions to relate these concepts. ResearchCyc has enough constants to cover the natural language words. And, we can use the ResearchCyc commands to map words to their corresponding ResearchCyc concepts.

3.1 Introduction

Our goal is to create a domain-independent text summarization system. Thus, we only use domain-independent features. In this project, we generate text summary based on concept distributions and concept hierarchies.

We generate a summary by the semantic meaning contribution of each word or phrase. Some researchers used #$nearestIsa, #$nearest-IsaOfType and #$conceptuallyRelated to resolve sense disambiguation at local sentence and paragraph level (Jon 2006). We resolve this problem at global document level. At first, we map the

phrases and words to the corresponding Cyc concepts, and increase the weight of each of those concepts. Then, we propagate the weight three levels upward to create the concept representation for the document. By propagating the weight, we will favor the more generalized terms over the specific ones when we generate text summary. We semantically annotate each word by its corresponding Cyc concept and assign the Cyc concept's weight to the word. For each sentence, we add the weight of each content word and normalize it by the number of terms, and assign the weight to the sentence. Then, we sort the sentence weight in descending order, and output the top 5-10 sentences with highest weight. The procedures are shown in Figure 3.1 and Figure 3.2. These figures outline the steps needed to generate new concepts, extract key sentences (Choi & Huang, 2009), and create new sentences to summarize the document. These steps are described in more detail in the following subsections.

3.2 Filter and Map Phrases to Cyc Concepts

The first issue is to map words within a document to Cyc concepts. There are two steps: first we map phrases to Cyc concepts, then map the single words which are not covered within a phrase to Cyc concept.

There are two reasons: first, sometimes a phrase has a semantic meaning different from combining the words within the phrase, like "hot dog" means a kind of food instead of a kind of hot animal; second, a phrase is easier to be mapped to Cyc concept, each word may have several semantic meanings, combining the words together does not multiply the semantic meanings of the phrase carries, instead reducing the semantic meanings. The words within the phrase serve as confinement among themselves instead of a multiplier. For example, "Apple" could mean 1) a kind of fruit or 2) a computer

company. Juice means a kind of liquid. "Apple juice" only has one meaning, a kind of juice.

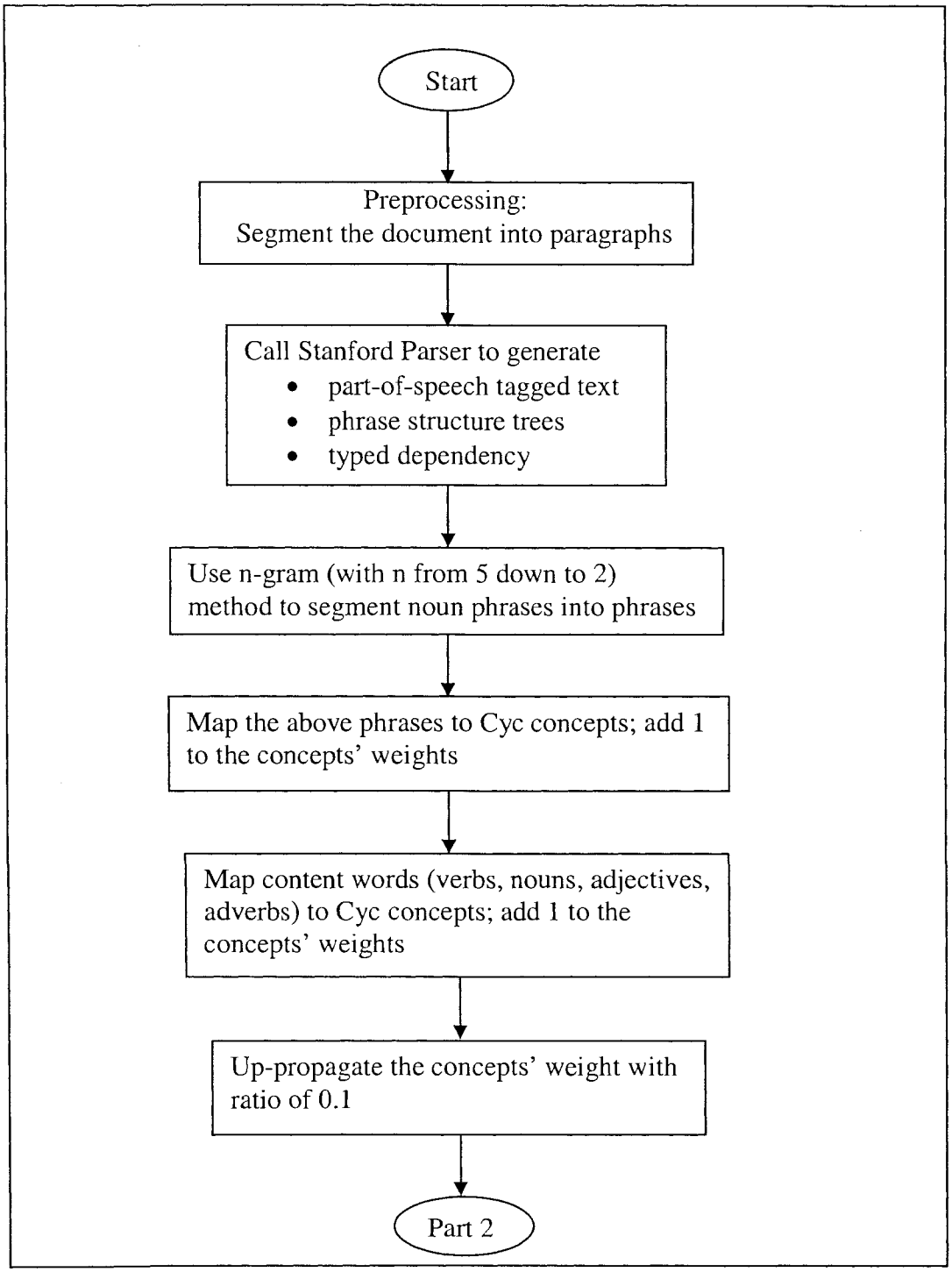


Figure 3.1 Flow Chart of Text Summarization System (Part 1).

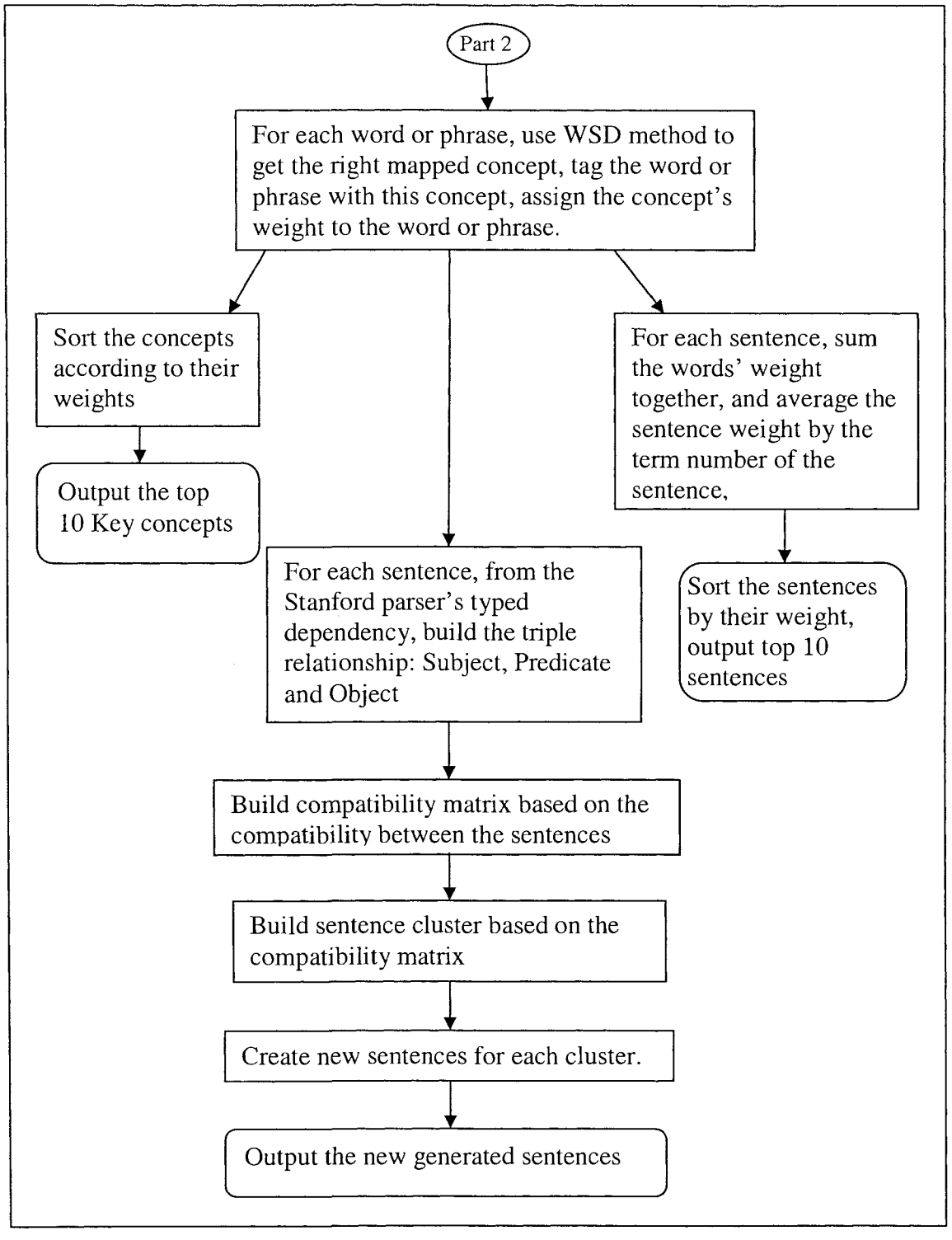
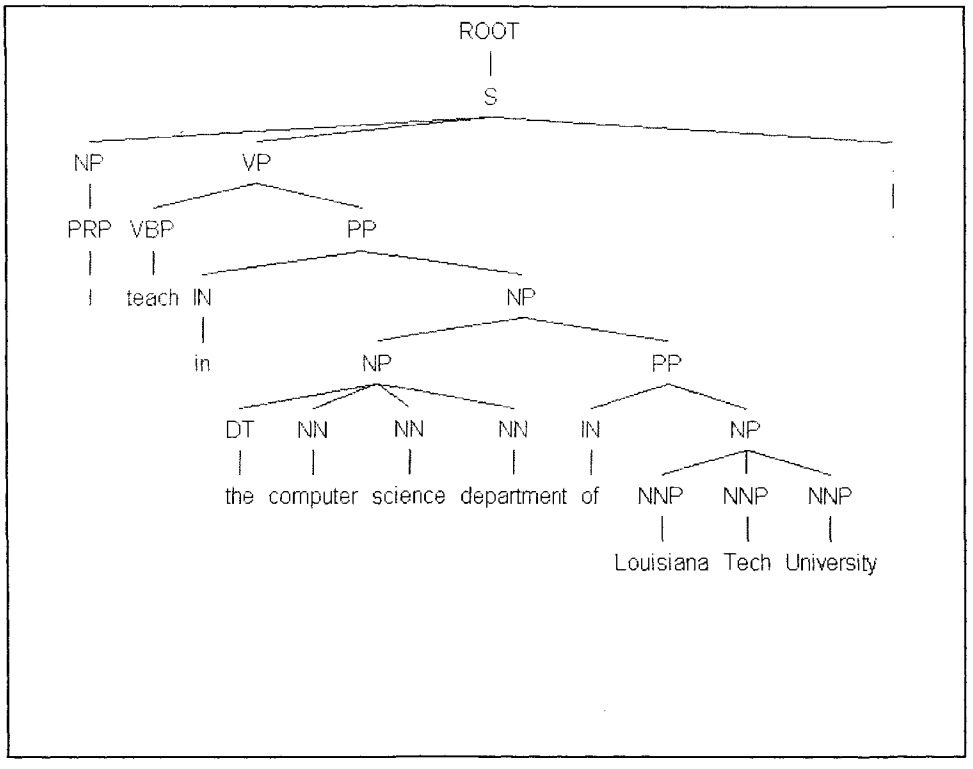


Figure 3.2 Flow Chart of Text Summarization System (Part 2).



Parsing: "I teach in the computer science department of Louisiana Tech University."

Figure 3.3 An Example of Phrase Structure

For phrases, since Cyc only has direct mapping from noun phrases to Cyc concepts, we only handle noun phrases. We use Stanford parser to parse the sentence to get the sentence's context-free phrase structure grammar representation. We extract the noun phrases from the representation. For each noun phrase, we apply n-gram word selection (n from 5 to 2) and filter out the noun phrase. The n-gram word selection extracts word sequences with n consecutive words from the candidate list. After we get the sequences for 5-gram, we try and filter out the noun phrase. The n-gram word selection extracts word sequences with n consecutive words from the candidate list. After we get the sequences for 5-gram, we try to use Cyc command "all-denots-of-string" to get their corresponding Cyc concept. If any phrase is mapped to Cyc a concept, the process

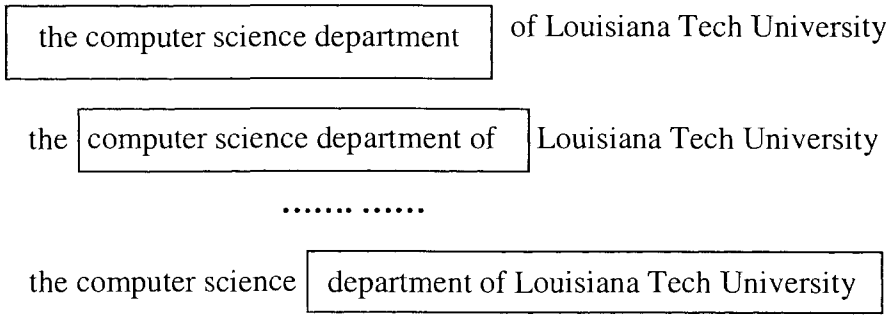


Figure 3.4 N-gram Parse: n=5.

deletes the sequence from the noun phrase, and then adds the weight of their corresponding Cyc concept by 1. If they do not map to any Cyc concept, the process applies the 4-gram method, and then repeats until all the words in the noun phrase are deleted or after the 2-gram extraction method is applied.

For example, we use a simple sentence written by a TA, "I teach in the computer science department of Louisiana Tech University." After we parse the sentence by the Stanford Parser, we get the results shown in Figure 3.3.

We extract the phrases labeled as "NP"(means noun phrase), "I" and "the computer science department of Louisiana Tech University." Since "I" just has one word, it is not a noun phrase, so we just ignore it for the moment.

Apply the 5-gram method to the noun phrase "the computer science department of

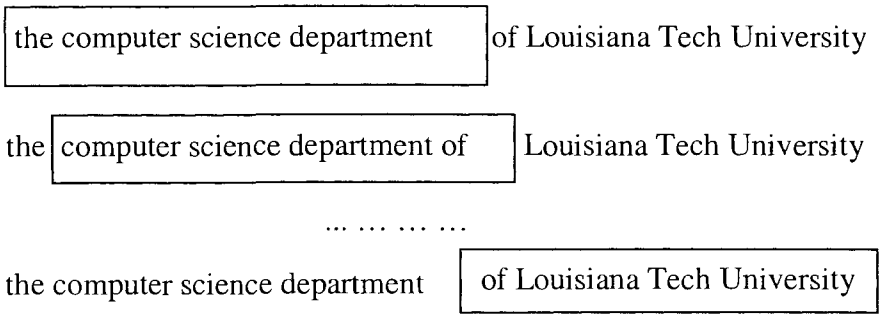


Figure 3.5 N-gram Parse: n=4.

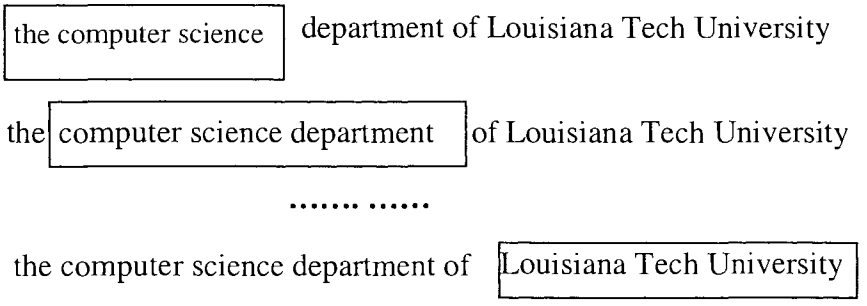


Figure 3.6 N-gram Parse: n=3.

Louisiana Tech University," and we get the following sequences "the computer science department of," "computer science department of Louisiana," "science department of Louisiana Tech," and "department of Louisiana Tech University" in Figure 3.4. Start from the first sequence, and use the Cyc command "all-denots-of-string" to check if the sequence has a corresponding Cyc concept.

None of the above 5-gram segment has a corresponding CYC concept, so we use the 4-gram extraction method, and we get "the computer science department," "computer science department of," "science department of Louisiana," "and department of Louisiana Tech," and "of Louisiana Tech University" in Figure 3.5. We use CYC command "all-denots-of-string" to check if the sequence has a corresponding CYC concept.

None of the above 4-gram segments has a corresponding Cyc concept, so we use the 3-gram extraction method, and we get "the computer science," "computer science department," "science department of," "department of Louisiana," "of Louisiana Tech," and "Louisiana Tech University" in Figure 3.6. We use Cyc command "all-denots-of-string" to check if the sequence has a corresponding Cyc concept. This time we have two catches: "computer science department" is mapped to Cyc concept "#$ComputerScienceDepartment" and "Louisiana Tech University" is mapped to

"#$LouisianaTechUniversity." we increase the weight of #$ComputerScienceDepartment by 1 and increase the weight of #$LouisianaTechUniversity by 1. The weight will be a fundamental element when we extract sentences from the document. After we delete these two sequences, we have two segments, "the" and "of." Since each of them just has one word in it, it cannot be a phrase, so we just stop the collocation checking for the phrase. The following figures show the segmenting process: The noun phrase is "the computer science department of Louisiana Tech University."

3.3 Map Word to Cyc Concept

In this step, we use Stanford parser to parse each sentence, and get the part-of-speech for each word within the sentence. Table 3.1 and 3.2 are used to show some of the

Table 3.1 Cycl Predicate for Relating Natural Language Word to Cyc Concepts.

Example queries	Output			
	Word	POS	wt	TERM
(and (denotation ?Word ?POS ?NUM ?TERM) (wordForms ?WORD nounStrings "apple") (genls ?POS Noun))	Apple-TheWord	Proper-CountNoun	0	AppleInc
	Apple-TheWord	Count Noun	0	(FruitFn AppleTree)
(and (denotation ?Word ?POS ?NUM ?TERM) (wordForms ?WORD verbStrings "eat") (genls ?POS Verb))	Eat-TheWord	Verb	0	EatingEvent
(and (denotation ?Word ?POS ?NUM ?TERM) (wordForms ?WORD adjectiveStrings "beautiful")(genls ?POS Adjective))	Slow-TheWord	Adverb	0	(LowAmountFn Speed)

Table 3.2 Concepts and Their Original Weights.

Concept	Weight
Greyhound-Dog	50
Harrier-TheDog	40
Terrier-TheDog	8
Wolf	10
Dog	20

concepts of this step. By using the part-of-speech tagging, we can significantly reduce the word's semantic meaning set. For example, the word "saw," if it is tagged with a noun, means "a tool that you use for cutting wood;" if it is tagged with a verb, it could mean "to cut something using a saw," or past tense of "see." If we know it is a noun word, we just map it to "saw: a tool that you use for cutting wood." After we get the part-of-speech for each content word, we take the word and its part-of-speech as argument and use Cyc #$Denotation to map the word to its corresponding Cyc concept. At this moment, the word can still be mapped to several Cyc concepts. We will choose the right concept for each word later. Then, we map the word to its Cyc Concepts, we check if the current word is in stop word list, if they just set their weight as 1, otherwise increase the weight of each of the concepts by 1. The stop word list is shown in Appendix A, which we created based on a list provided on www.thebananatree.org (2007).

The Cyc ontology includes a variety of predicates which relate an English word or phrase to Cyc concepts. For example, #$denotation is a predicate which relates a #$LexicalWord, "SpeechPart", and "sense number" to Cyc concepts. In this project, we handle noun, verb, adjective and adverb, because only these words will significantly

contribute to the document content. The samples of the "denotation" command for noun, verb, adjective and adverb are shown in Table 3.1.

From Table 3.1, we can see "apple" is mapped to 2 corresponding Cyc concepts, "AppleInc" and "(FruitFn AppleTree)." Change "apple" to any noun word, and we can get the noun word's corresponding Cyc concepts; "eat" is mapped to 1 corresponding Cyc concept "EatingEvent." Change "eat" to any verb word, and we can get the verb's corresponding Cyc concepts; "slowly" is mapped to 1 corresponding Cyc concept "(LowAmountFn Speed)." Change "slowly" to any adverbs, we can get the adverb word's corresponding Cyc concepts; "beautiful" is mapped to 1 corresponding Cyc concept "StunninglyBeautiful." Change "beautiful" to any adjectives, we can get the adjective word's corresponding Cyc concepts.

3.4 Propagate Concept Weights Upward

The process continues by propagating the weight of each concept three levels upward, each level scaled down by a ratio of 0.1. By propagating the weight upward, we favor the more generalized concepts. The propagation will help for Word Sense Disambiguation later. For example, we have five concepts: Greyhound-Dog, Harrier-TheDog, Terrier-FunctionalGroup, Wolf, and Dog; their weights are 50, 40, 8, 10, and 20. Propagate the concepts' weight three levels up. The original concept, weight table is shown in Table 3.2 and the concepts' graph structure is shown in Figure 3.7.

At each level's propagation and for each concept, we use Cyc command "min-genls" to get their parents and then propagate the weight to their parents. Thus, each concept's weight equals its original weight plus its children's weight multiplied by 0.1. Its grandchildren's weight is multiplied by 0.01. Its great-grandchildren's weight is

Table 3.3 Propagated Concept Hierarchy Weight after 3-Level Propagation

Concept/WT	Concept/WT	Concept/WT	Concept/WT
Current	Parent	Grandparent	Grandgrandparent
Greyhound-Dog/50	Hound/5	Dog 0.5	CanisGenus/0.05
			DomesticatedAnimal/0.05
Harrier-TheDog /40	Hound/4	Dog/0.4	CanisGenus/0.04
			DomesticatedAnimal/0.04
Terrier-FunctionalGroup /8	Dog/0.8	CanisGenus/0.08	CanineAnimal/0.008
		DomesticatedAnimal/0.08	TameAnimal/0.008
			NonPersonAnimal/0.008
Wolf /10	CanisGenus/1	CanineAnimal/0.1	Carnivore/0.01
			CarnivoreOrder/0.01
			TerrestrialOrganism/0.01
Dog/20	CanisGenus/2	CanineAnimal/0.2	Carnivore/0.02
			CarnivoreOrder/0.02
			TerrestrialOrganism/0.02
	Domesticated Animal/2	TameAnimal/0.2	Animal/0.02
		NonPersonAnimal/0.2	Animal/0.02

multiplied by 0.001. Table 3.3 represents Current concept, its parents, its grandparents, its great-grandparents and their propagated weight. Table 3.4 represents Concept and their final weight after three levels of propagation. Figure 3.8 is the concept hierarchy structure after three-level propagation.

3.5 Choose the Right Concept for Each Word

We choose concept for a word based on the assumption that one word keeps the same semantic meaning throughout the document. The semantic meanings of the key concepts occurred within the document should be close to some extent. For example, if a document is talking about fruit, "pear" occurred in the document, and "apple," "banana"

might also occur in the document. If "dog" occurred in the document, it's conceptually related words like "bite" and "bark" might also occur in the document.

In this project, we choose the right concept for each word based on weight. The steps are outlined as follows:

For each word or phrase within the sentence, retrieve its corresponding Cyc concepts,

1. If the word has only one corresponding Cyc concept, label the word with this concept and retrieve the concept's weight and assign the weight to the word, or else go to step 2. This weight will be used when we extract sentences from the document.

Table 3.4 Concepts and Their Final Weights after 3-Level Propagation.

Concepts	Weights
Greyhound-Dog	50
Harrier-TheDog	40
Terrier-FunctionalGroup	8
Wolf	10
Dog	21.7
Hound	9
CanisGenus	3.17
DomesticatedAnimal	2.17
CanineAnimal	0.308
TameAnimal	0.208
NonPersonAnimal	0.208
Carnivore	0.03
CarnivoreOrder	0.03
TerrestrialOrganism	0.03
Animal	0.04

2. If the word has more than one corresponding concept, retrieve their weights. If one concept's weight is higher than the others, choose the concept with the highest weight and label the word with this concept and assign its weight to the word, else go to step 3.
3. If we could not determine the word's concept based on its corresponding Cyc concepts' weight, we will try to choose the right concept based on their parents' weight.

   For each one of the word's corresponding Cyc concepts,

   a. Use SubL command "min-genls" to retrieve its parents and their parents' weight,

   b. If one Cyc concept's parents' weight is more than the others, take this concept as the word's corresponding Cyc concept, label the word with this concept, and assign the concept's weight to the word, or else go to step 4.

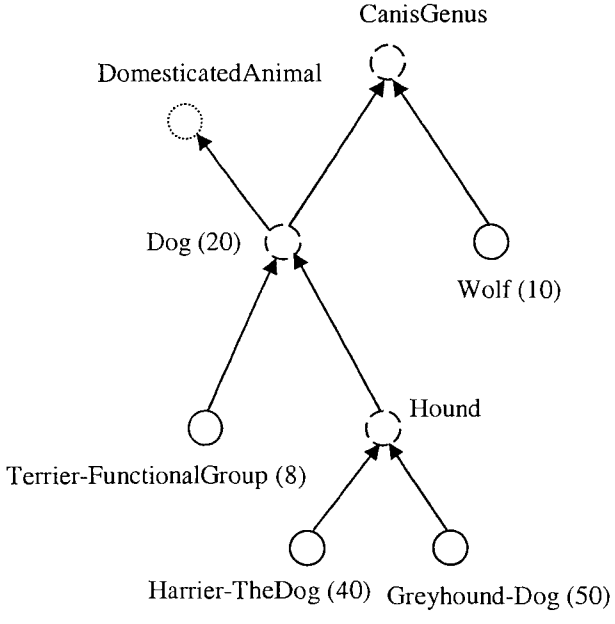


Figure 3.7 Nodes with Their Original Weights.

4. If we could not determine the right concept for the word, we will try to solve this problem by using the weights of the concepts' conceptually related concepts.

   For each candidate concept

   a. Use Cyc predicate #$conceptuallyRelated to retrieve the concept's related concept.

   b. For each retrieved related Cyc concept, retrieve its weight, and sum the weight together.

   c. If one concept's conceptually related concepts' weight sum is higher than others, take this Cyc concept as the word's corresponding Cyc concept, label the word with this concept and assign the concept's weight to it.

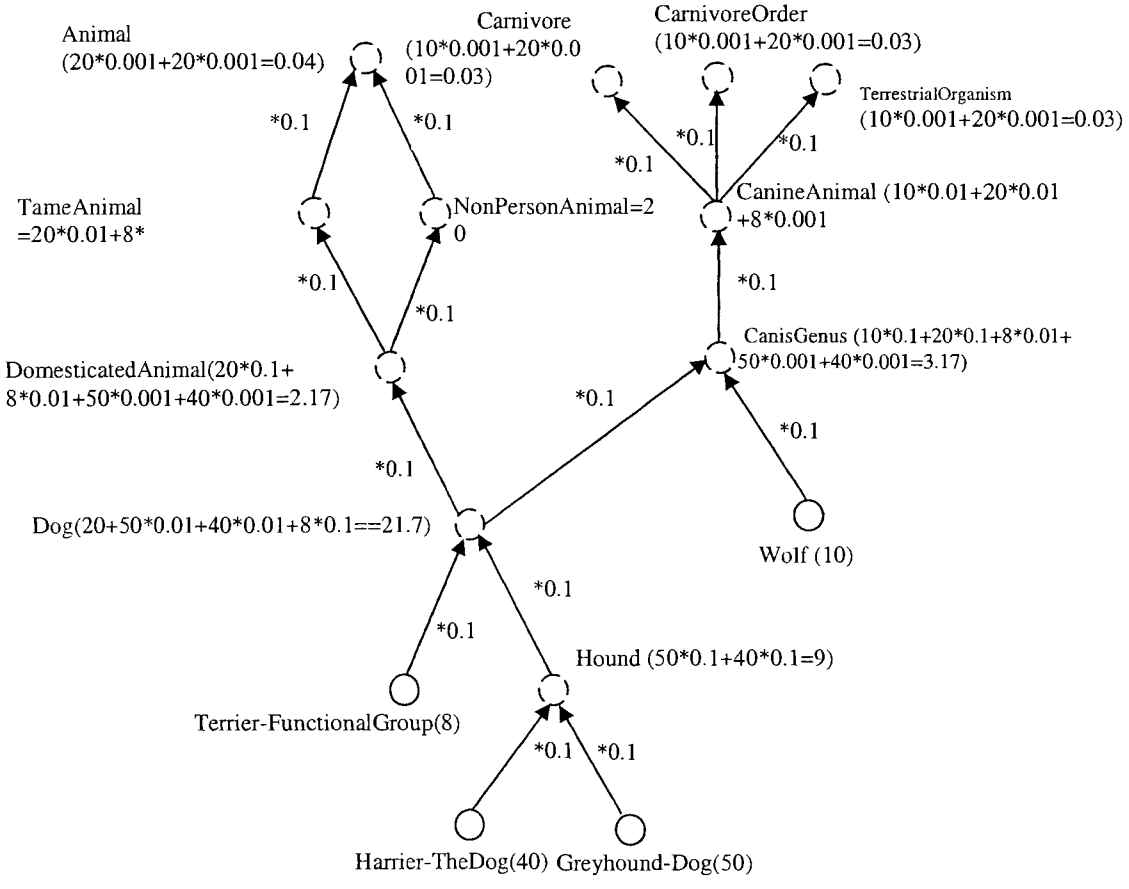


Figure 3.8 Concept Hierarchy after 3-Level Propagation.

3.5.1 WSD by Concepts' Weight

Since the weight of each concept is the sum of its original weight and its children, grandchildren, great-grandchildren's weight, when we compare the word's corresponding Cyc concepts' weight, we not only consider the word's corresponding Cyc concepts' occurrence frequency, but also their children, grandchildren, great-grandchildren's occurrences within the document.

For example, we have a word "WD;" it has two corresponding Cyc concept, one is X0, the other is Y0. Their hierarchical structure is shown in Figure 3.9.

c: means children

gc: means grandchildren

gcc : means great-grandchildren

X0 has children Xc1, Xc2; grandchildren Xgc11, Xgc12, and Xgc2 and great-grandchildren Xgcc21 and Xgcc22.

Y0 has children Yc1, Yc2, Yc3; grandchildren Ygc11, Ygc12 and Ygc2 and Ygc3; and great-grandchildren Ygcc1, Ygcc21 and Ygcc22.

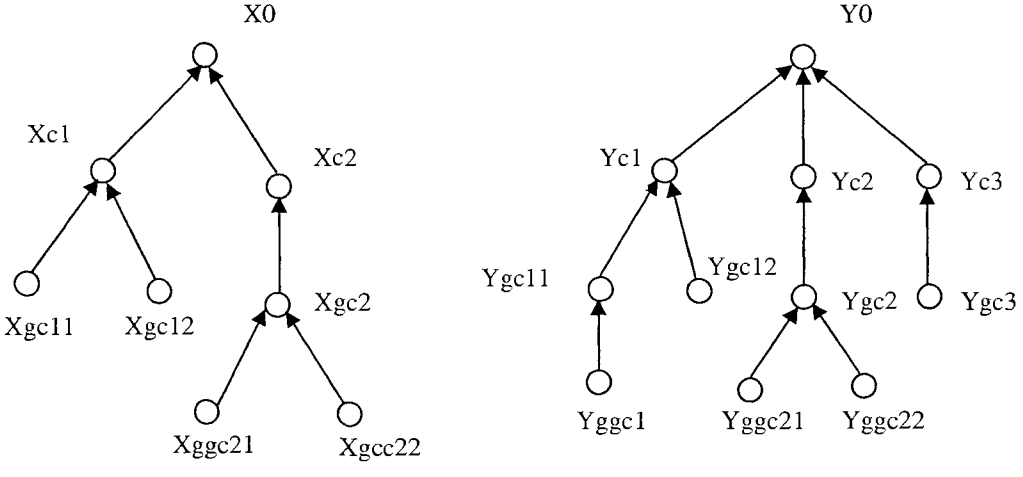


Figure 3.9 Hierarchical Structure of Concepts X0, Y0.

Their weights can be computed as

Wt(X0) = Wt(X0) + 0.1Wt (Xc1) +0.1Wt (Xc2) + 0.01Wt (Xgc11) + 0.01Wt (Xgc12) + 0.01Wt (Xgc2) +0.001 Wt (Xgcc21) + 0.001Wt (gcc22)

Wt(Y0) = Wt(Y0) + 0.1Wt(Yc1) +0.1Wt(Yc2) +0.1Wt(Yc3)+0.01Wt(Ygc11) + 0.01Wt(Ygc12) + 0.01Wt(Ygc2) + 0.01Wt(Ygc3) +0.001 Wt(Ygcc1) +0.001 Wt(Ygcc21) + 0.001Wt(Ygcc22)

If Wt(X0) is greater than Wt(Y0), choose X0 as the word corresponding Cyc concept. If Wt(Y0) is greater than Wt(X0), choose Y0 as the word corresponding Cyc concept. If they are equal, compare their parents' weight.

3.5.2 WSD by Concepts' Parents' Weights

By comparing the weights of the parent nodes, we not only consider the occurrence number of parent nodes, current node, current node's children, grandchildren, but also their peer level concepts and their children, and grandchildren.

The weight of the parent node is the sum of its occurrence and the propagated weight from its children, grandchildren, and great grandchildren. For example, we could not determine which one to choose for word "WD" based on the weight of X0 and Y0, so we try to disambiguate the senses based on their parents' nodes' weight. For parents Xp, Yp's hierarchical structure is shown in Figure 3.10.

X0 has peer Xp0, Xp0 has child Xp1, grandchildren Xp2.

Y0 has peer Yp0, Yp0 has child Yp1 and grandchildren Yp21 and Yp22.

XP andYP's weights can be computed as

Wt (XP) = Wt (XP) +0.1Wt(X0) +0.1Wt (Xp0) + 0.01Wt (Xc1) +0.01Wt (Xc2) + 0.01Wt (Xp1) + 0.001Wt (Xgc11) +0.001Wt (Xgc12) + 0.001Wt (Xgc2) +0.001Wt(Xp2).

Wt(YP) = Wt(YP)+ 0.1Wt(Y0) +0.1Wt(Yp0) + 0.01Wt(Yc1) +0.01Wt(Yc2) +0.01Wt(Yc3)+0.01Wt(Yp1)+0.001Wt(Ygc11) + 0.001Wt(Ygc12) + 0.001Wt(Ygc2) + 0.001Wt(Ygc3) +0.001 Wt(Yp21) +0.001 Wt(Yp22).

If Wt (XP) is greater than Wt (YP), choose X0 as the word corresponding Cyc concept. If Wt (YP) is greater than Wt (XP), choose Y0 as the word corresponding Cyc concept. If they are equal, compare their conceptually related terms' weight.

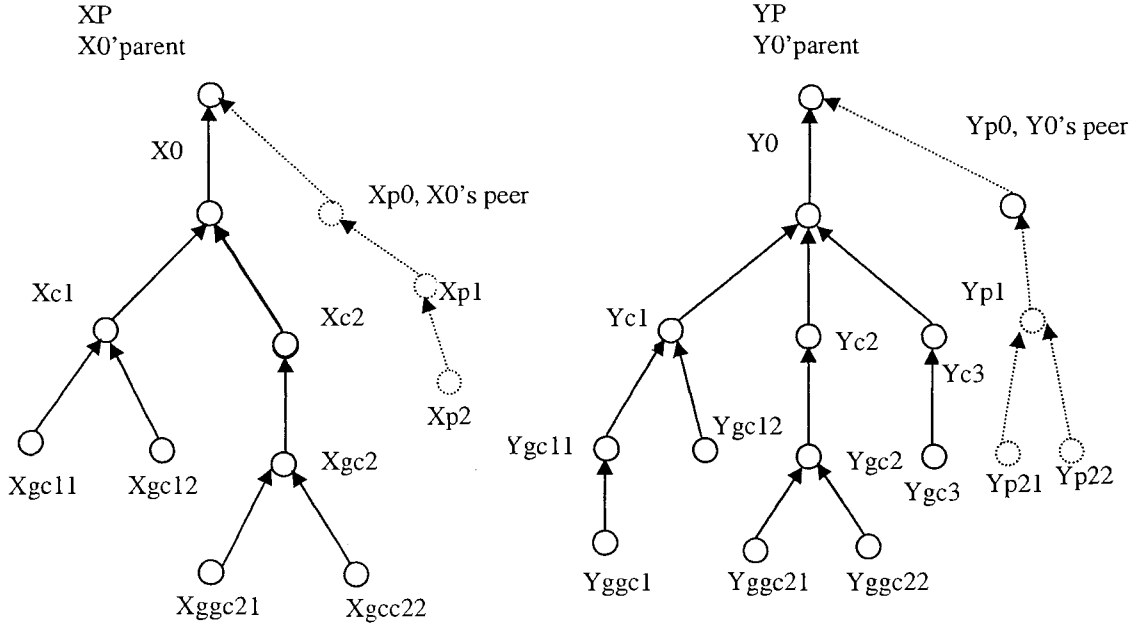


Figure 3.10 Hierarchical Structure of Concepts Xp, Yp.

3.5.3 WSD by Concepts' Conceptually Related Concepts

If we could not disambiguate between X0 and Y0 based on their parents' weight, we try to solve the problem by comparing their conceptually related concepts' weight.

When we use conceptually related concepts disambiguate concepts, we not only consider the conceptually related concepts and their children, grandchildren and great-grandchildren. For example, X0 has conceptually related concepts Xcr1, Xcr2, Xcr3, Xcr4, and Xcr5. Y0 has conceptually related concepts Ycr1, Ycr2 and Ycr3. Their weights are determined by their original weights and propagated weight from their children, grandchildren and great-grandchildren.

Find all the Cyc concept conceptually related to X0 and Y0, as shown in Figure 3.11. Find all the children, grandchildren, great-grandchildren for all the conceptually related concepts of X0 and Y0 as in Figure 3.12.

The weights of X0 and Y0's conceptually related concepts can be computed as

Wt(ConceptuallyRelated(X))=Wt(Xcr1)+ Wt(Xcr2)+ Wt(Xcr3)+ Wt(Xcr4)+ Wt(Xcr5)

Wt (Xcr1)= Wt(Xcr1)+ 0.1Wt(Xcr1c1)

Wt(Xcr2)= Wt(Xc2)+ 0.1Wt(Xcr1c2) +0.01Wt(Xcr1gc1) +0.01Wt(Xcr1gc2) +0.001 Wt(Xcr1ggc1) +0.001 Wt(Xcr1ggc2)

Wt(Xcr3)= Wt(Xcr1)

Wt(Xcr4)= Wt(Xcr4)+ 0.1Wt(Xcr1c4)

Wt(Xcr5)= Wt(Xcr5)+ 0.1Wt(Xcr1c5)

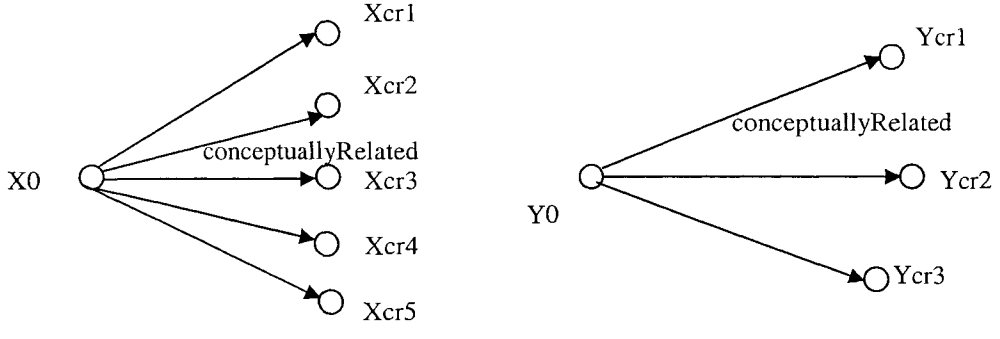


Figure 3.11 Concepts X0, Y0 and Their Conceptually Related Concepts.

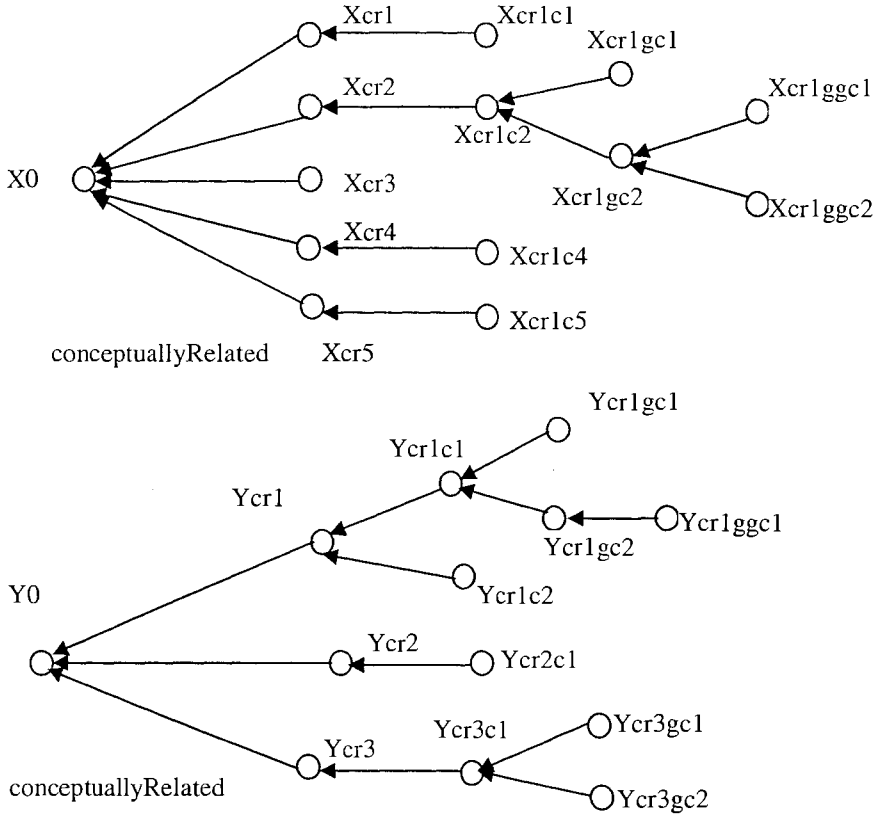


Figure 3.12 Concepts X0, Y0 and Their Conceptually Related Concepts' Hierarchical Structure.

Wt(ConceptuallyRelated(Y))=Wt(Ycr1)+ Wt(Ycr2)+ Wt(Ycr3)

Wt(Ycr1)=Wt(Ycr1)+0.1Wt(Ycr1c1))+0.01Wt(Ycr1gc1) +0.01Wt(Ycr1gc2) +0.001 Wt(Ycr1ggc1)

Wt(Ycr2)= Wt(Ycr2)+ 0.1Wt(Ycr2c1)

Wt(Ycr3)= Wt(Ycr3)+ 0.1Wt(Ycr3c1))+ 0.01Wt(Ycr3gc1) +0.01Wt(Ycr3gc2)

Compare the value of Wt(ConceptuallyRelated(X)) and Wt(Conceptually-Related(Y)), if Wt(ConceptuallyRelated(X)) is greater than Wt(ConceptuallyRelated(Y)), choose X0 as the word corresponding Cyc concept, or else choose Y0.

In our WSD, step by step, we consider the current concepts' occurrence frequency and their children, grandchildren and great-grandchildren, concepts' peer and parent, and

conceptually related concepts and conceptually related concepts' children, grandchildren and great-grandchildren. So we have a good chance to choose the right sense for a word.

3.6 Sentence Extraction

In this project, we extract key sentences based on the weight of the sentences. We provide two sets of output. The first set output is for output sentences ordered by the sentence weights. The other set of output is for output sentences ordered by their order occurred in the original document.

1. The first set of output sentences are computed in the following steps:

(A) Compute the weight for each sentence

For each sentence,

(1) For each phrase or word within a sentence, retrieve its corresponding Cyc concept and retrieve the Cyc concept's weight. If the word is a close-class word, assign its weight as 0.0.

(2) Sum the weight together, average the weight by the term number in the sentence. Term is defined as a word or phrase which has a Cyc corresponding concept. For example, "Louisiana Tech University" has one Cyc corresponding concept #$LouisianaTechUniversity, "Louisiana Tech University" will be considered as one term unit. "Dog" has one Cyc corresponding concept #$LouisianaTechUniversity, "Louisiana Tech University" will be considered as one term unit.

(B) Sort the sentences in descending order according to their weights.

(C) Output the top 5-10 sentences.

2. The second set of output sentences are computed as following steps:

(A2) Is the same as (A)

(B2) Is the same as (B)

(C2) Choose the top 5-10 sentences, reorder the sentences according to their order occurred in the original document.

(D2) Output the rearranged sentences.

3.7 Generate New Sentences

In this dissertation, we generate new sentences based on the sentences' compatibility. Sentences compatibility is built on the assumption that two sentences' subjects, predicates and objects are compatible with each other (Choi, 2008). Two concepts are considered compatible if

a) they are the same concepts;

b) one concept is the other concept's immediate super class;

c) two concepts are conceptually related.

We then Cluster the compatible sentences together and generate a new sentence for each cluster. A new sentence is generated by a method to create a new concept for a class of compatible terms is as follows:

1) If all the concepts are the same, output the original word;

2) If one concept is the immediate super class of other concepts, output this concept's corresponding natural language word.

3) If all of the concepts have the same immediate super class, output this super class's corresponding natural language word.

i) Create a new subject from all the subjects in the cluster using the method above;

ii) Create a new predicate from all the predicates in the cluster using the method above;

iii) Create a new object from all the object s in the cluster using the method above.

Output the new subject, predicate and object as a new sentence.

CHAPTER 4

EXPERIMENT AND RESULTS

In this chapter, we discuss testing process and show the results of our experiments. First, we generated new concepts based on the propagated concept hierarchy. Second, two sets of summarizing experiments were presented. The first part is the summary generation based on sentence extraction. The second part is summary generation based on new sentence generation.

4.1 Extract Key Concepts

The extracted key concepts could be both original mapped concepts and new generated concepts based on concept propagation. We process a file which introduces horses as our source document (PBS 1999). The extracted key concepts ordered by their weights are shown below:

#$Horse
#$Animal
#$BiologicalSpecies
#$Toe
(#$FrequentPerformerFn #$HerdingSomeAnimals)
#$Hoof
#$ancestors
#$possesses
#$Explorer
#$Prairie

4.2 Extract Sentences to Summarize Document

We use Java to interface with Stanford Parser to extract the part-of-speech tagging for each word within a sentence, and to extract phrase tree structure to filter out noun phrases. We use Java to interface with ResearchCyc map words and phrases to Cyc concept. We use SubL language to propagate the concepts' weight along Cyc concept hierarchy.

We ran several tests to evaluate the performance of our system. The testing documents are downloaded from online WebPages. We show a sample of our system output, compare our results with Microsoft AutoSummarizer. The original document is from PBS link (1999). The output from Microsoft AutoSummarizer is shown in Figure 4.1. The output from our system is shown in Figure 4.2. From the comparison, we can see that 80% of the outputs from both systems are the same sentences. For the other 20% of the outputs, some output sentences from our systems are better than those from Microsoft AutoSummarizer output; some output sentences from Microsoft AutoSummarizer are better than those from our system. Our results are comparable to Microsoft AutoSummarizer's result. We tested several other documents, and got the almost the same performance. Consider that we only use one feature: the concept weight, the performance is very encouraging. We can improve our system by adding other features such as title, location and cue words.

Even today, as HORSES shows, tens of thousands of wild horses roam the American West.

In North America, however, horses were wiped out.

Instead, paleontologists have uncovered fossils that show that horse ancestors varied in size: some large early horses gave way later to smaller ones.

The only survivors were horses in Asia and several zebras.

The horse we know today, however, evolved from an ancestor that was quite different.

But the star of the show is the animal that scientists call Equus caballus, the modern horse species that includes everything from miniature Shetland ponies to massive draft horses able to pull astounding loads.

These days, however, researchers have a far more complex picture of horse evolution -- and they have given the dawn horse a much less colorful name.

There are also rare glimpses of the world's most endangered horse, and an inside look at the art of the horse whisperers, the trainers who through their gentle touch can transform a wild bucking bronco into a stately show horse.

So where did the modern horses come from, the ones that spawned America's cowboy myth?

When fossil hunters first discovered the bones of this creature a century ago, they named it Eohippus --"the dawn horse" -- and believed it was the first link in an evolutionary chain that led directly to today's horse.

Figure 4.1 Output From Our Automatic Text Summarization System.

There are also rare glimpses of the world most endangered horse, and an inside look at the art of the horse whisperers, the trainers who through their gentle touch can transform a wild bucking bronco into a stately show horse.

But the star of the show is the animal that scientists call Equus caballus, the modern horse species that includes everything from miniature Shetland ponies to massive draft horses able to pull astounding loads.

The horse we know today, however, evolved from an ancestor that was quite different.

Interestingly, in modern horses, one toe has become the hoof, and the others remain as vestigial bumps higher up the leg.

When fossil hunters first discovered the bones of this creature a century ago, they named it Eohippus? the dawn horse? ? and believed it was the first link in an evolutionary chain that led directly to today horse.

Less than 10,000 years ago, however, many of these horse-like species became extinct, along with other browsing animals such as mammoths.

The only survivors were horses in Asia and several zebras.

In North America, however, horses were wiped out.

So where did the modern horses come from, the ones that spawned America's cowboy myth?

Even today, as HORSES shows, tens of thousands of wild horses roam the American West.

Figure 4.2 Output From Microsoft AutoSummarizer.

4.3 Generate New Sentences

This part demonstrates step by step implementations and results of the summary generation process based on generating new sentences. We generate a simple document and use it to demonstrate sentence generation process.

4.3.1 Generate New Sentences from Manually Composed Text

We manually composed a document which consists of sentences listed in Figure 4.3. Each sentence only consists of one subject, one predicate and one object.

The sentence generation steps are as following:

1. Extract subject, predicate and object triple (SPO) from each sentence and map the subject, predicate and object to their Cyc corresponding concepts based on the context. Save the concept representation of the triples and their original words into ResearchCyc knowledgebase. The SPOs of the above document are shown in Figure 4.4.
2. Compute the number N of new created SPOs in ResearchCyc, create a N*N matrix Commatrix, assign 1 to Commatrix(i,j); if SPO(i)'s subject, predicate

Crows eat celery.
Doves eat eggplant.
Ibis eat potato.
Ravens eat cabbage.
Cats eat dogs.
Cats eat foxes.
Cats eat wolf.
Cats eat greyhounds.
Cats eat terriers.
Cats eat harriers.
Cats eat hyenas.
Cats eat jackals.
Tigers eat venison.
Tigers eat beef.
Tigers eat pork.
Wolves drink strawberries.
Jackals eat blueberries.
Dogs eat banana.
Coyotes eat cherries.

Figure 4.3 Original Document.

```
0 (#$Dove #$EatingEvent (#$FruitFn #$EggplantPlant) (#$TheList "Doves" "eat"
"eggplant" "1"))
1 (#$Ibis #$EatingEvent (#$RootFn #$PotatoPlant) (#$TheList "Ibis" "eat"
"potato" "2"))
2 (#$Cat #$EatingEvent #$Dog (#$TheList "Cats" "eat" "dogs" "4"))
3 (#$Cat #$EatingEvent #$Fox (#$TheList "Cats" "eat" "foxes" "5"))
4 (#$Cat #$EatingEvent #$Wolf (#$TheList "Cats" "eat" "wolf" "6"))
5 (#$Cat #$EatingEvent #$Greyhound-Dog (#$TheList "Cats" "eat" "greyhounds"
"7"))
6 (#$Cat #$EatingEvent #$Terrier-FunctionalGroup (#$TheList "Cats" "eat"
"terriers" "8"))
7 (#$Cat #$EatingEvent #$Harrier-TheDog (#$TheList "Cats" "eat" "harriers"
"9"))
8 (#$Cat #$EatingEvent #$Hyena (#$TheList "Cats" "eat" "hyenas" "10"))
9 (#$Cat #$EatingEvent #$Jackal (#$TheList "Cats" "eat" "jackals" "11"))
10 (#$Tiger #$EatingEvent #$Venison (#$TheList "Tigers" "eat" "venison" "12"))
11 (#$Tiger #$EatingEvent #$Beef (#$TheList "Tigers" "eat" "beef" "13"))
12 (#$Tiger #$EatingEvent #$Pork (#$TheList "Tigers" "eat" "pork" "14"))
13 (#$Wolf #$DrinkingEvent (#$FruitFn #$StrawberryPlant) (#$TheList "Wolves"
"drink" "strawberries" "15"))
14 (#$Jackal #$EatingEvent (#$FruitFn #$BlueberryBush) (#$TheList "Jackals"
"eat" "blueberries" "16"))
15 (#$Dog #$EatingEvent (#$FruitFn #$BananaTree) (#$TheList "Dogs" "eat"
"banana" "17"))
16 (#$Coyote-Animal #$EatingEvent (#$FruitFn #$CherryTree) (#$TheList
"Coyotes" "eat" "cherries" "18"))
```

Figure 4.4 SPO (Subject, Predicate, and Object) Representation of the Text.

and object are compatible to SPO(j)'s corresponding subject, predicate and object, respectively, the measurement of compatibility is as follows:

X and Y are compatible if

a. if X and Y are the same concepts;

b. if X and Y have the same parents;

c. if X is Y's parents, or Y is X's parents;

d. if X is conceptually related to Y or vice versa.

The Compatible Matrix Commatrix of the document is in Figure 4.5.

3. Cluster the SPOs based on the compatibility matrix

The generated clusters are shown in Figure 4.6.

4. Create a new sentence for each cluster

a. Create a new subject

- if all the subjects of the cluster are same, take this subject as the final subject;
- if the subjects of the cluster have the same parents, retrieve their parents as the final subject;
- call getGeneratePhrase() method to output the natural language phrase of the final subject;
- if the phrase is count noun, insert quantifier "Some" before the phrase, and output the phrase;

b. Create a new predicate

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure 4.5 Compatible Matrix.

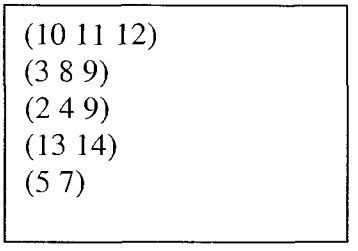


Figure 4.6 Compatible Clusters.

- if all the predicates of the cluster are the same, take this predicate as the final predicate;
- if the predicates of the cluster have the same parents, retrieve their parents as the final predicate;
- call getGeneratePhrase() method to output the natural language phrase of the final predicate.

c. Create a new object

- if all the objects of the cluster are the same, take this object as the final object;
- if the objects of the cluster have the same parents, retrieve their parents as the final object;
- call getGeneratePhrase() method to output the natural language phrase of the final object and output the phrase. The new generated sentences are as shown in Figure 4.7.

"Tigers eat mammal meat."
"Cats eat scavenger."
"Cats eat Canis."
"Some Canis consume fruit."
"Cats eat hound dog."

Figure 4.7 New Generated Sentences for the Document.

4.3.2 Generate New Sentences From Science Articles

To test our text summarization system, we use three science-related articles. The generated results are listed below. We extract some text from the following webpage "http://en.wikipedia.org/wiki/Dog," we get the following results.

(a)Top 10 key concepts generated from the document

#$Dog
#$BiologicalSubspecies
#$Wolf
#$Animal
#$BiologicalSpecies
(#$UnitOfCountFn #$HomoSapiens)
#$possesses
#$Action
#$CanisGenus
#$YearsDuration

(b)New generated sentences

"Dogs be carnivore."

(c) Extracted top 10 sentences

Dogs are highly social animals.
Mixed breed dogs and purebred dogs are both suitable as companions, pets, working dogs , or competitors in dog sports.
Dogs have thicker skins than similarly-sized wolves.
The English word dog , in common usage , refers to the domestic pet dog , Canis lupus familiaris.
The dog has developed into hundreds of varied breeds.
Despite these differences, dogs are able to distinguish dogs from other kinds of

animal.
These considerations affect both pets and the show dogs entered in dog shows.
Some members of the family have "dog "in their common names, such as the Raccoon.
Dog and the African Wild Dog.
Intelligence Dogs are valued for their intelligence.
Dogs require fewer calories to function than wolves.

We use a science article about tree tomato, and get following result (Morton, 1987-A).

(a) Top 10 key concepts generated from the document

#$Fruit
#$Tree-ThePlant
(#$FruitFn #$TomatoPlant)
#$FactoryBuildingComplex
#$Seed
#$TreeBranch
#$Leaf
#$Virus
#$FieldOfStudy
#$Sugar-Table

(b) New generated sentence

"Some plant parts be subject."
"Some external anatomical parts be things."

(c) Extracted top 10 sentences

The tree tomato, Cyphomandra betacea Sendt.
Tree tomato flowers are normally self-pollinating.
It fruits satisfactorily in northern greenhouses.
Red fruits are chosen for the fresh fruit markets because of their appealing color.
In South America and the Caribbean, the fruits are subject to attack by fruit flies Anastrepha sp.
The tree tomato is not tropical but subtropical.
In Haiti it grows and fruits to perfection at 6,000 ft (1,830 m).
In Colombia, the tree tomato has been found to be the preferred host of the tree tomato worm (Neoleucinodes sp) .
Tree tomato mosaic virus causes pale mottling on leaves and sometimes on the fruits which has not been considered a serious disadvantage.
Otherwise, the tree will develop a broad top with fruits only on the outer fringe.

We use a science article about grapefruit, and get following result (Morton, 1987-B).

(a)Top 10 key concepts generated from the document

#$Fruit
(#$FruitFn #$GrapefruitTree)
#$Tree-ThePlant
#$Florida-State
#$Seed
(#$FruitFn #$OrangeTree)
#$Texas-State
#$pulp
#$possesses
#$GeographicalRegion

(b) New generated sentence

"Pulp be colored."
"Fruits be thing."
"Grapefruit be thing."
"Pulp be tangible thing."
"Fruit be round."

These tests show that our system is able to create new key concepts and new sentences to summarize documents.

CHAPTER 5

CONCLUSION AND FUTURE DIRECTION

In this project, we develop a system that can generate an extraction based summary and can generate new sentences to summarize a document. To generate an extraction based summary, we used Stanford parser to tag words and phrases with their part-of-speeches, and used ResearchCyc to map those words and phrases to concepts. We propagate the weight of the concepts along ResearchCyc hierarchy. In addition, we also disambiguate between concepts and annotate each phrase and word with the disambiguated concept. We also label the phrase or word with the concept's weight and sum the weight of each phrase and word in each sentence as the sentence's weight. Then, we extract sentences with some of the highest weights. To generate new sentences to summarize the document, we extract the subject, predicate, and object for each sentence. We build clusters for the compatible sentences and then generate a new sentence for each cluster. Test results show that our sentence extraction method has performance equal to that of currently used by Microsoft systems. Our system is the first system to be able to generate abstract concepts and to create new sentences to summarize documents. Thus not comparison is done on these methods. Our tests show that our approaches are viable and have great potential for future usage and development.

Some future research directions are outlined as follows. For the current project, we used Stanford parser for syntax parsing. The performance of Stanford parser is

86.36% according to F1 measure. It is very close to the current state of the art performance (Klein 2003). However, there are syntax parsers which have better performances such as ENGCG. The performance of ENGCG is 99.7-100% of all words retain the appropriate part-of-speech tagging and about 97-98% of all words retain the appropriate typed dependency tagging. Since the mapping from a word or phrase to ResearchCyc concept is dependent on the part of speech tagging, by using a more accurate syntax parser, the performance of our document summarization system will improve.

For the word sense disambiguation, we only use the global features to make sense disambiguation, which is based on the assumption that a word that occurs in a document will keep the same meaning throughout the document. It simplifies the word sense disambiguation process but might affect the accuracy of the performance. In the future, we can integrate the technique which uses local context to resolve sense ambiguity.

For future improvements, we can use the characteristics of Cyc to make sense disambiguation at sentence level. For most of the Cyc's constant, it has a corresponding defining "microtheory". We can use #$defining-mt predicate to retrieve which microtheory the constant is defined in. All the assertions and constants defined in the microtheory share some assumptions. We can make sense disambiguation based on the algorithm as shown in Figure 5.1.

Cyc encodes important lexical information in various forms of "sem trans" (semantic translation) assertions. The semantic templates specify how the interpretations of a headword and its complements (which includes the subject, in the case of verb should be fused together to represent the semantics for some larger constituent (relative to some #$SubcategorizationFrame). Based on the semantic template, we can

disambiguate word sense on the sentence level. By integrating sentence level sense disambiguation to our global features based sense disambiguation, we can improve the accuracy of our word sense disambiguation to further improve the performance of our system.

Domain specific text summarization systems outperform open domain text summarization. In the future, we can use automatic text classification methods to classify a document into some predefined category, and add the features inherent to the category to our system. The domain knowledge will have use create better summary.

In this project, we use ResearchCyc as our knowledgebase, which consists of about 300,000 concepts and 3,000,000 assertions. Cycorp has another version of the knowledgebase called full Cyc, which consist of over 328,000 concepts and over 3,500,000 assertions. The performance of our word sense disambiguation and new sentences generation process is dependent on the concept hierarchy. The knowledgebase we are using is not well balanced. For future project, we should use a commercial and more powerful version of ResearchCyc called Full Cyc, which will allow our system to generate more and better sentences to summarize a document.

1. Retrieve the defining microtheory for each word in the sentence

   (a)For each word in the sentence,
   Use Cyc predicate #$Denotation, the word and its part-of-speech as arguments to retrieve its Cyc corresponding concepts. (b)For each of the corresponding concepts,
   Use defining-Mt predicate to retrieve the microtheory for which the concept is defined in. Increase the weight of this microtheory by 1.

2. Build a link between two concepts if they are related by any predicate

   For all the concepts retrieved during the last step, for each pair of concept, use subl command "assertions-mention-terms" to check if the two concepts are related; if they are related, build a link between them

3. Word sense disambiguation by the weight the word's defining theory weight

   For each word in the sentence,

   If it has only one corresponding concept, choose that concept, then it is done.

   If it has more than one corresponding concepts,
   For each of the corresponding concepts,
   Retrieve its defining microtheory's weight. Compare the weights if one concept's defining microtheory's weight is higher than all the others, map the word to that concept, it is done, or else

   For each of the corresponding concepts,
   Follow the link built from step 2; follow the links to retrieve the linked concepts.
   For each concept, retrieved the linked concepts' defining microtheory 's weight, and sum them together. Save this sum as the concept's Associated weight. Compare the concept s' associated weight, if one concept's the concept's associated weight is higher than all the others, map the word to that concept.
   If the disambiguation cannot be resolved, leave it to be resolved by global disambiguation system.

Figure 5.1 Sentence Level Word Sense Disambiguation.

APPENDIX A

STOP WORD LIST

Stop Word List

a	ab	about	above	ac	according
across	ads	ae	af	after	afterwards
against	albeit	all	almost	alone	along
already	also	although	always	among	amongst
an	and	another	any	anybody	anyhow
anyone	anything	anyway	anywhere	apart	are
around	as	ask	asked	asks	asp
at	author	av	available	baf	be
became	because	become	becomes	becoming	been
before	beforehand	behind	being	below	beside
besides	between	beyond	bf	biz	both
but	by	ca	can	cannot	canst
cd	cee	certain	cf	cfm	cfrd
cgi	choose	click	cm	co	com
conduct	conducted	conducts	consider	considered	consider
contrariwisecos		could	crd	cu	cx
date	dates	day	days	described	describes
design	designed	designs	determine	determined	dfe
different	discuss	discussed	do	does	doesn't
doing	dont	dost	doth	double	down
dr	dual	due	during	each	eb
ec	either	else	elsewhere	enough	et
etc	eu	even	ever	every	everybody
everyone	everything	everywhere	except	excepted	excepting
exception	exclude	excluding	exclusive	fa	fae
far	farther	farthest	few	ff	find
finds	first	for	formerly	forth	forward
found	free	from	front	further	furthermore
furthest	gcn	general	get	gets	gif
given	gm	go	got	had	halves
hardly	has	hast	hath	have	he
hence	henceforth	her	hers	here	hereabouts
hereafter	hereby	herein	hereto	hereupon	hers

herself him himself hindmost his hither
hitherto how however howsoever hr I
ie if in inasmuch inc include
included including indeed indoors inside insomuch
instead into investigate investigated inward inwards
is it its itself jpg just
kg kind km la last latter
latterly less lest let like liked
likes little low ltd made many
may maybe me mean means meant
meantime meanwhile might more moreover most
mostly mr mrs ms much must
my myself namely need neither net
never nevertheless new news next no
nobody none nonetheless noone nope nor
not nothing notwithstanding now nowadays nowhere
nu obtained of off often oh
ok on once one only onto
or org other others otherwise ought
our ours ourselves out outside over
own owns owned page per performance
performed perhaps pl plenty possible post
present presented presents provide provided provides
quite rather really related report required
results roll round said sake same
sang save saw say see seeing
seem seemed seeming seems seen seldom
selected selves send sent several sfrd
shalt she should show shows shown
sideways significant since slept slew slung
slunk smote so some somebody somehow
someone something sometime sometimes somewhat somewhere
sort spake spat speak speaks spoke
spoken sprang sprung srd stave staves
still studies such suppose supposing ta
take tested than that the thee
their them themselves then thence thenceforth
there thereabout thereabouts thereafter thereby therefore

therein	thereof	thereon	thereto	thereupon	these
they	this	tho	those	thou	though
thrice	through	throughout	thru	thus	thy
thyself	till	time	to	together	too
took	taken	toward	towards	try	types
unable	under	underneath	unless	unlike	until
up	upon	upward	upwards	us	use
uses	used	using	various	very	via
vol	vs	want	wants	wanted	was
we	week	weeks	well	were	what
whatever	whatsoever	when	whence	whenever	whensoever
where	whereabouts	whereafter	whereas	whereat	whereby
wherefore	wherefrom	wherein	whereinto	whereof	whereon
wheresoever	whereto	whereunto	whereupon	wherever	wherewith
whether	whew	which	whichever	whichsoever	while
whilst	whither	who	whoa	whoever	whole
whom	whomever	whomsoever	whose	whosoever	why
will	wilt	with	within	without	worse
worst	would	wow	www	ye	year
years	yet	yippee	you	your	yours
yourself	yourselves				

APPENDIX B

SOURCE CODE

SOURCE CODE

B-1 Retrieve Cyc concept Z's weight

```
( define gou(Z)
 (csetq x
    (ASK-TEMPLATE
      (LIST
      (QUOTE ?freqc))
      (LIST #$freq Z
      (QUOTE ?freqc)) #$ConWeightMt 0 NIL NIL NIL)
     )
 (ret x)
)
```

B-2 Retrieve the concept with highest weight from the list

```
(define maxx(fqlistmx)
(csetq currentmax -1)
(csetq midx 0)
(csetq maxIdxGroup nil)
(csome (fqitem fqlistmx)
 (pif (> fqitem currentmax)
     (progn
         (csetq currentmax fqitem)
         (csetq maxIdxGroup (list midx))
     )
    (pif (equalp fqitem currentmax)    (cpush midx maxIdxGroup)  )
 )
(cinc midx)
)
(ret maxIdxGroup)
)
```

B-3 Increase Cyc concept Z's weight by v

```
(define qa(Z v)
  (csetq x
      (ASK-TEMPLATE
      (LIST
      (QUOTE ?freqc))
      (LIST #$freq value
```

```
(*KE-PURPOSE* NIL))
    (KE-ASSERT-NOW
    (LIST #$freq z b) #$ConWeightMt)))
     (progn
     (WITH-BOOKKEEPING-INFO
     (NEW-BOOKKEEPING-INFO NIL
     (THE-DATE) NIL
     (THE-SECOND))
     (CLET
     (
     (*REQUIRE-CASE-INSENSITIVE-NAME-UNIQUENESS* NIL)
     (*THE-CYCLIST* NIL)
     (*KE-PURPOSE* NIL))
     (KE-UNASSERT-NOW
     (LIST #$freq Z (car (car x))) #$ConWeightMt)))
     (WITH-BOOKKEEPING-INFO
     (NEW-BOOKKEEPING-INFO NIL
     (THE-DATE) NIL
     (THE-SECOND))
     (CLET
     (
     (*THE-CYCLIST* NIL)
     (*KE-PURPOSE* NIL))
     (KE-ASSERT-NOW
     (LIST #$freq Z (+  (car (car x))  v ) )    #$ConWeightMt)))
     )
   )
)
```

B-4 First level propagation along the Cyc hierarchy

```
( define prop1(z)
(csetq x
    (ASK-TEMPLATE
      (LIST
      (QUOTE ?f1)
      (QUOTE ?freqc))
      (LIST #$freq
      (QUOTE ?f1)
      (QUOTE ?freqc)) #$ConWeightMt 0 NIL NIL NIL)
)
 (csetq xc1 0)
  (csome (xs x )  (print (car xs)) (print (car (cdr xs)))
  (csetq yxs
  (REMOVE-DUPLICATES
  (WITH-ALL-MTS
  (MIN-GENLS (car xs))))
  )
  (csetq yinisa
  (REMOVE-DUPLICATES
  (WITH-ALL-MTS
  (MIN-isa (car xs))))
  )
   (pif (> (list-length yinisa) 0)
  (csome (yinisaitem yinisa)
  (cpushnew yinisaitem yxs)
  )
```

```
 )
(csetq xcxs 0)
 (csome (xsxs yxs )  (print xsxs)
    (csetq chazhi
    (ASK-TEMPLATE
    (LIST
    (QUOTE ?ry))
    (LIST #$rongyu  xsxs
    (QUOTE ?ry)) #$ConWeightMt 0 NIL NIL NIL)
   )
 (pif  (cor
     (equalp xsxs #$Collection)
     (equalp xsxs #$Thing)
     (equalp xsxs #$Individual)
     (equalp xsxs #$SetOrCollection)
     (equalp xsxs #$PartiallyIntangibleIndividual )
     (equalp xsxs #$TemporalThing )
     (equalp xsxs #$SomethingExisting )
     (equalp xsxs #$PartiallyTangible)
     (equalp xsxs #$PartiallyIntangible )
     (equalp xsxs #$Intangible)
     (equalp xsxs #$SpatialThing)
     (equalp xsxs #$TemporalThing)
     (equalp xsxs #$SomethingExisting)
     (equalp xsxs #$SpatialThing-Localized)
     (equalp xsxs #$IntangibleIndividual)
     (equalp xsxs #$IntangibleExistingThing)
     (equalp xsxs #$CompositeTangibleAndIntangibleObject)
     (equalp xsxs #$TimeInterval)
     (equalp xsxs #$Situation)
     (equalp xsxs #$Event)
     (equalp xsxs #$PhysicalEvent)
     (equalp xsxs #$Configuration)
     (equalp xsxs #$MathematicalOrComputationalThing)
     (equalp xsxs #$MathematicalThing)
     (equalp xsxs #$MathematicalObject)
     (equalp xsxs #$SetOrCollection)
)    (print chazhi)
  (pif (equalp chazhi nil)
  (WITH-BOOKKEEPING-INFO
  (NEW-BOOKKEEPING-INFO NIL
  (THE-DATE) NIL
  (THE-SECOND))
  (CLET
   (
    (*THE-CYCLIST* NIL)
    (*KE-PURPOSE* NIL))
    (KE-ASSERT-NOW
    (LIST #$rongyu xsxs (* z (car (cdr xs))    )) #$ConWeightMt)))
   (progn
    (WITH-BOOKKEEPING-INFO
    (NEW-BOOKKEEPING-INFO NIL
    (THE-DATE) NIL
    (THE-SECOND))
    (CLET
    (
```

```
    (*REQUIRE-CASE-INSENSITIVE-NAME-UNIQUENESS* NIL)
    (*THE-CYCLIST* NIL)
    (*KE-PURPOSE* NIL))
    (KE-UNASSERT-NOW
    (LIST #$rongyu xsxs (car (car chazhi))) #$ConWeightMt)))
    (WITH-BOOKKEEPING-INFO
    (NEW-BOOKKEEPING-INFO NIL
    (THE-DATE) NIL
    (THE-SECOND))
    (CLET
    (
    (*THE-CYCLIST* NIL)
    (*KE-PURPOSE* NIL))
    (KE-ASSERT-NOW
    (LIST #$rongyu xsxs (+ (* z (car (cdr xs))) (car (car chazhi)))
     ) #$ConWeightMt)))
    )
)) ))
)
)
```

B-5 Second level propagation along the Cyc hierarchy

```
( define propf1(z)
(csetq x
   (ASK-TEMPLATE
    (LIST
     (QUOTE ?f1)
     (QUOTE ?freqc))
     (LIST #$rongyu
     (QUOTE ?f1)
     (QUOTE ?freqc)) #$ConWeightMt 0 NIL NIL NIL)
)
 (csome (xrs x )
     (csetq pinlv
     (ASK-TEMPLATE
     (LIST
     (QUOTE ?freqc))
     (LIST #$freq (car xrs)
     (QUOTE ?freqc)) #$ConWeightMt 0 NIL NIL NIL))
     (pif (equalp pinlv nil)
      (WITH-BOOKKEEPING-INFO
     (NEW-BOOKKEEPING-INFO NIL
     (THE-DATE) NIL
     (THE-SECOND))
    (CLET
    (
    (*THE-CYCLIST* NIL)
    (*KE-PURPOSE* NIL))
    (KE-ASSERT-NOW
    (LIST #$freq (car xrs)  (car (cdr xrs))  ) #$ConWeightMt)))
    (progn
     (WITH-BOOKKEEPING-INFO
     (NEW-BOOKKEEPING-INFO NIL
     (THE-DATE) NIL
     (THE-SECOND))
     (CLET
```

```
      (
      (*REQUIRE-CASE-INSENSITIVE-NAME-UNIQUENESS* NIL)
      (*THE-CYCLIST* NIL)
      (*KE-PURPOSE* NIL))
      (KE-UNASSERT-NOW
       (LIST #$freq (car xrs) (car (car pinlv))) #$ConWeightMt)))
       (WITH-BOOKKEEPING-INFO
       (NEW-BOOKKEEPING-INFO NIL
       (THE-DATE) NIL
       (THE-SECOND))
       (CLET
       (
       (*THE-CYCLIST* NIL)
       (*KE-PURPOSE* NIL))
       (KE-ASSERT-NOW
       (LIST #$freq (car xrs) (+  (car (cdr xrs)) (car (car pinlv)))
      ) #$ConWeightMt)))
  )
)
(csetq yxs
  (REMOVE-DUPLICATES
  (WITH-ALL-MTS
  (MIN-GENLS (car xrs))))
)
  (csetq yinisa
  (REMOVE-DUPLICATES
  (WITH-ALL-MTS
  (MIN-isa (car xrs))))
  )
    (pif (> (list-length yinisa) 0)
  (csome (yinisaitem yinisa)
  (cpush yinisaitem yxs)
  )
  )
(csetq xcxs 0)
  (csome (xsxs yxs )
  (csetq chazhi
    (ASK-TEMPLATE
    (LIST
    (QUOTE ?ry))
    (LIST #$rongyu1  xsxs
    (QUOTE ?ry)) #$ConWeightMt 0 NIL NIL NIL)
     )
 (pif (cor
      (equalp xsxs #$Collection)
      (equalp xsxs #$Thing)
      (equalp xsxs #$Individual)
      (equalp xsxs #$SetOrCollection)
      (equalp xsxs #$PartiallyIntangibleIndividual )
      (equalp xsxs #$TemporalThing )
      (equalp xsxs #$SomethingExisting )
      (equalp xsxs #$PartiallyTangible)
      (equalp xsxs #$PartiallyIntangible )
      (equalp xsxs #$Intangible)
      (equalp xsxs #$SpatialThing)
      (equalp xsxs #$TemporalThing)
```

```
      (equalp xsxs #$SomethingExisting)
      (equalp xsxs #$SpatialThing-Localized)
      (equalp xsxs #$IntangibleIndividual)
      (equalp xsxs #$IntangibleExistingThing)
      (equalp xsxs #$CompositeTangibleAndIntangibleObject)
      (equalp xsxs #$TimeInterval)
      (equalp xsxs #$Situation)
      (equalp xsxs #$Event)
      (equalp xsxs #$PhysicalEvent)
      (equalp xsxs #$Configuration)
      (equalp xsxs #$MathematicalOrComputationalThing)
      (equalp xsxs #$MathematicalThing)
      (equalp xsxs #$MathematicalObject)
      (equalp xsxs #$SetOrCollection)
)
 (print chazhi)
 (pif (equalp chazhi nil)
    (WITH-BOOKKEEPING-INFO
    (NEW-BOOKKEEPING-INFO NIL
    (THE-DATE) NIL
    (THE-SECOND))
    (CLET
   (
    (*THE-CYCLIST* NIL)
    (*KE-PURPOSE* NIL))
    (KE-ASSERT-NOW
    (LIST #$rongyu1 xsxs (* z (car (cdr xrs))    )) #$ConWeightMt)))
    (progn
     (WITH-BOOKKEEPING-INFO
     (NEW-BOOKKEEPING-INFO NIL
     (THE-DATE) NIL
     (THE-SECOND))
     (CLET
     (
      (*REQUIRE-CASE-INSENSITIVE-NAME-UNIQUENESS* NIL)
      (*THE-CYCLIST* NIL)
      (*KE-PURPOSE* NIL))
      (KE-UNASSERT-NOW
      (LIST #$rongyu1 xsxs (car (car chazhi))) #$ConWeightMt)))
    (WITH-BOOKKEEPING-INFO
    (NEW-BOOKKEEPING-INFO NIL
    (THE-DATE) NIL
    (THE-SECOND))
    (CLET
    (
    (*THE-CYCLIST* NIL)
    (*KE-PURPOSE* NIL))
    (KE-ASSERT-NOW
    (LIST #$rongyu1 xsxs (+ (* z (car (cdr xrs))) (car (car chazhi)))
     ) #$ConWeightMt)))
 ))) ) )
   (csetq finalrongyu
      (ASK-TEMPLATE
      (LIST
      (QUOTE ?f1)
      (QUOTE ?freqc))
```

```
      (LIST #$rongyu1
      (QUOTE ?f1)
      (QUOTE ?freqc)) #$ConWeightMt 0 NIL NIL NIL)
)
 (csetq xf1 0)
(csome (xfs finalrongyu )
    (csetq fpinlv
    (ASK-TEMPLATE
    (LIST
    (QUOTE ?freqc))
    (LIST #$freq (car xfs)
    (QUOTE ?freqc)) #$ConWeightMt 0 NIL NIL NIL) )
   (pif (equalp fpinlv nil)
   (WITH-BOOKKEEPING-INFO
  (NEW-BOOKKEEPING-INFO NIL
  (THE-DATE) NIL
  (THE-SECOND))
  (CLET
  (
   (*THE-CYCLIST* NIL)
   (*KE-PURPOSE* NIL))
   (KE-ASSERT-NOW
   (LIST #$freq (car xfs)  (car (cdr xfs))  ) #$ConWeightMt)))
   (progn
    (WITH-BOOKKEEPING-INFO
    (NEW-BOOKKEEPING-INFO NIL
    (THE-DATE) NIL
    (THE-SECOND))
    (CLET
    (
    (*REQUIRE-CASE-INSENSITIVE-NAME-UNIQUENESS* NIL)
    (*THE-CYCLIST* NIL)
    (*KE-PURPOSE* NIL))
    (KE-UNASSERT-NOW
    (LIST #$freq (car xfs) (car (car fpinlv))) #$ConWeightMt)))
   (WITH-BOOKKEEPING-INFO
   (NEW-BOOKKEEPING-INFO NIL
   (THE-DATE) NIL
   (THE-SECOND))
   (CLET
   (
    (*THE-CYCLIST* NIL)
    (*KE-PURPOSE* NIL))
    (KE-ASSERT-NOW
    (LIST #$freq (car xfs) (+  (car (cdr xfs)) (car (car fpinlv)))
     ) #$ConWeightMt)))   )) )  ) )
```

B-6 Third level propagation along the Cyc hierarchy

```
( define propf2(z)
(
clet
(
(x
    (ASK-TEMPLATE
      (LIST
      (QUOTE ?f1)
      (QUOTE ?freqc))
```

```
      (LIST #$rongyu1
      (QUOTE ?f1)
      (QUOTE ?freqc)) #$ConWeightMt 0 NIL NIL NIL)
 )
)
 (csetq xr1 0)
 (csome (xrs x )
  (csetq yxs
  (REMOVE-DUPLICATES
  (WITH-ALL-MTS
  (MIN-GENLS (car xrs))))
)
  (csetq yinisa
      (REMOVE-DUPLICATES
      (WITH-ALL-MTS
      (MIN-isa (car xrs))))
  )
   (pif (> (list-length yinisa) 0)
  (csome (yinisaitem yinisa)
  (cpush yinisaitem yxs)
  )
  )
(csetq xcxs 0)
  (csome (xsxs yxs )
  (print xsxs)
 (csetq chazhi
    (ASK-TEMPLATE
      (LIST
      (QUOTE ?ry))
      (LIST #$rongyu2  xsxs
      (QUOTE ?ry)) #$ConWeightMt 0 NIL NIL NIL)
     )
   (print chazhi)
(pif
(cor
      (equalp xsxs #$Collection)
      (equalp xsxs #$Thing)
      (equalp xsxs #$Individual)
      (equalp xsxs #$SetOrCollection)
      (equalp xsxs #$PartiallyIntangibleIndividual )
      (equalp xsxs #$TemporalThing )
      (equalp xsxs #$SomethingExisting )
      (equalp xsxs #$PartiallyTangible)
      (equalp xsxs #$PartiallyIntangible )
      (equalp xsxs #$Intangible)
      (equalp xsxs #$SpatialThing)
      (equalp xsxs #$TemporalThing)
      (equalp xsxs #$SomethingExisting)
      (equalp xsxs #$SpatialThing-Localized)
      (equalp xsxs #$IntangibleIndividual)
      (equalp xsxs #$IntangibleExistingThing)
      (equalp xsxs #$CompositeTangibleAndIntangibleObject)
      (equalp xsxs #$TimeInterval)
      (equalp xsxs #$Situation)
      (equalp xsxs #$Event)
      (equalp xsxs #$PhysicalEvent)
```

```
    (equalp xsxs #$Configuration)
    (equalp xsxs #$MathematicalOrComputationalThing)
    (equalp xsxs #$MathematicalThing)
    (equalp xsxs #$MathematicalObject)
    (equalp xsxs #$SetOrCollection).
)
 (print chazhi)
(pif (equalp chazhi nil)
(WITH-BOOKKEEPING-INFO
(NEW-BOOKKEEPING-INFO NIL
(THE-DATE) NIL
(THE-SECOND))
(CLET
 (
  (*THE-CYCLIST* NIL)
  (*KE-PURPOSE* NIL))
  (KE-ASSERT-NOW
  (LIST #$rongyu2 xsxs (* z (car (cdr xrs))    )) #$ConWeightMt)))
  (progn
   (WITH-BOOKKEEPING-INFO
   (NEW-BOOKKEEPING-INFO NIL
   (THE-DATE) NIL
   (THE-SECOND))
   (CLET
   (
   (*REQUIRE-CASE-INSENSITIVE-NAME-UNIQUENESS* NIL)
   (*THE-CYCLIST* NIL)
   (*KE-PURPOSE* NIL))
   (KE-UNASSERT-NOW
   (LIST #$rongyu2 xsxs (car (car chazhi))) #$ConWeightMt)))
   (WITH-BOOKKEEPING-INFO
   (NEW-BOOKKEEPING-INFO NIL
   (THE-DATE) NIL
   (THE-SECOND))
   (CLET
   (
   (*THE-CYCLIST* NIL)
   (*KE-PURPOSE* NIL))
   (KE-ASSERT-NOW
   (LIST #$rongyu2 xsxs (+ (* z (car (cdr xrs))) (car (car chazhi)))
    ) #$ConWeightMt)))
 )
)
)
 (+ xcxs 1)
 (pwhen (< xcxs (list-length yxs)) )
 )
 )
  (csetq finalrongyu
    (ASK-TEMPLATE
    (LIST
    (QUOTE ?f1)
    (QUOTE ?freqc))
    (LIST #$rongyu2
    (QUOTE ?f1)
    (QUOTE ?freqc)) #$ConWeightMt 0 NIL NIL NIL)
```

```
)
 (csetq xf1 0)
 (csome (xfs finalrongyu )
 (csetq fpinlv
 (ASK-TEMPLATE
    (LIST
    (QUOTE ?freqc))
    (LIST #$freq (car xfs)
    (QUOTE ?freqc)) #$ConWeightMt 0 NIL NIL NIL)
 )
 (pif (equalp fpinlv nil)
  (WITH-BOOKKEEPING-INFO
  (NEW-BOOKKEEPING-INFO NIL
  (THE-DATE) NIL
  (THE-SECOND))
  (CLET
  (
    (*THE-CYCLIST* NIL)
    (*KE-PURPOSE* NIL))
    (KE-ASSERT-NOW
    (LIST #$freq (car xfs)  (car (cdr xfs))  ) #$ConWeightMt)))
  (progn
    (WITH-BOOKKEEPING-INFO
    (NEW-BOOKKEEPING-INFO NIL
    (THE-DATE) NIL
    (THE-SECOND))
    (CLET
    (
    (*REQUIRE-CASE-INSENSITIVE-NAME-UNIQUENESS* NIL)
    (*THE-CYCLIST* NIL)
    (*KE-PURPOSE* NIL))
    (KE-UNASSERT-NOW
    (LIST #$freq (car xfs) (car (car fpinlv))) #$ConWeightMt)))
  (WITH-BOOKKEEPING-INFO
  (NEW-BOOKKEEPING-INFO NIL
  (THE-DATE) NIL
  (THE-SECOND))
  (CLET
  (
    (*THE-CYCLIST* NIL)
    (*KE-PURPOSE* NIL))
    (KE-ASSERT-NOW
    (LIST #$freq (car xfs) (+  (car (cdr xfs)) (car (car fpinlv)))
     ) #$ConWeightMt)))
  )
 )
 )
 )
 )
```

B-7 Word sense disambiguation

```
(define wsd(ar)
 (csetq fqlist nil)
(csome (xar ar)
   (csetq pinlv
    (ASK-TEMPLATE
```

```
    (LIST
    (QUOTE ?freqc))
    (LIST #$freq xar
    (QUOTE ?freqc)) #$ConWeightMt  0 NIL NIL NIL)
   )
  (pif (equalp fqlist nil)
    (csetq fqlist (car pinlv))
    (csetq fqlist (cpush (car (car pinlv)) fqlist ))
  )
)
 (print fqlist)
 (csetq fqlist1 nil)
 (csome (fql fqlist)
 (cpush fql fqlist1)
 )
 (print "fqlist1=") (print fqlist1)
(csetq currentlevel (maxx fqlist1))
(print currentlevel)
 (pif (equalp 1 (list-length currentlevel))
 (progn
  (csetq conweightpair nil)
  (cpush   (nth (car currentlevel) fqlist1) conweightpair)
  (cpush   (nth (car currentlevel) ar) conweightpair)
  (ret conweightpair)
 )
(progn
  (ret (parentwsd currentlevel ar fqlist1))
  )
)
)
```

B-8 Parent level word sense disambiguation

```
(define parentwsd (currentlevel ar fqlist1)
  (csetq maxparentGP (MAKE-VECTOR (list-length currentlevel)) )
  (csetq maxparentmax (MAKE-VECTOR (list-length currentlevel)) )
  (csetq maxparentweight (MAKE-VECTOR (list-length currentlevel)) )
  (csetq idx 0)
  (csome    (clitem  currentlevel )
   (csetq pitem
   (REMOVE-DUPLICATES
   (WITH-ALL-MTS
   (MIN-GENLS   (nth clitem ar)   )))
   )
   (csetq yinisa
   (REMOVE-DUPLICATES
   (WITH-ALL-MTS
   (MIN-isa (nth clitem ar) )))
   )
   (pif (> (list-length yinisa) 0)
   (csome (yinisaitem yinisa)
   (cpush yinisaitem pitem)
   )
   )
    (SET-AREF maxparentGP idx pitem)
     (pif (equalp pitem nil)
     (SET-AREF maxparentweight idx 0)
     (progn
```

```
(pif (equalp (list-length pitem) 1)
(progn
(csetq pinlvp
(ASK-TEMPLATE
(LIST
(QUOTE ?freqc))
(LIST #$freq  (car pitem)
(QUOTE ?freqc)) #$ConWeightMt  0 NIL NIL NIL)
)
(SET-AREF maxparentmax idx 0)
(SET-AREF maxparentweight idx (car (car pinlvp)))
)
(progn
(csetq pfqlist nil)
(csome (xarp pitem)
    (csetq ppinlv
          (ASK-TEMPLATE
        (LIST
          (QUOTE ?freqc))
          (LIST #$freq xarp
          (QUOTE ?freqc)) #$ConWeightMt  0 NIL NIL NIL)
    )
    (pif (equalp pfqlist nil)
           (pif (equalp ppinlv nil)
             (csetq pfqlist (cpush 0 pfqlist ))
               (csetq pfqlist (car ppinlv))
             )
         (pif (equalp ppinlv nil)
             (csetq pfqlist (cpush 0 pfqlist ))
             (csetq pfqlist (cpush (car (car ppinlv)) pfqlist ))
         )
   )
  )
  (csetq pfqlist1 nil)
   (csome (pfql pfqlist)
    (cpush pfql pfqlist1)
    )
   (csetq parentlevel (maxx pfqlist1))
   (SET-AREF maxparentmax idx parentlevel)
   (SET-AREF maxparentweight  idx (nth (car parentlevel) pfqlist1) )
 )  )  )  )
 (cinc idx)
)
(csetq xinzu nil)
(csetq qishi (- (list-length currentlevel) 1))
(cdotimes (idx1  (list-length currentlevel))
    (pif (equalp (aref maxparentweight (- qishi idx1)) nil)
       (cpush 0 xinzu)
       (cpush (aref maxparentweight (- qishi idx1) ) xinzu)
  )
 )
 (csetq plevelresult (maxx xinzu))
 (pif (equalp (list-length plevelresult) 1)
   (progn
          (csetq conweightpairp nil)
          (cpush   (nth  (nth (car plevelresult) currentlevel)  fqlist1)  conweightpairp)
```

```
            (cpush   (nth  (nth (car plevelresult) currentlevel) ar)   conweightpairp)
          (ret conweightpairp)
    )
    (progn
        (csetq gran nil)
        (csome (xyz plevelresult)
           (cpush (nth  (nth xyz currentlevel) ar)  gran)
        )
       (ret (conceptrelate  gran))
    )
  )
)
```

B-9 Word sense disambiguation by conceptuallyrelated terms

```
(define conceptrelate(grandlevelresult)
  (csetq  cepgroup (MAKE-VECTOR (list-length grandlevelresult)))
  (csetq  cepgroupweight (MAKE-VECTOR (list-length grandlevelresult)))
  (csetq cepcondidate (MAKE-VECTOR (list-length grandlevelresult)))
   (cdotimes (crelidx (list-length grandlevelresult))
   (set-aref cepcondidate crelidx (nth crelidx  grandlevelresult ))
   )
   (csetq grandlevelsize  (list-length grandlevelresult))
   (cdotimes (cepidx grandlevelsize)
            (csetq cepcluster
               (ASK-TEMPLATE
                     (LIST
                     (QUOTE ?relaterm))
                     (LIST #$conceptuallyRelated  (aref cepcondidate  cepidx )
                     (QUOTE ?relaterm)) #$EverythingPSC   0 NIL NIL NIL)
            )
              (csetq cepcluster1
               (ASK-TEMPLATE
                     (LIST
                     (QUOTE ?relaterm))
                     (LIST #$conceptuallyRelated (QUOTE ?relaterm)
                        (aref cepcondidate  cepidx ) ) #$EverythingPSC  0 NIL NIL NIL)
            )
      (csome (cp cepcluster1 )
       (csetq bflag 0)
       (csome (cpo cepcluster)
       (pif (equalp (car cp) (car cpo))
       (csetq bflag 1)
       )
       )
      (pif (equalp bflag 0)   (cpush cp cepcluster)      )
   )
            (csetq cweight 0)
            (csome (cepitem cepcluster)
              (csetq cpinlv
                     (ASK-TEMPLATE
                      (LIST
                      (QUOTE ?freqc))
                      (LIST #$freq (car cepitem)
                      (QUOTE ?freqc)) #$ConWeightMt  0 NIL NIL NIL)
               )
              (pif (equalp cpinlv nil)
```

```
                    (print "NOne")
                    (csetq cweight (+ cweight (car (car cpinlv))))
                  )
            )
            (SET-AREF cepgroupweight cepidx cweight)
)
 (csetq cepzu nil)
(csetq cqishi (- (list-length plevelresult) 1) )
(cdotimes (cidx1 (list-length plevelresult) )
 (pif (equalp (aref cepgroupweight (- cqishi cidx1)) nil)
  (cpush 0 cepzu)
  (cpush (aref cepgroupweight  (- cqishi cidx1) ) cepzu)
 )
)
(csetq ceplevelresult (maxx cepzu))
 (csetq conweightpair nil)
 (csetq pinlv
 (ASK-TEMPLATE
 (LIST
 (QUOTE ?freqc))
 (LIST #$freq (nth (car ceplevelresult) grandlevelresult)
 (QUOTE ?freqc)) #$ConWeightMt  0 NIL NIL NIL)
 )
  (cpush  (car  (car pinlv)) conweightpair)
  (cpush   (nth (car ceplevelresult) grandlevelresult) conweightpair)
  (ret conweightpair)
)
```

B-10 Cluster sentences and generate new sentences

```
(define clusterspo(spo  clusterqueue)
  (cdolist (cqueue clusterqueue)
    (csetq plist nil)
    (cdolist (cq0 cqueue)
          (csetq plistx nil)
          (cdolist (cq1 cqueue)
                (pif (equalp (compatible  (nth cq0 spo) (nth cq1 spo)) 1)
                     (cpushnew cq1 plistx)
                )
               )
            (csetq plist (append plist (list plistx)))
    )
    (csetq maxlen 1)
    (csetq maxlist (nth 0 plist))
    (cdolist (pl plist)
       (pif (>= (list-length pl ) maxlen)
          (progn
             (csetq maxlen (list-length pl ))
             (csetq maxlist pl)
        )
    )
    )
    (csetq spl nil)
    (csetq ppl nil)
    (csetq opl nil)
    (cdolist (pi maxlist)
        (cpushnew (first (nth pi spo))  spl )
```

```
        (cpushnew (second (nth pi spo)) ppl )
        (cpushnew (third (nth pi spo)) opl )
    )
(csetq gened nil)
    (pif (> (list-length spl) 1)
       (csetq gened  (list (gen spl)))
       (csetq gened  (list spl))
    )
    (pif (> (list-length ppl) 1)
       (csetq gened  (append  gened  (list  (gen ppl))))
       (csetq gened  (append  gened  (list ppl)))
    )

   (pif (> (list-length opl) 1)
       (csetq gened  (append  gened  (list  (gen opl))))
       (csetq gened  (append  gened  (list opl)))
    )
  )
    (csetq Generalizedsentence nil)
    (csetq kongge " ")
    (pif (equalp (list-length (first gened )) 1)
       (csetq  Generalizedsentence (second  (fourth (nth  (first maxlist) spo))))
        (progn
            (csetq Generalizedsentence "Some ")
            (csetq Generalizedsentence (cconcatenate  Generalizedsentence  (cum (generate-
                phrase (second (first  gened ))))))
       )
    )
        (csetq  Generalizedsentence (string-capitalize  Generalizedsentence 0 1))
        (csetq Generalizedsentence (cconcatenate  Generalizedsentence kongge))
       (pif (equalp (list-length (second gened)) 1)
                (csetq Generalizedsentence (cconcatenate  Generalizedsentence  (generate-phrase
                   (first (second  gened )))))
              (csetq Generalizedsentence (cconcatenate  Generalizedsentence  (generate-phrase
                   (second (second  gened )))))
        )
        (csetq Generalizedsentence (cconcatenate  Generalizedsentence kongge))
       (pif (equalp (list-length (third gened )) 1)
                (csetq  Generalizedsentence  (cconcatenate  Generalizedsentence  (fourth (fourth
                    (nth  (first maxlist) spo)))))
             (csetq Generalizedsentence (cconcatenate  Generalizedsentence  (generate-
                  phrase (second (third  gened )))))
        )
      (gen maxlist)
  )
)
```

B-11 Add plural forms to the generated concept

```
(define cum(Generalizedsentence)
        (csetq oriphrase Generalizedsentence)
        (csetq sublen (search " " (reverse oriphrase)))
        (pif (equalp sublen nil)
             (csetq subseqlen nil)
              (csetq subseqlen (- (length oriphrase) (search " " (reverse oriphrase))))
         )
        (pif (equalp subseqlen nil)
```

```
            (csetq mainnoun oriphrase)
            (csetq mainnoun (subseq oriphrase subseqlen))
        )
      (csetq  Noun_TheWord
          (ASK-TEMPLATE
          (LIST   (QUOTE ?f1)    )
          (LIST #$wordForms   (QUOTE ?f1)   #$nounStrings   mainnoun )
          #$GeneralEnglishMt 0 NIL NIL NIL
           )
    )
    (csetq  word_POS
    (ASK-TEMPLATE
    (LIST
    (QUOTE ?f1)
    )
    (LIST #$posForms
    (caar Noun_TheWord)
     (QUOTE ?f1)
    ) #$GeneralEnglishMt 0 NIL NIL NIL)
    )
    (csetq countnoun1 0)
      (csome (itt word_POS)
        (print (car itt))
       (pif (equalp (car itt) #$CountNoun)
          (csetq countnoun1 1)
       )
      )
    (pif (equalp countnoun1 1)
       (progn
         (csetq word_PLlist
         (ASK-TEMPLATE
         (LIST
          (QUOTE ?f1)
          )
         (LIST #$plural
         (caar Noun_TheWord)
         (QUOTE ?f1)
         ) #$GeneralEnglishMt 0 NIL NIL NIL)
         )
         (csetq  word_PL  (caar word_PLlist)  )
         (pif (equalp subseqlen nil)
            (ret word_PL)
             (ret (cconcatenate (subseq Generalizedsentence 0 subseqlen) word_PL ))
          )
       )
       (ret Generalizedsentence)
    )
)
```

B-12 Check if two concepts are compatible

```
(define compatpartP (partx party)
(csetq comflag 0)
 (pif (equalp partx party)
    (progn
    (csetq comflag 1)
    (ret comflag)
```

```
    )
)
(csetq parentgenx
  (REMOVE-DUPLICATES
  (WITH-ALL-MTS
  (MIN-GENLS  partx  )))
  )
(cpushnew partx parentgenx)
(csetq parentgeny
    (REMOVE-DUPLICATES
    (WITH-ALL-MTS
    (MIN-GENLS  party)))
)
(cpushnew party parentgeny)
(csome (gx parentgenx )
     (csome (gy parentgeny)
       (pif (equalp gx gy)
         (progn
          (csetq comflag 1)
          (ret comflag)
         )
       )
     )
)
 (pif (CYC-QUERY (quote (#$nearestGenls partx party)) #$EverythingPSC)
    (csetq comflag 1)
    (pif (CYC-QUERY (quote (#$nearestGenls party partx)) #$EverythingPSC)
       (progn
         (csetq comflag 1)
         (ret comflag)
    )
 )
)
(pif (CYC-QUERY (quote (#$nearestIsa partx party)) #$EverythingPSC)
    (csetq comflag 1)
    (pif (CYC-QUERY (quote (#$nearestIsa party partx)) #$EverythingPSC)
       (progn
          (csetq comflag 1)
          (ret comflag)
    ) )) (ret comflag)
)
```

B-13 Add new element to the cluster

```
(define clusterspo(spo  clusterqueue)
  (cdolist (cqueue clusterqueue)
    (print 'new_cluster)  (print cqueue)
    (csetq plist nil)
    (cdolist (cq0 cqueue)
       csetq plistx nil)
       (cdolist (cq1 cqueue)
              (pif (equalp (compatible  (nth cq0 spo) (nth cq1 spo)) 1)
                     (cpushnew cq1 plistx)
              )
            )
           (csetq plist (append plist (list plistx)))
     )
```

```
)
    (csetq maxlen 1)
    (csetq maxlist (nth 0 plist))
    (cdolist (pl plist)
    (pif (>= (list-length pl ) maxlen)
    (progn
    (csetq maxlen (list-length pl ))
    (csetq maxlist pl)
    )
    )
    )
    (csetq spl nil)
    (csetq ppl nil)
    (csetq opl nil)
    (cdolist (pi maxlist)
       (cpushnew (first (nth pi spo))  spl )
       (cpushnew (second (nth pi spo)) ppl )
       (cpushnew (third (nth pi spo))  opl )
    )
    (csetq gened nil)
    (pif (> (list-length spl) 1)
        (csetq gened   (list (gen spl)))
        (csetq gened   (list spl))
    )
    (pif (> (list-length ppl) 1)
       (csetq gened   (append   gened  (list  (gen ppl))))
            (csetq gened   (append   gened   (list ppl)))
    )

    (pif (> (list-length opl) 1)
       (csetq gened   (append   gened  (list  (gen opl))))
            (csetq gened   (append   gened   (list opl)))
    )
(csetq Generalizedsentence nil)
 (csetq kongge " ")
 (pif (equalp (list-length (first gened )) 1)
          (csetq  Generalizedsentence (generate-phrase (car (first  gened ))))
           (progn
              (csetq Generalizedsentence "Some ")
              (csetq Generalizedsentence (cconcatenate  Generalizedsentence
            (cum (generate-phrase (second (first  gened ))))))
           )
 )
(csetq  Generalizedsentence (string-capitalize  Generalizedsentence 0 1))
(csetq Generalizedsentence (cconcatenate  Generalizedsentence kongge))
(pif (equalp (list-length (second gened)) 1)
    (progn
            (csetq Generalizedsentence (cconcatenate  Generalizedsentence
          (generate-phrase (first (second  gened )))))
    )
   (progn
            (csetq Generalizedsentence (cconcatenate  Generalizedsentence
          (generate-phrase (second (second  gened )))))
    )
 )
   (csetq Generalizedsentence (cconcatenate  Generalizedsentence kongge))
```

```
  (pif (equalp (list-length (third gened )) 1)
          (csetq Generalizedsentence (cconcatenate  Generalizedsentence
     (generate-phrase (first (third  gened )))))
          (progn
            (csetq Generalizedsentence (cconcatenate   Generalizedsentence " some "))
            (csetq Generalizedsentence (cconcatenate   Generalizedsentence  (cum
                (generate-phrase   (second (third  gened ))))))
     )
  )
     (gen maxlist)
  )
)
```

B-14 Check if partx and party is parent-child relation

```
(define genparent(partx party)
(csetq parentgenx
  (REMOVE-DUPLICATES
  (WITH-ALL-MTS
  (MIN-GENLS   partx   )))
  )
(csetq parentgeny
  (REMOVE-DUPLICATES
  (WITH-ALL-MTS
  (MIN-GENLS   party)))
  )
(csome (gx parentgenx )
     (csome (gy parentgeny)
       (pif (equalp gx gy)
         (ret gx)
       )
     )
 )
 (ret nil)
)
```

B-15 Generate a new generalized concept from a list of concepts

```
(define  gen(maxlist)
(csetq qx nil)
(cpushnew (first maxlist) qx)
(cdolist (partx maxlist)
   (cdolist (party qx)
      (pif (equalp partx party)
         (cpushnew partx qx)
         (progn
            (csetq tparent  (genparent partx party))
            (pif (equalp  tparent  nil)
               (print 'nil)
               (cpushnew tparent qx)
            )
         )
      )
   )
)
(pif (> (list-length qx) 1)
)
(csome  (mxt maxlist )
```

```
  (csetq parentgenx1
  (REMOVE-DUPLICATES
  (WITH-ALL-MTS
  (MIN-GENLS mxt   )))
)
(csome (px1 parentgenx1)
  (WITH-BOOKKEEPING-INFO
  (NEW-BOOKKEEPING-INFO NIL
  (THE-DATE) NIL
  (THE-SECOND))
  (CLET
  (
  (*THE-CYCLIST* NIL)
  (*KE-PURPOSE* NIL))
  (KE-ASSERT-NOW
  (LIST #$xkid px1 mxt) #$TextSummarizationMt)))
  )
  )
   (csetq liuye1
    (ASK-TEMPLATE
     (LIST
      (QUOTE ?f1)
      (QUOTE ?freqc))
      (LIST #$xkid
      (QUOTE ?f1)
      (QUOTE ?freqc)) #$TextSummarizationMt 0 NIL NIL NIL)
    )
 (csetq depot nil)
 (csome (lx liuye1)
 (cpushnew  (FIRST lx) depot)
 )
 (csetq maxw 1)
 (csetq maxitem (first maxlist ))
  (csome (dpt depot)
    (csetq liuy1
    (ASK-TEMPLATE
     (LIST
      (QUOTE ?freqc))
      (LIST #$xkid
       dpt
      (QUOTE ?freqc)) #$TextSummarizationMt 0 NIL NIL NIL)
     )
     (pif (> (list-length liuy1) maxw)
     (progn
     (csetq maxitem dpt)
     (csetq maxw (list-length liuy1))(print 'greater)
     )    ) )
   (csetq spechildren
      (ASK-TEMPLATE
      (LIST
      (QUOTE ?f1))
      (LIST #$nearestGenls
      (QUOTE ?f1)
      maxitem) #$EverythingPSC 0 NIL NIL NIL)
    )
   (csetq addquantifier (list maxitem))
```

```
 (pif (> (list-length spechildren)  maxw)
    (cpush #$thereExists  addquantifier)
 )
 (csetq  x
    (ASK-TEMPLATE
    (LIST
    (QUOTE ?f1)
    (QUOTE ?freqc))
    (LIST #$xkid
    (QUOTE ?f1)
    (QUOTE ?freqc)) #$TextSummarizationMt 0 NIL NIL NIL)
    )
 (csome  (xkidd1  x)
   (WITH-BOOKKEEPING-INFO
   (NEW-BOOKKEEPING-INFO NIL
   (THE-DATE) NIL
   (THE-SECOND))
   (CLET
   (
    (*REQUIRE-CASE-INSENSITIVE-NAME-UNIQUENESS* NIL)
    (*THE-CYCLIST* NIL)
    (*KE-PURPOSE* NIL))
    (KE-UNASSERT-NOW
    (LIST #$xkid (car xkidd1) (second xkidd1) ) #$TextSummarizationMt)))
    )
 (ret addquantifier)
)
```

B-16 generate new sentences from cluster list

```
(define assemble1()
(csetq SPOs (open  "2 SPO.txt" :direction :output))
(csetq Commatrix (open  "3 Commatrix.txt" :direction :output))
(csetq Comclusters (open  "4 Comclusters.txt" :direction :output))
(csetq genes (open  "6 generalized.txt" :direction :output))
(csetq sents (open  "7 Gsentences.txt" :direction :output))
(csetq clustersentences (open  "5 ComclustersWithSentences.txt" :direction :output))
(csetq spo
   (ASK-TEMPLATE
   (LIST
   (QUOTE ?f1)
   (QUOTE ?f2)
   (QUOTE ?f3)
   (QUOTE ?f4)
    )
   (LIST #$zwb
   (QUOTE ?f1)
   (QUOTE ?f2)
   (QUOTE ?f3)
   (QUOTE ?f4)
   ) #$TextSummarizationMt 0 NIL NIL NIL)
)
 (csetq spolen (list-length spo))
 (csetq ss 0)
 (csetq spc " ")
 (csome (sbi spo)
 (format SPOs "~%")
```

```
(format SPOs "~4D " ss )
(prin1 sbi SPOs)
(prin1 (first sbi)) (prin1 '__) (prin1 (second sbi)) (prin1 '__)
(prin1 (third sbi)) (prin1 '__) (prin1 (fourth sbi))
(print ss)  (cinc ss)
)
(close SPOs)
(csetq compmatrix  (MAKE-VECTOR spolen) )
(cdotimes (nlens  spolen)
        (set-aref compmatrix  nlens (MAKE-VECTOR spolen)  )
)
(cdotimes (nleny  spolen)
  (print '__)
  (cdotimes (nlenx  spolen)
    (pif (> nlenx nleny)
       (progn
       (csetq ctemp (compatible (nth nlenx spo ) (nth nleny spo )))
       (set-aref (aref compmatrix  nleny) nlenx ctemp)
      )
      (pif (< nlenx nleny)
       (set-aref (aref compmatrix  nleny) nlenx 0)
       (set-aref (aref compmatrix  nleny) nlenx 1)
      )     )    ) )
(csetq spc " ")
(format Commatrix "~4A" spc )
(cdotimes (nlenz  spolen)
   (format Commatrix "~4D" nlenz )
)
(format Commatrix "~%"  )
(cdotimes (nleny  spolen)
  (format Commatrix "~4D" nleny )
  (cdotimes (nlenx  spolen)
        (format Commatrix "~4D" (aref (aref compmatrix  nleny)  nlenx) )
  )
  (format Commatrix "~%"  )
)
(close Commatrix)
(csetq unclustered nil)
(csetq clusterqueue nil)
(csetq candchainlen (- spolen 1))
(cdotimes (nleny  candchainlen)(print '_)
        (csetq newcluster nil)
        (csetq deltalen (- spolen nleny))     ;(cdec deltalen)
        (cdotimes (dnum deltalen)
          (csetq colm (+ dnum nleny))
       (pif (> (aref (aref compmatrix  nleny)  colm ) 0 )
                (csetq newcluster (append  newcluster (list colm)))
       )
        )
        (pif (equalp (list-length newcluster) 2)
          (pif (equalp (list-length clusterqueue) 0)
            (csetq clusterqueue (list newcluster))
            (csetq clusterqueue (append  clusterqueue (list newcluster)))
          )

        (pif (> (list-length newcluster) 2)
```

```
          (progn
                    (pif (equalp (list-length unclustered) 0)
                    (csetq unclustered (list newcluster))
                    (csetq unclustered (append  unclustered (list newcluster)))
                    )
          )
         )
          )
)
(pif (> (list-length unclustered) 0)
   (csome ( unc  unclustered )
           (print 'unc) (print unc)
      (csetq clusterqueue (validate unc compmatrix clusterqueue))
   )
   )
 (csetq clusterqueue  (filter clusterqueue))
(clusterspo spo  clusterqueue)
(close clustersentences)
(close  Comclusters)
(close genes)
(close sents)
 )
```

B-17 Build compatible clusters from compatible matrix

```
 (define validate (unlist compmatrix clusterqueue)
   (csetq unvalid nil)
   (csetq unvalid (append unvalid (list unlist)))
   (cdo
      (
          (cmptime 0 (1+ cmptime))
          (ckrepeat nil (>= cmptime (list-length unvalid) ) )
      )
     ((eval ckrepeat))
     (progn
       (csetq unvaliditem (nth cmptime unvalid))
       (csetq chainlen (list-length unvaliditem
       (csetq fy (- chainlen 1))
       (csetq fx -1)
       (csetq breakx 0)
       (cdo
          ((iy 0 (1+ iy)))
          ((cor (equalp breakx 1) (>= iy (- chainlen 1))))
            (cdo
            ((ix (+ iy 1) (1+ ix)))
            ((cor (equalp breakx 1) (>= ix chainlen)))
            (progn
              (pif (equalp (aref (aref compmatrix (nth iy  unvaliditem))  (nth ix unvaliditem))  0)
                    (progn
                          (csetq breakx 1)
                          (csetq fy iy)
                          (csetq fx ix)
          )  )       )      )       )
            (pif (equalp breakx 0)
          (csetq clusterqueue (append clusterqueue (list unvaliditem)))
          (progn
            (csetq processeditem nil)
```

```
          (pif (> fy 0)
              (cdotimes (iiy fy)
                (csetq processeditem (append processeditem (list (nth iiy unvaliditem))))
               )
              )
          (csetq miditem1 nil)
          (csetq miditem2 nil)
          (cdotimes (iv (- (list-length unvaliditem) fy ))
                 (pif (equalp iv 0)
                (csetq miditem1  (LIST (nth fy unvaliditem)))
                (pif (equalp (+ fy iv) fx)
                   (csetq miditem2 (append miditem2 (list (nth   (+ fy iv) unvaliditem))))
                       (progn
                          (csetq miditem1 (append miditem1 (list (nth (+ fy iv) unvaliditem))))
                          (csetq miditem2 (append miditem2 (list (nth (+ fy iv) unvaliditem))))
        )                    )                  )                )
              (csetq finitem1 (append processeditem miditem1 ))
              (csetq finitem2 (append processeditem miditem2 ))
              (csetq unvalid (append unvalid (list finitem1 )))
              (csetq unvalid (append unvalid (list finitem2 )))
           )       )            )      )
(ret clusterqueue)
)
```

B-18 Filter out the small cluster which is covered by the big cluster in the cluster list

```
(define filter (queuef)
(csetq queuelen (list-length queuef))
(csetq queuelenmatrix  (MAKE-VECTOR   queuelen))
(csetq queuematrix  (MAKE-VECTOR   queuelen))
(csetq cnter 0)
(csome (queueitem queuef)
    (set-aref queuelenmatrix cnter (list-length queueitem))
    (cinc cnter)
)
(csetq leftlen queuelen)
(csetq newlen 0)
(cdo
         (
          (cmptime 0 (1+ cmptime))
          (ckrepeat nil (cor (>= cmptime queuelen ) (>= newlen queuelen) )       )
         )
    ( (eval ckrepeat))
    ( progn
      (csetq buff nil)
      (csetq maxq 0)
      (cdotimes (ctm queuelen)
        (pif (>  (aref queuelenmatrix ctm) maxq )
          (progn
             (csetq maxq (aref queuelenmatrix ctm))
             (csetq buff nil)
             (cpush ctm buff)
          )
          (pif (equalp (aref queuelenmatrix ctm) maxq )
            (cpush ctm buff)
          )         )                )
      (cdotimes (cds (list-length buff))
```

```
          (set-aref queuematrix (- queuelen 1 cds newlen) (nth (nth cds buff) queuef))
          (set-aref queuelenmatrix  (nth cds buff) 0)
        )
        (csetq newlen (+ newlen (list-length buff)))
      )
)
(cdotimes  (iqg queuelen)
(pif (equalp  (aref queuematrix iqg) nil)
(iqg)
)
)
 (csetq qlen_1 (- queuelen 1))
     (cdotimes ( ql qlen_1)
        (csetq tmp (aref queuematrix  ql))
        (cdotimes (ql1 (- qlen_1  ql ))
           (csetq tmp1 (aref queuematrix (+ ql  ql1 1)))
               (csetq qqf (containq tmp tmp1))
               (pif (= qqf 1)
              (set-aref queuematrix ql nil)
               )      )     )
(csetq xqueue nil)
(cdotimes  (iqg queuelen)
   (pif (equalp  (aref queuematrix iqg) nil)
      (csetq fp 0)
 (cpush (aref queuematrix iqg)  xqueue)
)
)
(ret xqueue)
)
```

B-19 Check if cluster vA is covered by cluster vB, or cluster vV is covered by cluster vA

```
(define containq(vA vB)  (print vA)(print vB)
(csetq qs 0)
(cdolist (la vA)
  (csetq qf 1)
  (pif (equalp qf 1)
     (progn
       (csetq abf 0)
       (cdolist (lb vB)
         (pif (equalp la lb)
                (csetq abf 1)
          )
       )
       (pif (equalp abf 0)
          (csetq qf 0)
       )
     )
  )
  (csetq qs (+ qs qf))
)
(pif (equalp qs (list-length vA))
    (csetq finq 1)
    (csetq finq 0)
)
(ret finq)
)
```

B-20 Check if sentence compx is compatible to compy

```
(define compatible(compx compy)
(csetq subx (first compx))
(csetq suby (first compy))
(csetq predx (second compx))
(csetq predy (second compy))
(csetq objx (third compx))
(csetq objy (third compy))
(csetq sf (compatpart subx suby))(print subx)(prin1 suby)(prin1 sf)
(csetq pf (compatpart predx predy))(print predx)(prin1 predy)(prin1 pf)
(csetq of (compatpart objx objy))(print objx)(prin1 objy)(prin1 of)
(csetq chkf (+ sf pf of))
(pif (EQUALP chkf 3)
   (ret 1)       (ret 0)
)
)
```

B-21 Check if concept partx and party compatible

```
(define compatpart (partx party)
(csetq comflag 0)
(pif (equalp partx party)
    (progn
    (csetq comflag 1)
    (ret comflag)
    )
)
(csetq parentgenx
  (REMOVE-DUPLICATES
  (WITH-ALL-MTS
  (MIN-GENLS  partx  )))
  )
(csetq parentgeny
  (REMOVE-DUPLICATES
  (WITH-ALL-MTS
  (MIN-GENLS  party)))
  )
(csome (gx parentgenx )
    (csome (gy parentgeny)
      (pif (equalp gx gy)
        (progn
          (csetq comflag 1)
          (ret comflag)
        )
      )
    )
 )
 (pif (CYC-QUERY (quote (#$nearestGenls partx party)) #$EverythingPSC)
    (csetq comflag 1)
    (pif (CYC-QUERY (quote (#$nearestGenls party partx)) #$EverythingPSC)
       (progn
          (csetq comflag 1)
          (ret comflag)
    )
 )
)
```

```
(pif (CYC-QUERY (quote (#$nearestIsa partx party)) #$EverythingPSC)
(csetq comflag 1)
(pif (CYC-QUERY (quote (#$nearestIsa party partx)) #$EverythingPSC)
    (progn
    (csetq comflag 1)
    (ret comflag)
    )
)
)
(pif (CYC-QUERY (quote (#$conceptuallyRelated partx party)) #$EverythingPSC)
(csetq comflag 1)
(pif (CYC-QUERY (quote (#$conceptuallyRelated party partx)) #$EverythingPSC)
    (progn
    (csetq comflag 1)
    (ret comflag)
    )
)
)
(ret comflag)
)
```

B-22 Programs to call stanford parser to parse file and map words & phrases to Cyc concepts to

1. **generate top 10 key concept**
2. **generate new key sentences**
3. **extract top 10 key sentences**

```
package edu.stanford.nlp.parser.lexparser;
import edu.stanford.nlp.trees.AbstractTreebankLanguagePack;
import edu.stanford.nlp.fsm.ExactGrammarCompactor;
import edu.stanford.nlp.io.NumberRangeFileFilter;
import edu.stanford.nlp.io.NumberRangesFileFilter;
import edu.stanford.nlp.ling.HasTag;
import edu.stanford.nlp.ling.HasWord;
import edu.stanford.nlp.ling.Sentence;
import edu.stanford.nlp.objectbank.TokenizerFactory;
import edu.stanford.nlp.parser.ViterbiParser;
import edu.stanford.nlp.process.DocumentPreprocessor;
import edu.stanford.nlp.process.Function;
import edu.stanford.nlp.process.WhitespaceTokenizer;
import edu.stanford.nlp.trees.*;
import edu.stanford.nlp.util.Pair;
import edu.stanford.nlp.util.*;
import java.io.*;
import java.net.URL;
import java.net.URLConnection;
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.*;
import java.util.zip.GZIPInputStream;
import java.util.zip.GZIPOutputStream;
import org.opencyc.cycobject.*;
import org.opencyc.util.*;
import org.opencyc.api.*;

class SubPredObjgroup
  {
  ArrayList<String[]> mtrx  ;
```

```
  ArrayList<Integer[]> itrx ;
  ArrayList<zhuweibin> zwb;

  SubPredObjgroup(ArrayList<String[]> mtrx0, ArrayList<Integer[]>itrx0,  ArrayList<zhuweibin> zwb0 )
  {
  mtrx = mtrx0;
  itrx = itrx0;
  zwb  = zwb0;
  }
  ArrayList<String[]> getmtrx()
  {  return mtrx;
  }
  ArrayList<Integer[]> getitrx()
  {  return itrx;
  }
  ArrayList<zhuweibin> getzwb()
  {  return zwb;
  }
  }

class nppair
{     int length;
            double weight;
            CycFort[] concepts;
            nppair(ArrayList<CycFort> outfort, int ln)
            {     length= ln;
                        concepts=new CycFort[outfort.size()];
                        for(int i=0;i<outfort.size();i++)
                        concepts[i]=outfort.get(i);
            }
            public int getlength()
            {
            return length;
            }
            public void setweight(CycAccess cycAccess)
            {
            try{
            String temp=concepts[0].cyclify();
                        CycList kua=null;
                        if(temp.startsWith("#$"))
                        kua=cycAccess.converseList("(gou "+temp+")");
                        if(temp.startsWith("("))
                        kua=cycAccess.converseList("(gou (quote "+temp +"))");
                        String biang;
                        String biang1;
                        double weight0=0.0;
                        if(kua!=null)
                        {
                          biang=kua.first().toString();
                          biang1=biang.substring(1,biang.length()-1);
                            weight0=new Double(biang1);
                        }
              if(concepts.length==1)  weight=weight0;
              else
                if(concepts.length>1)
                {
```

```
                    int loc=0;
                    String biangi;
                    String biang1i;
                    double weight0i;
                    weight0i=weight0;
                    for(int i=1; i<concepts.length; i++)
                    {
                    String tempi=concepts[i].cyclify();
                            CycList kuai=null;
                            if(tempi.startsWith("#$"))
                            kuai=cycAccess.converseList("(gou "+tempi+")");
                            if(tempi.startsWith("("))
                            kuai=cycAccess.converseList("(gou (quote "+tempi +"))");
                            if(kuai!=null)
                            {
                              biangi=kuai.first().toString();
                              biang1i=biangi.substring(1,biangi.length()-1);
                              weight0i=new Double(biang1i);
                            }
                            if(weight0i>weight0)  loc=i;
                    }
                    weight=weight0i;
                    if(loc!=0)
                    {
                            CycFort ftemp=concepts[loc];
                            concepts[loc]=concepts[0];
                            concepts[0]=concepts[loc];
                    }
                    }

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

            public int size()
            {return concepts.length;}

            public CycFort nppairitem(int i)
            {return concepts[i];}

            public double getweight()
            {return weight;}
}

class nppairlist
{
            int length;
            ArrayList<nppair> nplist;
            nppairlist()
            {
                    nplist=new ArrayList<nppair>();
                    length=0;
            }

            public void add(nppair outpr)
            {
            nplist.add(outpr);
```

```
            length++;
            }

            public int size()
            {return length; }

            public nppair getnppair(int i)
            {return nplist.get(i);}
}

class cizuji
{
int start;
int len;
ArrayList<CycFort> cizu;
cizuji(int st, int ln, ArrayList<CycFort> czu)
{
start=st;
len=ln;
cizu=czu;
}

public int getstart()
{
return start;
}
public int getlength()
{
return len;
}

public ArrayList<CycFort>  getcizu()
{
return cizu;
}
}

 class zhuweibin
{
int zhu, wei, bin;

public zhuweibin(int zzhu, int wwei, int bbin)
{
zhu=zzhu;
wei=wwei;
bin=bbin;

}
public int getzhu()
{
return zhu;
}

public int getwei()
{
return wei;
```

```
}

public int getbin()
{
return bin;
}
}

class constituentpair
{
int  gov, dep;
public constituentpair(int gg, int dd)
{
gov=gg;
dep=dd;
}
public int getgov()
{
return gov;
}

public int getdep()
{
return dep;
}
}

class tui
{
CycFort[]  qun;
CycFort[] keng;
Double[] kengq;
CycFort[] tou;
CycFort[][] wei;
boolean[] cheng;
Double[] weiq;
CycAccess cycAccess;
ArrayList<ArrayList<CycFort>> zuzu=new ArrayList<ArrayList<CycFort>>();
public tui( ArrayList<CycFort> keng0, ArrayList<Double> kengq0, ArrayList<CycFort> tou0,
ArrayList<ArrayList<CycFort>> wei0,CycAccess cycA)
{
cycAccess=cycA;
keng=new CycFort[keng0.size()];
for(int i=0; i<keng0.size();i++)
keng[i]=keng0.get(i);

kengq=new Double[kengq0.size()];
for(int i=0; i<kengq0.size();i++)
kengq[i]=kengq0.get(i);
tou=new CycFort[tou0.size()];
for(int i=0; i<tou0.size();i++)
tou[i]=(CycFort)(tou0.get(i));
weiq=new Double[tou.length];
wei=new CycFort[wei0.size()][];
for(int i=0; i<wei0.size();i++)
{
```

```
ArrayList<CycFort> xu=wei0.get(i);
wei[i]=new CycFort[xu.size()];
for(int j=0;j<xu.size();j++)
wei[i][j]=xu.get(j);
}
cheng=new boolean[tou.length];
for(int i=0;i<tou0.size();i++)
cheng[i]=false;
for(int i=0;i<tou0.size();i++)
{
ArrayList<CycFort> huahua=new ArrayList<CycFort>();
zuzu.add(huahua);
}
}
public ArrayList<CycFort> appd(CycFort jiao)
{
ArrayList<CycFort> rong=new ArrayList<CycFort>();
for(int i=0; i<tou.length;i++)
{
CycFort temp=tou[i];
if(jiao.equals(temp))
{
if(cheng[i]) return zuzu get(i);
for(int j=0;j<wei[i].length;j++)
{
   boolean shi=true;
   for(int k=0;k<keng.length;k++)
   if(wei[i][j].equals(keng[k])){ rong.add(wei[i][j]); shi=false; break;}
   if(shi)
   {
   ArrayList<CycFort> xx=appd(wei[i][j]);
   for(int m=0;m<xx.size();m++)
   rong.add(xx.get(m));
   }
   int xia=rong.size();
}
cheng[i]=true;
zuzu.set(i,rong);
break;
}
}
return rong;
}

public CycFort proc()
{
for(int i=0;i<tou.length;i++)
{
CycFort temp=tou[i];
appd(temp);
}
for(int sq=0;sq<tou.length;sq++)
{
CycFort cf=(CycFort)(tou[sq])     ;
ArrayList<CycFort> zx=zuzu.get(sq);
int ho=      zx.size();
```

```
}
for(int i=0;i<zuzu.size();i++)
{
ArrayList<CycFort> kaihua=zuzu.get(i);
for(int j=0;j<kaihua.size()-1;j++)
 for(int k=j+1;k<kaihua.size();k++)
 if(kaihua.get(j).equals(kaihua.get(k)))
 kaihua.remove(k);
 double sum=0;
 try{
  CycList kua=cycAccess.converseList("(gou "+tou[i].cyclify()+")");
  if(kua!=null)
  {
   String biang=kua.first().toString();
   String biang1=biang.substring(1,biang.length()-1);
  sum=new Double(biang1);
  }
 for(int m=0;m<kaihua.size();m++)
 {
   CycList kua1=cycAccess.converseList("(gou "+kaihua.get(m).cyclify()+")");
   String biang=kua.first().toString();
   String biang1=biang.substring(1,biang.length()-1);
   Double bia=new Double(biang1);
   sum+=bia;
 }
 }catch (Exception e) {e.printStackTrace();}
weiq[i]=sum;
}
int to=0;
int ba=-1;
double zui=0.0;
for(int i=0;i<kengq.length;i++)
{
if(kengq[i]>zui)
{
to=i;
zui=kengq[i];
}
}
for(int i=0;i<weiq.length;i++)
{
if(weiq[i]>zui)
{
ba=i;
zui=weiq[i];
}
}
CycFort zda;
if(ba>=0) zda=tou[ba];
else zda=keng[to];
return zda;
}
}

 class pospair
{
```

```
String word, pos;
public pospair(String wword, String ppos)
{
word=new String(wword);
pos=new String(ppos);
}
public String getword()
{
return word;
}

public String getpos()
{
return pos;
}
}

 class weightpair
{
double senweight;
int senorder;
public weightpair(double weight, int order)
{
senweight=weight;
senorder=order;
}
public double getweight()
{
return senweight;
}

public int getorder()
{
return senorder;
}
}

class sumtree
{
ArrayList<CycFort> prt;
ArrayList<CycFort> siblings;
ArrayList<CycFort> kid;
public sumtree(ArrayList<CycFort> pt,ArrayList<CycFort> sib,ArrayList<CycFort> kd)
{
prt=pt;
siblings=sib;
kid=kd;
}

public ArrayList<CycFort> getparent()
{
return prt;
}
public ArrayList<CycFort> getsiblings()
{
return siblings;
```

```
}
public ArrayList<CycFort> getchildren()
{
return kid;
}
}

SubPredObjgroup extractSubPredObj(String[][] WordArray,int sentenceorder,int num,PrintWriter pw,
PrintWriter pw3, Tree ansTree,TreebankLanguagePack tlp,List sentence)
{
   boolean vx=false;
   boolean sx=false;
   boolean ox=false;
   String sv=new String();
   String ss=new String();
   String so=new String();
   int locpos=-1;
  ArrayList<String[]> mtrx=new ArrayList<String[]>();
  ArrayList<Integer[]> itrx=new ArrayList<Integer[]>();
  Filter<Dependency> dependencyFilter;
  GrammaticalStructureFactory gsf;
 Filter<String> puncWordFilter = tlp.punctuationWordRejectFilter();
 gsf = tlp.grammaticalStructureFactory(puncWordFilter);
 GrammaticalStructure gs = gsf.newGrammaticalStructure(ansTree);
 Collection<TypedDependency> sss=gs.typedDependenciesCollapsed();=
 ArrayList<constituentpair> subj=new ArrayList<constituentpair>();
 ArrayList<constituentpair> psubj=new ArrayList<constituentpair>();
 ArrayList<constituentpair> obj=new ArrayList<constituentpair>();
 ArrayList<constituentpair> agent=new ArrayList<constituentpair>();
 ArrayList<constituentpair> cop=new ArrayList<constituentpair>();
 pw3.println(sentence.toString());
 for (TypedDependency td : sss) {
             String gv=td.gov().value();
             String rln= td.reln().toString();
             String dp=td.dep().value();
             int govIdx = td.gov().index();
             int dpIdx = td.dep().index();
             constituentpair xin=new constituentpair(govIdx-1,dpIdx-1);
             pw3.println(td.toString());
             if(rln.indexOf("agent")>=0)         agent.add(xin);
             if(rln.indexOf("dobj")>=0)          obj.add(xin);
             if(rln.indexOf("nsubjpass")>=0)    psubj.add(xin);
             if(rln.indexOf("cop")>=0)          cop.add(xin);
             if((rln.indexOf("nsubj")>=0)&&(rln.indexOf("nsubjpass")<0)) subj.add(xin);
             if((rln.indexOf("subj")>=0)||(rln.indexOf("agent")>=0))
             {
                 Integer locps=td.gov().index();
                 Integer locpss=td.dep().index();
                 locpos=locps.intValue();
                 vx=true;
                 sx=true;
                 sv=gv;
                 ss=dp;
                 int inspct=-1;
                 for(int ivt=0;ivt<itrx.size();ivt++)
                 {
```

```
 if(locpos==itrx.get(ivt)[1]){inspct=ivt;break;}
}
if(inspct==-1)
{
String[] spo=new String[3];
Integer[] spo1=new Integer[3];
if(rln.indexOf("nsubjpass")>=0)
{
spo[2]=ss; spo1[2]=locpss;
spo[1]=sv; spo1[1]=locps;
}
else
{
spo[0]=ss; spo1[0]=locpss;
spo[1]=sv; spo1[1]=locps;
}
 mtrx.add(spo);
        itrx.add(spo1);
        }
else
{
 if(rln.indexOf("nsubjpass")>=0)
{
String[] xs=mtrx.get(inspct);
xs[2]=ss;
mtrx.set(inspct,xs);
Integer[] xsx=itrx.get(inspct);
xsx[2]=locpss;
itrx.set(inspct,xsx);
  }
else{
String[] xs=mtrx.get(inspct);
xs[0]=ss;
mtrx.set(inspct,xs);
Integer[] xsx=itrx.get(inspct);
xsx[0]=locpss;
itrx.set(inspct,xsx);
}
}
}
if(rln.indexOf("obj")>=0)
{
vx=true;
ox=true;
Integer locps1=govIdx;
int locpos1=locps1.intValue();
Integer locpss1=dpIdx;
sv=gv;
so=dp;
int inspct=-1;
for(int ivt=0;ivt<itrx.size();ivt++)
 if(locpos1==itrx.get(ivt)[1]){inspct=ivt;break;}
if(inspct==-1)
{
if(rln.indexOf("iobj")>=0)
{
```

```
Integer[] spo1=new Integer[4];
String[] spo=new String[4];
spo[3]=so;spo1[3]=locpss1;
spo[1]=sv;spo1[1]=locps1;
itrx.add(spo1);
mtrx.add(spo);
}
if(rln.indexOf("dobj")>=0)
{
Integer[] spo1=new Integer[3];
String[] spo=new String[3];
spo[2]=so;spo1[2]=locpss1;
spo[1]=sv;spo1[1]=locps1;
itrx.add(spo1);
mtrx.add(spo);
}
        }
else
{
String[] xs=mtrx.get(inspct);
if(rln.indexOf("dobj")>=0)
{
xs[2]=so;
mtrx.set(inspct,xs);
Integer[] xsx=itrx.get(inspct);
xsx[2]=locpss1;
itrx.set(inspct,xsx);
}
if(rln.indexOf("iobj")>=0)
{
String[] xs1=new String[4] ;
xs1[0]=xs[0];
xs1[1]=xs[1];
xs1[2]=xs[2];
xs1[3]=so;
mtrx.set(inspct,xs1);
Integer[] xsx=itrx get(inspct);
Integer[] xsx1=new Integer[4] ;
xsx1[0]=xsx[0];
xsx1[1]=xsx[1];
xsx1[2]=xsx[2];
xsx1[3]=locpss1;
itrx.set(inspct,xsx1);
}
}
}
}
pw.println(sentence.toString());
if(agent!=null)
 if(agent.size()>0)
        for(int ag=0;ag<agent.size();ag++)
        {
        constituentpair bao=agent.get(ag);
        int gao=bao.getgov();
        int gao1=bao.getdep();
        pw.println(gao+"  "+gao1);
```

```
            pw.println(WordArray[sentenceorder][gao]+"  "+WordArray[sentenceorder][gao1]);
            }
 if(psubj!=null)
  if(psubj.size()>0)
            for(int ps=0;ps<psubj.size();ps++)
            {
            constituentpair bao=psubj.get(ps);
            int gao=bao.getgov();
            int gao1=bao.getdep();
            pw.println(gao+"  "+gao1);
            pw.println(WordArray[sentenceorder][gao]+"  "+WordArray[sentenceorder][gao1]);
           }
 if(obj!=null)
  if(obj.size()>0)
            for(int ob=0;ob<obj.size();ob++)
            {
            constituentpair bao=obj.get(ob);
            int gao=bao.getgov();
            int gao1=bao.getdep();
            pw.println(gao+"  "+gao1);
            pw.println(WordArray[sentenceorder][gao]+"  "+WordArray[sentenceorder][gao1]);
            }
 if(subj!=null)
  if(subj.size()>0)
            for(int sj=0;sj<subj.size();sj++)
            {
            constituentpair bao=subj.get(sj);
            int gao=bao.getgov();
            int gao1=bao.getdep();
            pw.println(gao+"  "+gao1);
            pw.println(WordArray[sentenceorder][gao]+"  "+WordArray[sentenceorder][gao1]);
            }
 if(cop!=null)
  if(cop.size()>0)
            for(int cp=0;cp<cop.size();cp++)
            {
            constituentpair bao=cop.get(cp);
            int gao=bao.getgov();
            int gao1=bao.getdep();
            pw.println(gao+"  "+gao1);
            pw.println(WordArray[sentenceorder][gao]+"  "+WordArray[sentenceorder][gao1]);
            }

ArrayList<zhuweibin> tmps=new  ArrayList<zhuweibin>();
if(subj.size()>0)
{
for(int ii=0; ii<subj.size(); ii++)
{
int gg=subj.get(ii).getgov();
int dd=subj.get(ii).getdep();
boolean flg=false;
if(obj.size()>0)
for(int jj=0;jj<obj.size();jj++)
{
constituentpair ctmp=obj.get(jj);
int ipp=ctmp.getgov();
```

```
                if(gg==ipp)
                {
                zhuweibin zhwbb=new zhuweibin(dd,gg,ctmp.getdep());
                tmps.add(zhwbb);
                flg=true;
                break;
                }
                }
                if(!flg)
                {
                if(cop.size()>0)
                for(int jj=0;jj<cop.size();jj++)
                {
                constituentpair ctmp=cop.get(jj);
                int ipp=ctmp.getgov();
                if(gg==ipp)
                {
                zhuweibin zhwbb=new zhuweibin(dd, ctmp.getdep(),gg );
                tmps.add(zhwbb);
                flg=true;
                break;
                }
                }
                }
                }
                }
                if(psubj.size()>0)
                {
                for(int ii=0; ii<psubj.size(); ii++)
                {
                int gg=psubj.get(ii).getgov();
                int dd=psubj.get(ii).getdep();
                if(agent.size()>0)
                for(int jj=0;jj<agent.size();jj++)
                {
                constituentpair ctmp=agent.get(jj);
                int ipp=ctmp.getgov();
                if(gg==ipp)
                {
                zhuweibin zhwbb=new zhuweibin(ctmp.getdep(),gg, dd );

                tmps.add(zhwbb);
                break;
                }
                }

                }
                }
                 SubPredObjgroup sporesult=new SubPredObjgroup(mtrx,itrx,tmps);
                 return  sporesult;
}


 public int jiansuo(Tree diyi)
 {
            int crt=-1;
```

```
            if(diyi.isLeaf())
            {
            crt=gongli++;
            qishi.add(crt);
            wei.add(1);
            kuk.add(diyi);
            }
            else
            {
            Tree[] yu=diyi.children() ;
            crt=jiansuo(yu[0]);
            if(yu.length>1)
            {
            for(int j=1;j<yu.length;j++)
            jiansuo(yu[j]);
            }
            kuk.add(diyi);
            qishi.add(crt);
            wei.add(yu.length);
            }
    return crt;
 }

public static ArrayList<String>  advprocess(CycAccess cycAccess, CycFort mte1,String wordtable)
{
char c[]=new char[1];c[0]='\"';String t=new String(c);
ArrayList alst=new ArrayList();
CycVariable cv1=CycObjectFactory.makeCycVariable("?WORD");
CycVariable cv2=CycObjectFactory.makeCycVariable("?POS");
CycVariable cv3=CycObjectFactory.makeCycVariable("?NUM");
CycVariable cv4=CycObjectFactory.makeCycVariable("?TERM");
alst.add(cv1);
alst.add(cv2);
alst.add(cv3);
alst.add(cv4);
String sssf=new String("(#$and (#$denotation ?WORD ?POS ?NUM ?TERM) (#$wordForms ?WORD
#$adverbStrings ");
String nounword=new String(wordtable.toLowerCase());
String tttf=new String(t+nounword+t);
String vvvf=new String (") (#$genls ?POS #$Adverb))");
CycList cl1=cycAccess.makeCycList(sssf+tttf+vvvf);
ArrayList<String> tp= new ArrayList<String>();
boolean connart=false;
try{
CycList res=cycAccess.askWithVariables(cl1,alst,mte1);
CycListVisitor clv0=res.cycListVisitor();
if(!clv0.hasMoreElements())
{
CycList xinfa=cycAccess.getDenotsOfString(wordtable.toLowerCase());
if(!xinfa.isEmpty())
{
for(int ti=0;ti<xinfa.size();ti++)
{
 if (xinfa.get(ti) instanceof CycConstant || xinfa.get(ti) instanceof CycNart)
 {
   tp.add(DefaultCycObject.cyclify(xinfa.get(ti)).toString());
```

```
    connart=true;
    }
}
}
}
 int yx=0;
while(clv0.hasMoreElements())
{
if(yx%4<3)clv0.nextElement();
else
 {
  Object advtemp=clv0.nextElement();
  if (advtemp instanceof CycConstant || advtemp instanceof CycNart)
  {
  tp.add(DefaultCycObject.cyclify(advtemp).toString());
  connart=true;
  }
 }
yx++;
}
}catch(Exception e){System.out.println("can not find  ");}
ArrayList<String> nncycs=new ArrayList<String>();
  if(connart)
  {
  int shu=0;
  String[] tp1=new String[tp.size()];
  tp1[0]=tp.get(0);
  shu=1;
  for(int ii=1;ii<tp.size();ii++)
  {
  String tmp=tp.get(ii);
  Boolean bmp=false;
  for(int jj=0;jj<shu;jj++)
  {
  if(tmp.equals(tp1[jj]))
  {
  bmp=true;   break;
  }
  }
  if(!bmp){  tp1[shu]=tmp;shu++;  System.out.println("IN not equal  "+tmp);}
  }
  for(int ii=0;ii<shu;ii++)
  {
  nncycs.add(tp1[ii]);
  }
  return nncycs;
  }
  return null;
 }

  public static ArrayList<String> adjprocess(CycAccess cycAccess, CycFort mte1,String wordtable)
{
char c[]=new char[1]; c[0]='\"';String t=new String(c);
ArrayList alst=new ArrayList();
CycVariable cv1=CycObjectFactory.makeCycVariable("?WORD");
CycVariable cv2=CycObjectFactory.makeCycVariable("?POS");
```

```
CycVariable cv3=CycObjectFactory.makeCycVariable("?NUM");
CycVariable cv4=CycObjectFactory.makeCycVariable("?TERM");
alst.add(cv1);
alst.add(cv2);
alst.add(cv3);
alst.add(cv4);
String sssf=new String("(#$and (#$denotation ?WORD ?POS ?NUM ?TERM) (#$wordForms ?WORD
#$adjStrings ");
String nounword=new String(wordtable.toLowerCase());
String tttf=new String(t+nounword+t);
String vvvf=new String (") (#$genls ?POS #$Adjective))");
CycList cl1=cycAccess.makeCycList(sssf+tttf+vvvf);
ArrayList<String> tp= new ArrayList<String>();
boolean connart=false;
try{
CycList res=cycAccess.askWithVariables(cl1,alst,mte1);
CycListVisitor clv0=res.cycListVisitor();
if(!clv0.hasMoreElements())
{
CycList xinfa=cycAccess.getDenotsOfString(wordtable.toLowerCase());
if(!xinfa.isEmpty())
{
for(int ti=0;ti<xinfa.size();ti++)
if (xinfa.get(ti) instanceof CycConstant || xinfa.get(ti) instanceof CycNart)
{
tp.add(DefaultCycObject.cyclify(xinfa.get(ti)).toString());
connart=true;
}
}
}
 int yx=0;
while(clv0.hasMoreElements())
{
if(yx%4<3)clv0.nextElement();
else
 {
  Object adjtemp=clv0.nextElement();
  if (adjtemp instanceof CycConstant || adjtemp instanceof CycNart)
  {
  tp.add(DefaultCycObject.cyclify(adjtemp).toString());
  connart=true;
  }
 }
yx++;
}
}catch(Exception e){System.out.println("can not find  ");}
ArrayList<String> nncycs=new ArrayList<String>();
   if(connart)
   {
   int shu=0;
   String[] tp1=new String[tp.size()];
   tp1[0]=tp.get(0);
   shu=1;
   for(int ii=1;ii<tp.size();ii++)
   {
   String tmp=tp.get(ii);
```

```
  Boolean bmp=false;
  for(int jj=0;jj<shu;jj++)
  {
  if(tmp.equals(tp1[jj]))
  {
  bmp=true;  break;
  }
  }
  if(!bmp){  tp1[shu]=tmp;shu++;  System.out.println("IN not equal  "+tmp);}
  }
  for(int ii=0;ii<shu;ii++)
  nncycs.add(tp1[ii]);
  return nncycs;
  }
  return null;
 // else nncyc.add(null);
}

public static ArrayList<String> verbprocess(CycAccess cycAccess, CycFort mte1,String wordtable)
{
char c[]=new char[1];
c[0]="\"";
String t=new String(c);
ArrayList alst=new ArrayList();
CycVariable cv1=CycObjectFactory.makeCycVariable("?WORD");
CycVariable cv2=CycObjectFactory.makeCycVariable("?POS");
CycVariable cv3=CycObjectFactory.makeCycVariable("?NUM");
CycVariable cv4=CycObjectFactory.makeCycVariable("?TERM");
alst.add(cv1);
alst.add(cv2);
alst.add(cv3);
alst.add(cv4);
String sssf=new String("(#$and (#$denotation ?WORD ?POS ?NUM ?TERM) (#$wordForms ?WORD
#$verbStrings ");
String nounword=new String(wordtable.toLowerCase());
String tttf=new String(t+nounword+t);
String vvvf=new String (") (#$genls ?POS #$Verb))");
CycList cl1=cycAccess.makeCycList(sssf+tttf+vvvf);
ArrayList<String> tp= new ArrayList<String>();
boolean connart=false;
try{
CycList res=cycAccess.askWithVariables(cl1,alst,mte1);
CycListVisitor clv0=res.cycListVisitor();
if(!clv0.hasMoreElements())
{
CycList xinfa=cycAccess.getDenotsOfString(wordtable.toLowerCase());
if(!xinfa.isEmpty())
{
for(int ti=0;ti<xinfa.size();ti++)
{
 if (xinfa.get(ti) instanceof CycConstant || xinfa.get(ti) instanceof CycNart)
 {
   tp.add(DefaultCycObject.cyclify(xinfa.get(ti)).toString());
   connart=true;
   }
}
```

```
}
}
 int yx=0;
while(clv0.hasMoreElements())
{
if(yx%4<3)clv0.nextElement();
else
 {
  Object verbtemp=clv0.nextElement();
  if (verbtemp instanceof CycConstant || verbtemp instanceof CycNart)
  {
  tp.add(DefaultCycObject.cyclify(verbtemp).toString());
  connart=true;
  }
 }
yx++;
}
}catch(Exception e){System.out.println("can not find  ");}
ArrayList<String> nncycs=new ArrayList<String>();
   if(connart)
   {
   int shu=0;
   String[] tp1=new String[tp.size()];
   tp1[0]=tp.get(0);
   shu=1;
   for(int ii=1;ii<tp.size();ii++)
   {
   String tmp=tp.get(ii);
   Boolean bmp=false;
   for(int jj=0;jj<shu;jj++)
   {
   if(tmp.equals(tp1[jj]))
   {
   bmp=true;   break;
   }
   }
   if(!bmp){  tp1[shu]=tmp;shu++;  System.out.println("IN not equal  "+tmp);}
   }
   for(int ii=0;ii<shu;ii++)
    nncycs.add(tp1[ii]);

   return nncycs;
   }
   return null;
}

public static ArrayList<String> nnprocess(CycAccess cycAccess, CycFort mte1,String wordtable)
{
char c[]=new char[1];   c[0]='\"'; String t=new String(c);
ArrayList alst=new ArrayList();
CycVariable cv1=CycObjectFactory.makeCycVariable("?WORD");
CycVariable cv2=CycObjectFactory.makeCycVariable("?POS");
CycVariable cv3=CycObjectFactory.makeCycVariable("?NUM");
CycVariable cv4=CycObjectFactory.makeCycVariable("?TERM");
alst.add(cv1);
alst.add(cv2);
```

```
alst.add(cv3);
alst.add(cv4);
String sssf=new String("(#$and (#$denotation ?WORD ?POS ?NUM ?TERM) (#$wordForms ?WORD
#$nounStrings ");
String nounword=new String(wordtable.substring(0,1)+wordtable.substring(1).toLowerCase());
String tttf=new String(t+nounword+t);
String vvvf=new String (") (#$genls ?POS #$Noun))");
CycList cl1=cycAccess.makeCycList(sssf+tttf+vvvf);
ArrayList<String> tp= new ArrayList<String>();
boolean connart=false;
try{
CycList res=cycAccess.askWithVariables(cl1,alst,mte1);
System.out.println(res.removeDuplicates());
CycListVisitor clv0=res.cycListVisitor();
if((wordtable.charAt(0)<=90) && (wordtable.charAt(0)>=65))
{
 String qtest0=new String("(#$termPhrases ?WHAT ?CONSTRAINT ");
 String qtest2=new String(")");
 String qtestfinal=new String(qtest0+t+nounword+t+qtest2);
 ArrayList qalstr=new ArrayList();
 CycVariable qcv1r=CycObjectFactory.makeCycVariable("?WHAT");
 CycVariable qcv3r=CycObjectFactory.makeCycVariable("?CONSTRAINT");
 qalstr.add(qcv1r),
 qalstr.add(qcv3r);
 CycList qc=cycAccess.makeCycList(qtestfinal);
 CycList qresr=cycAccess.askWithVariables(qc,qalstr,mte1);
 CycListVisitor clv00=qresr.cycListVisitor();
 if(clv00.hasMoreElements())
 {
 while(clv00.hasMoreElements())
 {
 Object nntemp=clv00.nextElement();
 if (nntemp instanceof CycConstant || nntemp instanceof CycNart)
 {
 tp.add(DefaultCycObject.cyclify(nntemp).toString());
 connart=true;
 }
 clv00.nextElement();
 }
 }
}
if(!clv0.hasMoreElements())
{
CycList xinfa=cycAccess.getDenotsOfString(wordtable.toLowerCase());
if(!xinfa.isEmpty())
{
for(int ti=0;ti<xinfa.size();ti++)
{
 if (xinfa.get(ti) instanceof CycConstant || xinfa.get(ti) instanceof CycNart)
    {
    tp.add(DefaultCycObject.cyclify(xinfa.get(ti)).toString());
    connart=true;
    }
}
}
else
```

```
{
 String qtest0=new String("(#$termPhrases ?WHAT ?CONSTRAINT ");
 String qtest2=new String(")");
 String qtestfinal=new String(qtest0+t+nounword+t+qtest2);
 ArrayList qalstr=new ArrayList();
 CycVariable qcv1r=CycObjectFactory.makeCycVariable("?WHAT");
 CycVariable qcv3r=CycObjectFactory.makeCycVariable("?CONSTRAINT");
 qalstr.add(qcv1r);
 qalstr.add(qcv3r);
 CycList qc=cycAccess.makeCycList(qtestfinal);
 CycList qresr=cycAccess.askWithVariables(qc,qalstr,mte1);
 CycListVisitor  clv00=qresr.cycListVisitor();
 if(clv00.hasMoreElements())
{
 while(clv00.hasMoreElements())
{
 Object nntemp=clv00.nextElement();
 if (nntemp instanceof CycConstant || nntemp instanceof CycNart)
 {
 tp.add(DefaultCycObject.cyclify(nntemp).toString());
 connart=true;
 }
 clv00.nextElement();
}
}
}
}
int yx=0;
if(clv0.hasMoreElements())
{
 while(clv0.hasMoreElements())
{
if(yx%4<3)clv0.nextElement();
else
 {
  Object nntemp=clv0.nextElement();
  if (nntemp instanceof CycConstant || nntemp instanceof CycNart)
  {
  tp.add(DefaultCycObject.cyclify(nntemp).toString());
  connart=true;
  }
  }
 yx++;
 }
 }
 }catch(Exception e){System.out.println("can not find  ");}
 ArrayList<String> nncycs=new ArrayList<String>();
   if(connart)
   {
   int shu=0;
   String[] tp1=new String[tp.size()];
   tp1[0]=tp.get(0);
   shu=1;
   for(int ii=1;ii<tp.size();ii++)
   {
   String tmp=tp.get(ii);
```

```
  Boolean bmp=false;
  for(int jj=0;jj<shu;jj++)
  {
  if(tmp.equals(tp1[jj]))
  {
  bmp=true;   break;
  }
  }
  if(!bmp){  tp1[shu]=tmp;shu++;  System.out.println("IN not equal  "+tmp);}
  }
  for(int ii=0;ii<shu;ii++)
  nncycs.add(tp1[ii]);
  return nncycs;
  }
 else  return null;
  }

 public static ArrayList<cizuji> npprocess(CycAccess cycAccess,int num,int i,Tree temp,CycFort
mtc,CycFort mt1)
{
ArrayList<cizuji> czj=new ArrayList<cizuji>();
try{
  List<Tree>  ltree=temp.getLeaves();
  int nplen=ltree.size();
  String[] lString=new String[nplen];
  for(int k=0; k<nplen; k++)  lString[k]=ltree.get(k).toString();
  if(nplen<2)return czj;
  ArrayList<Integer> remainsegpos=new ArrayList<Integer>();
  ArrayList<Integer> remainseglen=new ArrayList<Integer>();
   remainsegpos.add(0);
  remainseglen.add(nplen);
  int maxlen;
for(int stepsize=5; stepsize>1; stepsize--)
{
int ttp=0;
int x=remainseglen.size();
if(x==0)continue;
for(int pq=0;pq<remainseglen.size();pq++)
if(ttp<remainseglen.get(pq))ttp=remainseglen.get(pq);
maxlen=ttp;
ArrayList<Integer> altodelete=new ArrayList<Integer>();
if(maxlen<stepsize)continue;
int limt0=remainseglen.size();
for(int segno=0; segno<limt0;segno++)
{
int stackend=remainseglen.size();
if(remainseglen.get(segno)<stepsize)continue;
ArrayList<Integer> cycstartpos=new ArrayList<Integer>();
ArrayList<Integer> cycstartlen=new ArrayList<Integer>();
for(int ss=remainsegpos.get(segno); ss<=remainsegpos.get(segno)+remainseglen.get(segno)-stepsize; ss++)
{
int www=remainsegpos.get(segno)+remainseglen.get(segno)-stepsize;
{
  String stepsizeString=new String();
  for(int sgg=0; sgg<stepsize; sgg++)
  {
```

```
   stepsizeString+=lString[ss+sgg];
   if(sgg!=stepsize-1)stepsizeString+=" ";
  }
  CycList jieguo=cycAccess.getDenotsOfString(stepsizeString);
  if(!jieguo.isEmpty())
  {  ArrayList<CycFort> tang=new ArrayList<CycFort>();
    for(int tn=0;tn<jieguo.size();tn++)
    tang.add((CycFort)(jieguo.get(tn)));
    cizuji wei=new cizuji(ss,stepsize, tang);
    czj.add(wei);
    cycstartpos.add(ss);
    cycstartlen.add(stepsize);
    ss=ss+stepsize-1;
   }
  }
 }
if(cycstartpos.size()>0)
 {
  altodelete.add(segno);

  int cycsegsize=cycstartpos.size();
  for(int cc=cycsegsize-1; cc>0; cc--)
  {
   if(cycstartpos.get(cc)==cycstartpos.get(cc-1)+cycstartlen.get(cc-1))    {
     cycstartlen.set(cc-1, cycstartlen.get(cc)+cycstartlen.get(cc-1));
           cycstartpos.remove(cc);
           cycstartlen.remove(cc);
   }
  }
  cycsegsize=cycstartpos.size();
  if(cycstartpos.get(0)!=remainsegpos.get(segno))
  {
    remainsegpos.add(remainsegpos.get(segno));
    remainseglen.add(cycstartpos.get(0));
    for(int qt=0;qt<cycstartpos.size();qt++)
    remainsegpos.add(cycstartpos.get(qt)+cycstartlen.get(qt));
    for(int qts=1; qts<cycstartpos.size(); qts++)
    remainseglen.add(cycstartpos.get(qts)-remainsegpos.get(stackend+qts+1));
    remainseglen.add(remainsegpos.get(segno)+remainseglen.get(segno)-
    remainsegpos.get(stackend+cycsegsize));
  }
  else
  {
    for(int qt=0;qt<cycstartpos.size();qt++)
    remainsegpos.add(cycstartpos.get(qt)+cycstartlen.get(qt));
    for(int qts=0;qts<cycstartpos.size()-1;qts++)
    remainseglen.add(cycstartpos.get(qts+1)-remainsegpos.get(stackend+qts));
    remainseglen.add(remainsegpos.get(segno)+remainseglen.get(segno)-
remainsegpos.get(stackend+cycsegsize-1));
   }
  }
    int xsize=remainsegpos.size();
    for(int kk=xsize-1;kk>=limt0;kk--)
    {
    if(remainseglen.get(kk)<2)  { remainsegpos.remove(kk);  remainseglen.remove(kk); }
    }
```

```
}
   int lenel=altodelete.size();
   if(lenel>0)
    for(int ad=0;ad<lenel;ad++)
    {
     int emp=altodelete.get(ad);
     remainsegpos.remove(emp);
     remainseglen.remove(emp);
    }
}
}catch(Exception e){System.out.println("error in NP processing");}
 return czj;
  }

 public void extractsentence(double[] weight0, int[] cishu01,List<List<? extends HasWord>> document)
{
try{
            for(int d=0;d<cishu01.length;d++)
            {
            if(cishu01[d]<1) cishu01[d]=1;
            weight0[d]/=cishu01[d];
            if(cishu01[d]<4) weight0[d]=0.01;
            }
               weightpair[] weight1=new weightpair[document.size()];
               for(int w1=0;w1<weight0.length;w1++)
               {          weight1[w1]=new weightpair(weight0[w1],w1);          }
               int[] wg=new int[weight0.length];
               for(int qq=0;qq<weight1.length-1;qq++)
               {
               double bs=weight1[qq].getweight();
               int bt=qq;
               double bs1=bs;

               for(int ww=qq+1;ww<weight1.length;ww++)
               {
                bs1=weight1[ww].getweight();
                if(bs1>bs)
                {
                 bt=ww;
                 bs=bs1;
                 }
               }
               wg[qq]=bt;
               weightpair ppt=weight1[qq];
               weight1[qq]=weight1[bt];
               weight1[bt]=ppt;
               }
               int seglen=document.size()/3;
               FileWriter bypo = new java.io.FileWriter( "ByPosition.txt", false );
               PrintWriter bypoout = new java.io.PrintWriter( bypo, true );
                FileWriter bypo1 = new java.io.FileWriter( "ByPosition5.txt", false );
               PrintWriter bypoout1 = new java.io.PrintWriter( bypo1, true );
               int geshu=0;
               for(int gg=0;gg<weight0.length;gg++)
               {
               if(weight0[gg]>=weight1[seglen-1].getweight())
```

```
                {
                bypoout.println(document.get(gg)+ "  ");
                bypoout.println();
                bypoout.println(weight0[gg]+ "  ");
                geshu++;
                if(geshu==seglen) break;
                 }
                }
                 double weightmax;
                 if(seglen<=10) weightmax=weight1[seglen-1].getweight();
                 else weightmax=weight1[9].getweight();
                 for(int gg=0;gg<weight0.length;gg++)
                {
                if(weight0[gg]>=weightmax)
                {
                if(geshu<10)  bypoout1.println(document.get(gg)+ "\n");
                geshu++;
                if(geshu==seglen) break;
                 }
                }
                bypoout.close();
                bypoout1.close();
                FileWriter byimport = new java.io.FileWriter( "zByImportance.txt", false );
                PrintWriter byimportout = new java.io.PrintWriter( byimport, true );
                FileWriter byimport5 = new java.io.FileWriter( "zByImportance5.txt", false );
                PrintWriter byimportout5 = new java.io.PrintWriter( byimport5, true );
                for(int gg=0; gg<seglen;gg++)
                {
                byimportout.println(document.get(weight1[gg].getorder()));
                byimportout.println(weight1[gg].getweight()+" cishu "+cishu01[weight1[gg].getorder()]);
                if(gg<10)byimportout5.println(document.get(weight1[gg].getorder())+"\n");
                }
                byimportout.close();
                byimportout5.close();
} catch (Exception ioe) {
    System.out.println("wrong in extract");
  }
 }

  boolean ifstop(String wd, String[] stoplist )
 {
 int len=stoplist.length;
   boolean checked=false;
   for(int i=0; i<len; i++)
   {
     if(stoplist[i].equals(wd))
            {
              checked=true;
              break;
            }
   }
   if(checked)System.out.println("caught in func "+wd);
   return checked;
 }

 public String processfile(String filename0)
```

```
 {
 int i=0;
String filename = filename0 +"Y";
String thisLine;
 try
 {
FileInputStream pfin = new FileInputStream(filename0);
BufferedReader pInput = new BufferedReader (new InputStreamReader(pfin));
FileWriter prout = new java.io.FileWriter( filename, false );
PrintWriter prout1 = new java.io.PrintWriter( prout, true );
while ((thisLine = pInput.readLine()) != null)
{
int tm=thisLine.length();
if(tm<=0) continue;
char t=thisLine.charAt(tm-1);
if( (t<=57)&&(t>=48) || (t<=90)&&(t>=65)|| (t<=122)&&(t>=97)) prout1.println(thisLine+".");
else prout1.println(thisLine);
}
pInput.close();
prout1.close();
}
catch (IOException ioe) {
ioe.printStackTrace();
}
return filename;
  }


public boolean pushconfreq(CycObject newfort,ArrayList<CycObject>content, ArrayList<Integer> fq)
{
boolean newterm=true;
if(content.size()==0)
{
 content.add(newfort);
 fq.add(1);
}
else
{
 for(int i=0;i<content.size();i++)
 {
   if(newfort.equals(content.get(i)))
   {
     fq.set(i,fq.get(i)+1);
            newterm=false;
            break;
   }
 }
 if(newterm)
 {
   content.add(newfort);
   fq.add(1);
 }
}
return newterm;
}
```

```
private void parseFiles(String[] args, int argIndex, boolean tokenized, TokenizerFactory
tokenizerFactory,  DocumentPreprocessor documentPreprocessor, boolean fromXML,
String sentenceDelimiter, Function<List<HasWord>, List<HasWord>> escaper, int tagDelimiter)  {
PrintWriter pwOut = op.tlpParams.pw();
PrintWriter pwErr = op.tlpParams.pw(System.err);
TreePrint treePrint = getTreePrint();
int numWords = 0;
int numSents = 0;
int numUnparsable = 0;
int numNoMemory = 0;
int numFallback = 0;
int numSkipped = 0;
Timing timer = new Timing();
TreebankLanguagePack tlp = op.tlpParams.treebankLanguagePack();
// set the tokenizer
if (tokenized) {
  tokenizerFactory = WhitespaceTokenizer.factory();
}
if (tokenizerFactory == null) {
  tokenizerFactory = tlp.getTokenizerFactory();
}
documentPreprocessor.setTokenizerFactory(tokenizerFactory);
documentPreprocessor.setSentenceFinalPuncWords(tlp.sentenceFinalPunctuationWords());
documentPreprocessor.setEncoding(op.tlpParams.getInputEncoding());
boolean saidMemMessage = false;
timer.start();
for (int i = argIndex; i < args.length; i++) {
  String filename = processfile(args[i]);
  try {
    List<List<? extends HasWord>> document; // initialized just below
    if (fromXML) {
      document = documentPreprocessor.getSentencesFromXML(filename, sentenceDelimiter, true);
    } else {
      document = documentPreprocessor.getSentencesFromText
     (filename, escaper, sentenceDelimiter, tagDelimiter);
    }
    System.err.println("Parsing file: " + filename + " with " + document.size() + " sentences.");
    PrintWriter pwo = pwOut;
    if (Test.writeOutputFiles) {
       String ext = Test.outputFilesExtension == null ? "stp":
       Test.outputFilesExtension;
       String fname = filename + "." + ext;
       if (Test.outputFilesDirectory != null) {
       String fseparator = System.getProperty("file.separator");
       if (fseparator == null || fseparator.equals("")) {
         fseparator = "/";
       }
       int ind = fname.lastIndexOf(fseparator);
       fname = fname.substring(ind + 1);
       if ( ! "".equals(Test.outputFilesDirectory)) {
         fname = Test.outputFilesDirectory + fseparator + fname;
       }
      }
      try {
       pwo = op.tlpParams.pw(new FileOutputStream(fname));
      } catch (IOException ioe) {
```

```
   ioe.printStackTrace();
  }
}
int num = 0;
treePrint.printHeader(pwo, op.tlpParams.getOutputEncoding());
Log.makeLog();
Log.current.println("Initializing Cyc server connection, and caching frequently used terms.");
FileWriter sent = new java.io.FileWriter( "/home/mei/researchcyc-1.0/server/cyc/run/Sentences.txt",
 false );
PrintWriter sent1 = new java.io.PrintWriter( sent, true );
CycAccess cycAccess;
        int nono=0;
        ArrayList<ArrayList<String[]>> zuhe=new ArrayList<ArrayList<String[]>>();
        ArrayList<ArrayList<Integer[]>> zuhex=new ArrayList<ArrayList<Integer[]>>();
         int[] cishu= new int[document.size()];
        int[] wordcount=new int[document.size()];
        int[] conceptno=new int[document.size()];
        int[] punctno=new int[document.size()];
        int[] closepuncno=new int[document.size()];
        String[][] WordArray=new String[document.size()][];  /
        String[][] wordforms=new String[document.size()][];
        String[][] PosArray=new String[document.size()][];
        Double[][] FreqArray=new Double[document.size()][];
        int[][] pFreqArray=new int[document.size()][];
        nppairlist[] npgrouplist=new nppairlist[document.size()] ;
        CycFort[][] CycFortArray=new CycFort[document.size()][];
        String[][] CycStringArray=new String[document.size()][];
        CycFort[][][] CycFortGroupArray=new CycFort[document.size()][][];
        String[][][] CycStringGroupArray=new String[document.size()][][];
        int[][] biaozhi=new int[document.size()][];
        ArrayList<String> testnonCyc=new ArrayList<String>();
        ArrayList<Double> testnonCycFreq=new ArrayList<Double>();
        ArrayList<String> nonCyc=new ArrayList<String>();
        ArrayList<Double> nonCycFreq=new ArrayList<Double>();
        double[] weight0=new double[document.size()];
        int[] cishu01=new int[document.size()];
        int sentenceorder=0;
        ArrayList<?>[] zwbb=new ArrayList<?>[document.size()] ;
         try
         {
         cycAccess = new CycAccess();
        cycAccess.traceOn();
        CycFort mtc=cycAccess.getKnownConstantByName("EnglishMt");
        CycFort mt1=cycAccess.getKnownConstantByName("PennTagDataMt");
          CycFort mte=cycAccess.getKnownConstantByName("EverythingPSC");
          CycFort gemt=cycAccess.getKnownConstantByName("GeneralEnglishMt");
          CycFort pmt = cycAccess.getKnownConstantByName("ConWeightMt");
         CycConstant uvmt=cycAccess.getConstantByName("UniversalVocabularyMt");
         CycConstant countNo0=cycAccess.getConstantByName("freq");
         CycConstant zhuwb=cycAccess.getConstantByName("zwb");
         String[] stoplist=new String[548];
          BufferedReader br = new BufferedReader(new FileReader("stoplist1.txt"));
         int lln=0;
         String thisLine;
         while ((thisLine = br.readLine()) != null)
         stoplist[lln++]=thisLine;
```

```
            br.close();
            FileWriter swsd = new java.io.FileWriter( "sentencewsd.txt", false );
            PrintWriter swsdout = new java.io.PrintWriter( swsd, true );
            FileWriter ofw = new java.io.FileWriter( "generalized.txt", false );
            PrintWriter pout = new java.io.PrintWriter( ofw, true );
            FileWriter fw = new java.io.FileWriter( "double.txt", false );
            PrintWriter pw = new java.io.PrintWriter( fw, true );
            FileWriter fw1 = new java.io.FileWriter( "triple.txt", false );
            PrintWriter pw1 = new java.io.PrintWriter( fw1, true );
            FileWriter fw3 = new java.io.FileWriter( "fullGrammR.txt", false );
            PrintWriter pw3 = new java.io.PrintWriter( fw3, true );
            FileWriter fw4 = new java.io.FileWriter( "keyword.txt", false );
            PrintWriter pw4 = new java.io.PrintWriter( fw4, true );
            FileWriter fr = new java.io.FileWriter( "count.txt", false );
            PrintWriter wr = new java.io.PrintWriter( fr, true );

            num=-1;
            String[][] penntocyc=new String[16][2];
           penntocyc[0][0]=new String("#$regularDegree");          penntocyc[0][1]= new String("JJ");
           penntocyc[1][0]=new String("#$comparativeDegree");   penntocyc[1][1]= new String("JJR");
           penntocyc[2][0]=new String("#$superlativeDegree");    penntocyc[2][1]= new String("JJS");
           penntocyc[3][0]=new String("#$regularAdverb");        penntocyc[3][1]= new String("RB");
           penntocyc[4][0]=new String("#$comparativeAdverb");    penntocyc[4][1]= new
String("RBR");
           penntocyc[5][0]=new String("#$superlativeAdverb");    penntocyc[5][1]= new String("RBS");
           penntocyc[6][0]=new String("#$tensed");               penntocyc[6][1]= new String("VB");
           penntocyc[7][0]=new String("#$pastTense-Generic");    penntocyc[7][1]= new String("VBD");
           penntocyc[8][0]=new String("#$presentParticiple");    penntocyc[8][1]= new String("VBG");
           penntocyc[9][0]=new String("#$perfect-Generic");      penntocyc[9][1]= new String("VBN");
           penntocyc[10][0]=new String("#$pluralVerb-Generic");  penntocyc[10][1]= new
String("VBP");
           penntocyc[11][0]=new String("#$singularVerb-Generic"); penntocyc[11][1]= new
String("VBZ");
            penntocyc[12][0]=new String("#$nonPlural-Generic");   penntocyc[12][1]= new
String("NN");
            penntocyc[13][0]=new String("#$plural-Generic");      penntocyc[13][1]= new String("NNS");
            penntocyc[14][0]=new String("#$pnSingular");          penntocyc[14][1]= new String("NNP");
            penntocyc[15][0]=new String("#$pnPlural");            penntocyc[15][1]= new String("NNPS");
            Formatter formatter = new Formatter(sent1, Locale.US);
            for (List sentence : document)
           {
            num++;
            formatter.format("%5d  ", num);
            sent1.println(sentence);
              numSents++;
              int len = sentence.size();
            biaozhi[num]=new int[len];
            for(int zi=0;zi<len;zi++)
            biaozhi[num][zi]=-1;sentenceorder=num;
            WordArray[sentenceorder]=new String[len];
            wordforms[sentenceorder]=new String[len];
            PosArray[sentenceorder]=new String[len];
            FreqArray[sentenceorder]=new Double[len];
            pFreqArray[sentenceorder]=new int[len];
            CycFortArray[sentenceorder]=new CycFort[len];
            CycStringArray[sentenceorder]=new String[len];
```

```
            // CycFortArray[sentenceorder]=new CycFort[len];
             CycStringGroupArray[sentenceorder]=new String[len][];
          CycFortGroupArray[sentenceorder]=new CycFort[len][];
           numWords += len;
           pwErr.println("Parsing [sent. " + num + " len. " + len + "]: " + sentence);
        Tree ansTree = null;
        try {
         if ( ! parse(sentence)) {
          pwErr.print("Sentence couldn't be parsed by grammar.");
          if (pparser != null && pparser.hasParse() && fallbackToPCFG) {
           pwErr.println("... falling back to PCFG parse.");
           ansTree = getBestPCFGParse();
           numFallback++;
          } else {
           pwErr.println();
           numUnparsable++;
          }
         } else {
          ansTree = getBestParse();
    Tree trgbp=ansTree;
    kuk=new ArrayList<Tree>();
    qishi=new ArrayList<Integer>();
    wei=new ArrayList<Integer>();
    gongli=0;
    jiansuo(ansTree);
              Tree[] treechid=trgbp.children();
              List<Tree>  allchildren=new ArrayList<Tree>();
              ArrayList<Integer> juyustart = new ArrayList<Integer>();
              ArrayList<Integer> juyulen = new ArrayList<Integer>();
              allchildren.add(treechid[0]);
              juyustart.add(0);
              juyulen.add(Trees.leaves(ansTree).size());
              Iterator listcon=allchildren.iterator();
              int ghgh=0;
for (int ilist=0; ilist<allchildren.size(); ilist++)
{
Tree temp=allchildren.get(ilist);
int tempstart=juyustart.get(ilist);
int templen=juyulen.get(ilist);
String LabelOfNode=new String(temp.label().toString());
Tree[] allchid=temp.children();
 if(LabelOfNode.equals("NP"))
 {
List<Tree> kak=Trees.leaves(temp);
 boolean fg=false;
if(fg)
 {
 for (int jlist=0; jlist<allchid.length; jlist++)
 allchildren.add(allchid[jlist]);
 juyustart.add(juyulen.get(juyulen.size()-1)+juyustart.get(juyustart.size()-1));
 juyulen.add(allchid.length);
 }
 else
 {
 ArrayList<cizuji> guog= npprocess(cycAccess,num, i,temp, mtc, mt1);
if(!guog.isEmpty())
```

```
{
if(npgrouplist[num]==null)npgrouplist[num]=new nppairlist();
for(int yi=0;yi<guog.size();yi++)
{
 cizuji zhang=guog.get(yi);
 int qi=zhang.getstart();
 int chang=zhang.getlength();
 pFreqArray[num][qi+tempstart]=-npgrouplist[num].size()-1;
 ArrayList<CycFort> npcycfort=zhang.getcizu();
 nppair currentnp=new nppair(npcycfort,chang);
 npgrouplist[num].add(currentnp);
 ArrayList<String> zu=new ArrayList<String>();
 for(int q=0;q<npcycfort.size();q++)
 zu.add(DefaultCycObject.cyclify(npcycfort.get(q)).toString());
 for(int wu=0;wu<chang;wu++)
 {
 biaozhi[num][tempstart+qi]=qi;
 CycStringGroupArray[sentenceorder][tempstart+qi]=new String[zu.size()];
 }
 for(int liu=0;liu<zu.size();liu++)
 {
 CycStringGroupArray[sentenceorder][tempstart+qi][liu]=zu.get(liu);
 String target=zu.get(liu);
 if(target.startsWith("#$"))cycAccess.converseList("(qa "+target+" 1)");
 else
 {
              if(target.startsWith("("))
              {
              //if(target instanceof CycNart)
              {
              String mumu=new String("(qa (quote "+target +" ) 1)");
              cycAccess.converseList(mumu);
              }
              }
 }
 }
}
}
 }
}
                  if(!(LabelOfNode.equals("NP")))
                  {
                  for (int jlist=0; jlist<allchid.length; jlist++)
                    {
                          allchildren.add(allchid[jlist]);
                          Tree fei1=allchid[jlist];
                          int wrt=-1;
                          for(int gfd=0;gfd<kuk.size();gfd++)
                          {
                          Tree fei2=kuk.get(gfd);
                          if(fei2==fei1)
                          {
                          wrt=gfd;
                          break;
                          }
                          }
```

```
                juyustart.add(qishi.get(wrt));
                juyulen.add(wei.get(wrt));
                }
        }
        ghgh++;
}
        LabeledScoredConstituentFactory lcf=new LabeledScoredConstituentFactory();
        Set<Constituent> trcon=ansTree.constituents();
        Iterator itcon=trcon.iterator();

        ArrayList<Tree>  subtrees = new ArrayList<Tree>();
        ArrayList<pospair> senpos=new ArrayList<pospair>();
        for(int j=0;j<ansTree.numChildren() ;j++)
        {
        Tree branch=ansTree.getChild(j);
        if(!branch.isLeaf())subtrees.add(branch);
        }
        for(int k=0;k<subtrees.size();k++)
        {
        Tree node=subtrees.get(k);
        int nodesize=node.numChildren();
        if(nodesize>1)
          {
                for(int k1=0;k1<nodesize;k1++)
                {
                Tree branch1=node.getChild(k1);
                // subtrees.add(branch1);
                subtrees.add(k+1,branch1);
                }
                }
                if(nodesize==1)
                {
                 Tree branch2=node.firstChild();
                 if(branch2.isLeaf())
                 {
                 String wd, ps;
                 wd=branch2.value();
                 ps=node.value();
                 pospair pspair=new pospair(wd,ps);
                 senpos.add(pspair);
                 }
                 else subtrees.add(k+1,branch2);
                 }
        }
                 int j=0;
                for(int i0=senpos.size()-1;i0>=0;i0--)
                {
                pospair temp=senpos.get(i0);
                WordArray[sentenceorder][j]=temp.getword();
                PosArray[sentenceorder][j]=temp.getpos();
                String cycpos=null;
                for(int qy=0;qy<16;qy++)
                {
                 if(PosArray[sentenceorder][j].equals(penntocyc[qy][1]))
                 {
                 cycpos=penntocyc[qy][0];
```

```
                break;
                }
               }
               if(cycpos!=null)
               {
               String getlexword=new String("(#$wordForms ?WHAT "+cycpos+"
               \""+WordArray[sentenceorder][j]+"\")");
               //String getlexword=new String("#$wordForms ?WHAT "+cycpos+"
        \""+WordArray[sentenceorder][j]+"\"");
               CycList getlexwordlist=cycAccess.makeCycList(getlexword);
               CycVariable lexq=CycObjectFactory.makeCycVariable("?WHAT");
               CycList lexqres=cycAccess.askWithVariable(getlexwordlist,lexq,mtc);
               if(lexqres!=null)
                 if(lexqres.size()>0)
                   {
                   String lxword=DefaultCycObject.cyclify(lexqres.first()).toString();
                   wordforms[sentenceorder][j]=lxword;
                   }
               }
               j++;
               }
     SubPredObjgroup   spo=extractSubPredObj(
      WordArray.sentenceorder,num,pw,pw3,ansTree,tlp,sentence);
     zwbb[sentenceorder]=spo.getzwb();
     ArrayList<zhuweibin> tmps=spo.getzwb();
     zuhe.add(spo.getmtrx()); zuhex.add(spo.getitrx());
     pw1.println(sentence.toString());
     if(tmps!=null)
     if(tmps.size()>0)
     for(int sj=0;sj<tmps.size();sj++)
     {
     pw1.println(tmps.get(sj).getzhu()+"  "+tmps.get(sj).getwei()+"  "+tmps.get(sj).getbin());
     pw1.println(WordArray[sentenceorder][tmps.get(sj).getzhu()]+"
                  "+WordArray[sentenceorder][tmps.get(sj).getwei()]+"
                 "+WordArray[sentenceorder][tmps.get(sj).getbin()]);
     }
    for(int sws=0;sws<WordArray[sentenceorder].length;sws++)
   swsdout.print(WordArray[sentenceorder][sws]+" | ");
   swsdout.println();
   int count_nonword_num=0;
    CycObject[][] senlevelwsd=senWSD(cycAccess, WordArray[sentenceorder],
    PosArray[sentenceorder]);
    for(int sws=0;sws<WordArray[sentenceorder].length;sws++)
       {
                   swsdout.print(WordArray[sentenceorder][sws]+" | ");
                   if((senlevelwsd[sws]!=null)&&(senlevelwsd[sws].length>0))
                     {   for(int ww=0;ww<senlevelwsd[sws].length;ww++)
                        swsdout.print(senlevelwsd[sws][ww].toString()+"  ");
             }
                   swsdout.println();
                 }

      wordcount[sentenceorder]=WordArray[sentenceorder].length-count_nonword_num;
      sentenceorder++;     wr.println();
}
```

```
        CycConstant tap=cycAccess.getConstantByName("is-Underspecified");
        cycAccess.assertGaf(pmt,countNo0,tap,1);

    } catch (OutOfMemoryError e) {
     if (Test.maxLength != -0xDEADBEEF) {
      // this means they explicitly asked for a length they cannot handle. Throw exception.
      pwErr.println("NOT ENOUGH MEMORY " + Test.maxLength);
      throw e;
     } else {
      if ( ! saidMemMessage) {
       printOutOfMemory(pwErr);
       saidMemMessage = true;
      }
      if (pparser.hasParse() && fallbackToPCFG) {
       try {
        String what = "dependency";
        if (dparser.hasParse()) {
         what = "factored";
        }
        pwErr.println("Sentence too long for " + what + " parser.  Falling back to PCFG parse...");
        ansTree = getBestPCFGParse();
        numFallback++;
       } catch (OutOfMemoryError oome) {
        oome.printStackTrace();
        numNoMemory++;
        pwErr.println("No memory to gather PCFG parse. Skipping...");
        pparser.nudgeDownArraySize();
       }
      } else {
       pwErr.println("Sentence has no parse using PCFG grammar Skipping...");
       numSkipped++;
      }
     }
    } catch (UnsupportedOperationException uoe) {
     pwErr.println("Sentence too long (or zero words).");
     numWords -= len;
     numSkipped++;
    }
    try {
     treePrint.printTree(ansTree, Integer.toString(num), pwo);
    } catch (RuntimeException re) {
     pwErr.println("TreePrint.printTree skipped: out of memory");
     re.printStackTrace();
     numNoMemory++;
     try {
      treePrint.printTree(null, Integer.toString(num), pwo);
     } catch (Exception e) {
      e.printStackTrace();
     }
    }
   }
        sent1.close();
        wr.close();
        swsdout.close();
        cycAccess.converseList("(prop1 "+"0.05)");
        cycAccess.converseList("(propf1 "+"0.05)");
```

```
            cycAccess.converseList("(propf2 "+"0.05)");
            }
            catch (Exception e) {
          Log.current.errorPrintln(e.getMessage());
          Log.current.printStackTrace(e);
        }
        treePrint.printFooter(pwo);
        if (Test.writeOutputFiles) {
         pwo.close();
        }
        System.err.println("Parsed file: " + filename + " [" + num + " sentences].");
       } catch (IOException e) {
        pwErr.println("ERROR: Couldn't open file: " + filename);
       }
     } // end for each file args[argIndex]
     long millis = timer.stop();
     if ( saidMemMessage) {
      printOutOfMemory(pwErr);
     }

     }
}
```

REFERENCES

Baker, J.K. (1979). Trainable grammars for speech recognition. *The Journal of the Acoustical Society of America,* 65(1), 132.

Barzilay, R & Elbadad, M. (1999). Using lexical chains for text summarization. In Mani, I. & Maybury, M. T. (Eds.), *Advances in Automatic Text Summarization* (pp. 111-121). Cambridge: MIT Press.

Baxendal, P.B. (1958). Machine-made index for technical literature-an experiment. *IBM Journal of Research and Development,* 2:4, 354-361.

Booth, T. L. & Thomson, R. A. (1973). Applying probability measures to abstract languages. *IEEE Transactions on Computers*, C-22, 442–450.

Buckley, C. & Cardie, C. (1997). SMART Summarization System. In Firmin Hand, T. and Sundheim, B. (Eds.), *TIPSTER-SUMMAC Summarization Evaluation, Proceedings of the TIPSTER Text Phase III Workshop.*

Carroll, J., Minnen, G. &Briscoe, T. (1999). Corpus annotation for parser evaluation. In *Proceedings of the EACL workshop on Linguistically Interpreted Corpora*, (pp. 35-41)

Chaves, R. P. (2001). WordNet and Automated Text Summarization. In *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium*, (pp. 109-116).

Choi, B. (2008). Method and Apparatus for Creating New Sentences to Summarize Documents, *Report of Invention No. 2008-21*, Office of Intellectual Property, Louisiana Tech University, August 13, 2008.

Choi, B. and Huang, X.M. (2009). Text Summarization Using Concept Hierarchy. *International Conference on Agents and Artificial Intelligence (ICAART),* Proto, Portugal, (pp. 281-286).

Curtis, J., Cabral, J., Baxter, D. (2006). On the Application of the Cyc Ontology to Word Sense Disambiguation. *FLAIRS Conference* (pp. 652-657).

Doran, W. Stokes, N., Carthy, J., & Dunnion, J. (2004). Comparing lexical chain-based summarisation approaches using an extrinsic evaluation. *In the Proceedings of the Global WordNet Conference (GWC 2004).*

Edmundson, H.P. (1969). New Methods in Automatic Extraction. *Journal of ACM*, 16(2), 264-285.

Fellbaum, C. (1998). *WordNet. An electronic lexical database*. Cambridge: MIT Press.

Firth, J. R. (1935). The technique of semantics. *Transactions of the Philological Society*, 36-72.

Frakes, W.B. (1992). Stemming Algorithms. In Frakes, W.B. & *BAEZA-YATES, R.* (Eds.) , *Information Retrieval-Data Structures and Algorithms* (pp. 131-160), *Upper Saddle River:* Prentice Hall.

Halliday, M. & Hasan, R. (1976). *Cohesion in English*. London: Longman Publishing Group.

Hatch, P., Stokes, N., Carthy, J. (2000). Lexical Chaining for Web-Based Retrieval of Breaking News. *In the proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems AH2000* (pp. 327-330).

Hovy, E. H. & Lin, C-Y. (1997). Automated Text Summarization in SUMMARIST. *In the Proceedings of the ACL97/EACL97 Workshop on Intelligent Scalable Text Summarization*. Madrid, Spain.

Hovy, E.H. and Lin, C-Y. (1998). Automated Text Summarization and the SUMMARIST system. *TIPSTER Text Program Phase III final report*, October 1998.

Jackson, H. (2007). Stop Word List. http://www.thebananatree.org/stoplist.html

Jones, K. S. (1998). Automatic summarizing: factors and directions, In Mani, I. & Maybury, M. T. (Eds.), *Advances in Automatic Text Summarization*, Cambridge: MIT Press.

Klein, D. & Manning, C. (2003). Fast Exact Inference with a Factored Model for Natural Language Parsing. *In Advances in Neural Information Processing Systems 15*(pp. 3-10), Cambridge : MIT Press.

Klein, D. & Manning, C. (2003). Accurate Unlexicalized Parsing. *In Proceedings of the 41st Meeting of the Association for Computational Linguistics*, (pp. 423-430).

Kupiec, J., Pedersen, J. and Chen, F. (1995). A Trainable Document Summarizer. *In Proceedings of the Eighteenth Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR)*, (pp. 68–73).

Li, S., You, W. , Li, T., Chen, H. (2004).Lexical Chain and its application in text filtering. *ITCC (2),* pp. 288-292

Lin, C-Y. (1995). Knowledge Based Automated Topic Identification. *In the Proceedings of the the 33rd Annual Meeting of the Association for Computational Linguistics.* pp. 308-310. Morristown: Association for Computational Linguistics.

Lin, C. Y. and Hovy. E.H. (1997). . Identifying Topics by Position. *In the Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP97).* Washington, D.C.

Lin, C-Y. (1998). Assembly of Topic Extraction Module in SUMMARIST. *In Working Notes of AAAI Spring Symposium on Intelligent Text Summarization.* Stanford, California.

Lin, C-Y. (1999). Training a Selection Function for Extraction. *In the 8th International Conference on Information and Knowledge Management (CIKM 99)*, pp. 55-62.

Luhn, H. P. (1958). The Automatic Creation of Literature Abstracts, *IBM Journal of Research and Development*, 2: 2, pp. 159-162.

Mani, I. & Maybury, M. (2001). *Advances in Automatic Text Summarization.* Cambridge: The MIT Press.

Mann, W.C. and Thompson, S.A. (1988). Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. Text 8(3), 243-281.

Marcu, Daniel. (1997). The Rhetorical Parsing of Natural Language Texts. *The Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, (ACL'97/EACL'97).* pp. 96-103.

Marcu, D. (1999). Discourse trees are good indicators of importance in text. In I. Mani and M. Maybury (Eds), *Advances in Automatic Text Summarizations*(pp. 123-136), Cambridge: The MIT Press.

Marneffe, M.C., MacCartney, B., Manning, C.D. (2006).). Generating Typed Dependency Parses from Phrase Structure Parses. *In Proceedings of the Fifth international Conference on Language Resources and Evaluation*, pp. 449-454.

Mitra, M., Singhal, A., & Buckley, C. (1997). Automatic Text Summarization by Paragraph Extraction. In *Proceedings of the Workshop on Intelligent Scalable Summarization at the ACL/EACL Conference*, pp. 39–46.

Morris, J. & Hirst, G. (1991). Lexical cohesion computed by thesaural relations as an indicator of the structure of text, *Computational Linguistics* 17(1): 21-43.

Morton, J. (1987-A). Tree Tomato. In Morton, J., Fruits of warm climates (pp. 437–440). Miami: Florida Flair Books.

Morton, J. (1987-B). Grapefruit. In Morton, J., Fruits of warm climates (pp. 152–158). Miami: Florida Flair Books.

Passonneau, R. J and Litman, D. H. (1997). Discourse segmentation by human and automated means. *Computational Linguistics*, pp. 103-139.

PBS (1999). Horses. www.pbs.org/wnet/nature/episodes/horses/introduction/3153/.

Salton, G.& McGill, M. (1983). *Introduction to modern information retrieval.* New York: McGraw Hill Book Co.

Salton, G., Allen, J., Buckley, C., and Singhal, A. (1994). Automatic Analysis, Theme Generation, and Summarization of Machine-Readable Texts. *Science*, 264, 1421–1426.

Salton, G., Singhal, A. , Mitra , M., Buckley, C.(1997). Automatic Text Structuring and Summarization. *Information Processing & Management*, 33(2), 193–207.

Santorini, B. (1990). Part-of-Speech Tagging Guidelines for the Penn Treebank Project, ftp://ftp.cis.upenn.edu/pub/treebank/doc/tagguide.ps.gz.

Silber, H.G. & McCoy, F. K. (2000). Efficient text summarization using lexical chains. In *Proceedings of the 13th International Conference on Intelligent User Interfaces, IUI2000* (pp. 252-255).

Silber, H.G. & McCoy, F. K. (2002). Efficiently Computed Lexical Chains As an Intermediate Representation for Automatic Text Summarization. *Computational Linguistics*, 28:4, 487-496.

St. Onge, D. (1995). *Detection and Correcting Malapropisms with Lexical Chains*, M.Sc. Thesis, University of Toronto, Canada.

Teufel, S. & Moens, M. (1997). Sentence Extraction as a Classification Task. In *Proceedings of the Workshop on Intelligent Scalable Summarization. ACL/EACL Conference* (pp. 58–65).