Keyphrase Extraction: A Tool for Design Patterns

By
Qi Wu

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Master of Science in Information Systems Science

School of Computer Science
Carleton University
Ottawa, Ontario

March 2002

© Copyright 2002, Qi Wu

National Library
of Canada

Acquisitions and
Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale
du Canada

Acquisitions et
services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-79800-3

Canada

The undersigned hereby recommend to
The Faculty of Graduate Studies and Research
Acceptance of the thesis

Keyphrase Extraction: a Tool for Design Patterns

submitted by

Qi Wu

In partial fulfillment of the requirements for the degree of
Masters of Science in Information and Systems Science

Director, School of Computer Science

Thesis Supervisor

Carleton University
March, 2002

Abstract

In the last ten years, software developers and designers in the Object-Oriented programming community have created a large collection of design patterns. The rapid development of the collection of patterns has brought about a searching bottleneck between pattern users and the large number of published patterns; they have to review to find the ones they need. Since developers can benefit from using patterns, it is important to help them find the ones they require in a timely manner.

A keyphrase, which represents the subject contents of an article, is one aid that helps readers to identify papers of interest. Keyphrases can also be a suitable basis for implementing a searchable index. However, we have found that many pattern-related articles and papers do not list keyphrases.

This thesis develops a keyphrase extraction tool that can extract keyphrases from design pattern related articles. The algorithm developed is an adapted version of GenEx, which is an algorithm designed for the keyphrase extraction on wide-category, domain independent articles. We adapted GenEx to the language characteristics of the pattern domain in order to give a better extraction performance than the non-adaptive version of the algorithm.

Acknowledgements

I would like to especially thank my supervisor, Dr. Dwight Deugo, for his research guidance and personal encouragement throughout this thesis.

I would like to dedicate this thesis to my parents, my brother and my wife for their love and support over all the years.

I gracefully acknowledge Miss Jennifer Jenson, Dr. Weiping Liu and his family's sincere support during my study period.

Table of Contents

Abstract	iii
Acknowledgements	iv
<u>Chapter 1</u> Introduction	1
1.1 Problem and Motivation	1
1.2 The Expected Contributions	2
1.3 Thesis Outline	3
<u>Chapter 2</u> Background Literature	5
2.1 Research on Keyphrase Extraction and Related Area	5
2.1.1 <i>Important concepts for</i> <i>extraction performance evaluation</i>	5
2.1.2 <i>Text summarization and information extraction</i>	8
2.1.3 <i>Different approaches on keyphrase extraction</i>	11
2.1.4 <i>Another related work - index</i>	12
2.2 Algorithms for Keyphrase Extraction	13
2.2.1 <i>Microsoft Word97</i>	13
2.2.2 <i>Part-of-Speech Tagger: frequent noun phrases</i>	14
2.2.3 <i>Verity's Search 97</i>	15
2.2.4 <i>Kea</i>	15

2.3	GenEx - a Supervised Machine Learning Algorithm	16
2.3.1	<i>Keyphrase extraction procedures of GenEx</i>	16
2.3.2	<i>Processing procedures of Extractor in GenEx</i>	18
2.3.3	<i>Supervised learning of GenEx algorithm</i>	20
2.4	Summary	21
 Chapter 3 Software Design Pattern and Application in		
	the Design of Pattern-Specific Extraction Tool	22
3.1	Structure of Software Design Patterns	22
3.2	Pattern Languages	26
3.3	Selection of Design Patterns for Development of the Extraction Tool	27
3.3.1	<i>Extraction procedures</i>	27
3.3.2	<i>Different mechanisms of keyphrase extraction</i>	28
3.3.3	<i>Modeling design and specifications</i>	29
3.3.4	<i>Selection of patterns for design and implementation</i>	30
3.3.5	<i>Determination of the design patterns to be used</i>	32
3.4	Summary	34
 Chapter 4 Important Concepts in the Design of Extraction Tool		
		35
4.1	Phrase Occurring Frequency	35
4.2	Observations through Experimentation	37
4.3	Stopwords	39
4.4	Stemming	42
4.5	Summary	45
 Chapter 5 Adaptation of GenEx to Pattern-Specific Extraction		
		46
5.1	Process of Supervised Learning of GenEx Algorithm	46
5.2	Parameters of the Extractor	47
5.3	Adaptation of GenEx to Pattern Specific Extraction	48
5.3.1	<i>Observation of keyphrase extraction on</i>	

<i>pattern articles with GenEx</i>	48
5.3.2 <i>Adaptation with parameters</i>	51
5.3.3 <i>Adaptation with other factors</i>	54
5.3.4 <i>From supervised machine learning to manual tuning</i>	54
5.4 Summary	55
Chapter 6 Evaluation of Performance of the Extraction Tool	56
6.1 Selection of Extractors and Evaluation Corpora	56
6.1.1 <i>Three extractors joining the evaluations</i>	57
6.1.2 <i>Selection of evaluation corpora</i>	57
6.1.3 <i>Evaluation scheme</i>	58
6.2 Results of Evaluation of Three Extractors	59
6.2.1 <i>Average extraction rates</i>	59
6.2.2 <i>Three measures</i>	60
6.2.3 <i>Extraction results with different sizes of articles</i>	63
Chapter 7 Performance Analysis	65
7.1 Discussion on average extraction rates	65
7.2 Discussion on Precision, Recall and F-measure	66
7.2.1 <i>F-measure</i>	66
7.2.2 <i>Recall</i>	67
7.2.3 <i>Precision</i>	67
7.2.4 <i>Difference of 3-measure values with training</i> <i>corpora and testing corpora</i>	67
7.3 Zero Measure Values and the Effect on Extraction	68
7.4 Effect of the Article Size on Performance	70
7.5 Extraction on other Articles	71
7.6 Summary	71
Chapter 8 Conclusions	73

8.1 Summary of the Thesis	73
8.2 Future Work.....	74
Bibliography.....	75
Appendix 1	80
A1.1 Extraction Results with Training Corpora	80
A1.2 Extraction Results with Testing Corpora	87
Appendix 2	92
Appendix 3	98

List of Tables

3.1 Two design patterns that process data streams.....	31
3.2 Some design patterns that have hierarchical structure	31
4.1 Occurring frequencies of 1-word phrases.....	38
4.2 Occurring frequencies of 2-word phrases.....	38
4.3 Occurring frequencies of 3-word phrases.....	38
4.4 Occurring frequencies of 1-word phrases after removing stopwords	40
4.5 Occurring frequencies of 2-word phrases after removing stopwords	41
4.6 Occurring frequencies of 3-word phrases after removing stopwords	41
5.1 Parameter definition of Extractor.....	47
5.2 Parameters of Extractor	48
5.3 Distributions of extracted 1-, 2- and 3-word keyphrase - GenEx extractor	49
5.4 Distributions of 1-, 2- and 3-word keyphrase - author provided.....	50
5.5 Parameter definition of adapted algorithm	53
5.6 Parameter type and value ranges of adapted algorithm.....	53

6.1 Measures of extraction on training corpora with our extraction tool.....	61
6.2 Measures of extraction on training corpora with GenEx demo program	61
6.3 Measures of extraction on training corpora with Word97	61
6.4 Measures of extraction on testing corpora with our extraction tool.....	62
6.5 Measures of extraction on testing corpora with GenEx demo program	62
6.6 Measures of extraction on testing corpora with Word97	62
7.1 F-measure of 3 different extraction algorithms on both corpora.....	66
7.2 Zero-measure-values of the extractors with both corpora	69

List of Figures

2.1 Presentation of extraction performance in set theory.....	8
2.2 Illustration of GenEx Keyphrase Extraction.....	17
2.3 Conceptual Diagram of GenEx Algorithm.....	18
3.1 Structure of Filters	28
3.2 Multi-inputs Filter Data Flow.....	29
3.3 Class Diagram of Extraction Tool.....	33
6.1 Performance against the number of training documents	58
6.2 Average extraction rates for 3 extractors on 2 types of corpora	60
6.3 Average F-measures for 3 extractors on 2 types of corpora	63
6.4 F-measures with size-incremented training corpora	64
7.1 Average of zero-measure-values.....	69

Chapter 1 Introduction

1.1 Problem and Motivation

Software design patterns, which incorporate many of the advanced OOA/OOD techniques, have been proposed to describe the solutions to software design and implementation problems. They can improve the structure of software, simplify software maintenance, help avoid architectural drift, improve communication among software developers and empower less experienced personnel to produce high-quality software. Because software design patterns describe document recurring solutions using a standard format, facilitate the communication among software developers during system design and implementation and enhance software engineering, they are especially useful for the teamwork environment [2].

Today, there exist a large collection of design patterns, which cover many aspects of software engineering; more and more experienced software developers and researchers continue to contribute their experiences in the form of design patterns [3]. Different international conferences are held yearly with titles on software design patterns, such as PLOP [4], EuroPLOP [5] and KoalaPLOP [6].

There are also workshops on design patterns, like OOPLSA [7], ECOOP [8]. Besides, many special column web pages; such as, Hillside Pattern Home [9], Pattern Depot [10], Open Directory [11], The Pattern Almanac [12], The Interaction Design Pattern Page [13], and many personal web pages especially discuss software design patterns.

Keyphrases, which represent the subject contents of an article, are one aid to identify papers of interest. They can also be a suitable basis for implementing a searchable index. However, only a small number of the many design pattern related articles available on-line identify the patterns' keywords/keyphrases. "This is a permanent fault" [14], commented an organizer of a software design pattern conference, and is the problem that this thesis addresses.

1.2 The Expected Contributions

At present, there exist some keyphrase extraction algorithms and they can be used to extract keyphrases from pattern articles, but these algorithms are wide category, domain independent. The expected contribution of the thesis is to select an existing algorithm and adapt it to the characteristics of pattern articles aiming to obtain a higher keyphrase extraction performance than non-adapted ones.

The steps of the adaptation to reach the goal are as follows:

- Compare and analyze different keyphrase extraction algorithms and select a proper one for the adaptation. In the thesis, GenEx algorithm is the selected algorithm for adaptation.

- Analyze GenEx algorithm and determine the parameters that can reflect the general syntactic properties of an article.
- Analyze the syntactic characteristics of design pattern articles.
- Adapt GenEx algorithm to the characteristics of pattern articles.
- Develop an extraction tool with the adapted version of GenEx algorithm.
- Tune the extraction tool by adjusting the values of parameters that affect extraction results.
- Perform evaluation on the developed extraction tool by measuring average extraction rate, Precision, Recall and F-measure, together with the parallel comparisons with the other two different extractors, i.e., GenEx keyphrase extractor and the keyphrase extractor of Word 97.

1.3 Thesis Outline

Chapter 2 surveys the background literature on keyphrase extraction and other keyphrase extraction related natural language processes. Five extraction algorithms are reviewed. The characteristics of one of the algorithms, GenEx are deeply analyzed, since it is selected for the adaptation to pattern articles in the thesis. Chapter 3 studies the characteristics of software design patterns and design pattern articles, which is the basis for the adaptation. Then the design patterns for data-stream process and object management are introduced; some of them will be chosen for the design and implementation of the extraction tool. In Chapter 4, some important approaches about extraction design are deeply

investigated, such as occurrence frequency, stopwords and stemming. Chapter 5 introduces the parameters that reflect the characteristics of pattern articles and other factors, then discusses how to carry out the adaptation of GenEx to these characteristics and factors from different angle of points. Chapter 6 introduces the evaluation procedures and the results. As the important part of the thesis, it discusses the selections of the evaluation corpora, other keyphrase extractors for the comparisons and evaluation scheme. It then lists all the experimental and calculated results. Chapter 7 compares and discusses the extraction results from practical extractions on both training corpora and testing corpora, with GenEx online demon program, the keyphrase extraction function of MS Word97 and our developed extraction tool, in terms of average extraction rate, three common extraction performance measures and zero-measure-values. The conclusions are given in Chapter 8.

Chapter 2 Background Literature

To develop a pattern-specific extraction tool, we need a good understanding of keyphrase extraction mechanism. In this chapter, the research areas closely related to keyphrase extraction in natural language process (NLP) are introduced, following by the review of the different approaches of keyphrase extraction and then, five prominent keyphrase extraction algorithms are reviewed. The emphasis will be put on GenEx, which is the algorithm to be used for the adaptation to the pattern-specific extraction.

2.1 Research on Keyphrase Extraction and Related Area

Before starting the review of the background literature to facilitate the later discussion, we first introduce some concepts on extraction performance evaluation.

2.1.1 Important concepts for extraction performance evaluation

- Precision and recall

Precision measure is the ratio of correctly extracted information against all the extracted information. Recall measure is the ratio of correct information extracted from a text against all available information [54]. Here, the word "correctly" or "correct" is in agreement between the authors and the extraction tool. For the description of keyphrase extraction, if m is the number of phrases identified as keyphrases by author, n is the number of phrases identified as keyphrases by the extractor, and a is classified as a keyphrase by both the author and the extractor, then,

- Precision of the extraction performance $p = a / n$ and
- Recall of the extraction performance $r = a / m$.

For the convenience of calculation, we further introduce another descriptions of the same concepts into our discussion. Let's declare b as the number of phrases identified as keyphrases by the extractor, but not identified as keyphrases by the author, and c as the number of phrases classified as keyphrases by the author, but not classified as keyphrases by the extractor [1]. Then,

$$m = a + c \text{ and } n = a + b.$$

Incorporated into the above formulas, we have:

- Precision $p = a / n = a / (a + b)$
- Recall $r = a / m = a / (a + c)$

Measurements of Precision and Recall reflect the degree of accuracy of the extraction. They are equally important, but negatively correlated, because

there is a trade-off between Precision and Recall. If Recall is 100%, Precision is close to 0. If Precision is 100%, then Recall is close to 0 [1]. Only using either of the methods for performance evaluation probably leads to only focusing on extreme case of one measure and overlooking the performance as a whole, so a comprehensive measure is needed.

- F-measure

A combination of the measures of Precision and Recall was introduced in the 4th Message Understanding Conference (MUC-4) [55], which is a single measure, called F-measure, and denoted by F .

$$F = ((\beta^2 + 1.0) * Precision * Recall) / (\beta^2 * Precision + Recall)$$

$$= ((\beta^2 + 1.0) * p * r) / (\beta^2 * p + r)$$

Define β as the relative weight in the evaluation of Precision and Recall. Make Precision and Recall have equal weights, i.e., $\beta = 1.0$. Then,

$$F = 2 * (Precision * Recall) / (Precision + Recall)$$

$$= 2 * (p * r) / (p + r) = 2a / (2a + b + c)$$

Let's change a point of view. Keyphrase extraction is a set classification problem. Phrases in a text can be classified as keyphrases or non-keyphrases. We assume that author provided keyphrase set to be a standard set. Therefore, the evaluation performance is changed to examine how large the intersection area a of the left circle ($a + c$) and the right circle ($a + b$), where the outer circle S represents the set of all phrases in an article. The left circle represents the standard keyphrase set m and the right circle represents the extractor extracted keyphrases

set n . The area of the intersection a of the two circles m and n presents the degree to which the two sets match each other. Obviously, the larger the intersection area a , the higher the extraction performance of the extractor.

Standard set $(a + c)$ is the number of author provided keyphrases, which assumes that all the keyphrases are in the space of candidate phrases (in fact, the practical cases are not always like so; some authors provide keyphrases that are not shown in the original articles. We will have deep discussion in later articles).

The extraction performance in set theory is illustrated in Figure 2.1:

Figure 2.1 Extraction performance in set presentation

2.1.2 Text summarization and information extraction

Research in language processing and knowledge discovery from a text is divided into four major directions according to the types of issues being addressed: Information retrieval (IR), text categorization, text summarization, and

information extraction [15]. Among them, text summarization and information extraction are the closest to keyphrase extraction [1].

- Text summarization involves summarizing documents by either collecting a set of key sentences from the text as the output, or generating sentences based on the structured template instantiated with the relevant information. Some research papers explore the task of producing a document summary by extracting key sentences from that textual document. The related research have been reported by Luan [16]; Edmundson [17]; Marsh *et al.* [18]; Paice and Jones [19]; Johnson *et al.* [20]; Salton *et al.* [21]; Kupiec *et al.* [22]; Brandow *et al.* [23]; Jang and Myaeng, [24].

Text summarization is similar to the task of keyphrase extraction, but it is more difficult. The extracted sentences often lack cohesion because anaphoric references are not resolved [20, 23]. Anaphors are pronouns (e.g., “it”, “they”), definite noun phrases (e.g., “the car”), and demonstratives (e.g., “this”, “these”) that refer to previously discussed concepts. When a sentence is extracted out of the context from its neighboring sentences, it may be impossible, or very difficult, for the reader of the summary to determine the referents of the anaphors. Johnson *et al.* [20] attempt to automatically resolve anaphors, but their system tends to produce an overly long summary [1]. Keyphrase extraction avoids this problem because anaphors are not included in the keyphrases. Also, a list of keyphrases has no structure. Unlike a list of

sentences, a list of keyphrases can be randomly permuted without significant consequences.

- Information extraction (IE) is concerned with isolating relevant text fragments within the document, extracting the relevant information from the fragments and then piecing together the target information in a coherent framework [18]. An information extraction system seeks specific information in a document according to predefined guidelines to a given topic area. For example, if the topic area concerns job advertisements, the guidelines might specify that the information extraction system should identify (a) the job title for the position, (b) the position responsibility and (c) the salary and other related information.

Most information extraction systems are knowledge-based systems. The highest performance achieved was reported at the 5th Message Understanding Conference (MUC-5) [25]. The work has demonstrated that a learning algorithm can perform as well as a manually constructed system. For example, Soderland and Lehnert [26] use decision tree induction as the learning component in their information extraction system. We may view the predefined guidelines for a given topic area as defining a template to be filled in by the information extraction system.

Information extraction and keyphrase extraction are also very closely related to each other, but with different focuses. Information extraction is detailed and general, while keyphrase extraction is specific and condensed [1]. Such differences require substantially different algorithms.

2.1.3 Different approaches on keyphrase extraction

Following are related scientific works on keyphrase extraction with different approaches.

- Muñoz [27] uses an unsupervised learning algorithm to discover two-word keyphrases. His algorithm is based on Adaptive Resonance Theory (ART) of neural networks. A large list of phrases tends to be produced with Munoz's algorithm, so the precision and F-measure are lower. Also, the algorithm is only applicable to two-word keyphrases [1].
- Nakagawa [28] works on automatic term recognition based on statistics of compound nouns, to extract terms consisting of simple and compound nouns. His algorithm is based on two essential aspects of the nature of terms, namely Unithood and Termhood, followed by quantifying the terms, scoring each candidate, and then finding final terms. The experimental results of his method are promising with higher values for Recall and F-measures, but his algorithm specifically works on Japanese and the goal is to obtain terms in a text, not as keyphrases.
- Krulwich and Burkey [29] use heuristics to extract significant phrases from a document. Their approach is based on syntactic clues, such as the use of italics, the presence of phrases in section headers, and the use of acronyms. The motivation is to produce phrases for use as features when automatically classifying documents, but the algorithm tends to produce a relatively large list of phrases, so the Precision and F-measure are lower [1].

- The mutual information statistic is another approach to discover two-word keyphrases. Steier and Belew [30] have done a large amount of work on it. This approach has the same limitations as Muñoz [27] because when considered as a keyphrase extraction algorithm, only two-word phrases are produced, and the Precision is lower [1].

2.1.4 Another related work - index

Another related work addresses the problem of automatically creating an index. More people have put efforts on it, such as, Fagan [31], Salton [21], Ginsberg [32], Leung and Kan [33]. Among them, Leung and Kan [33] provide a good survey of this work. There are two general classes of indexes: indexes that are intended for human readers to browse (back-of-book indexes) and indexes that are intended for use with information retrieval software (search engine indexes). Search engine indexes are not suitable for human browsing, since they usually index every occurrence of every word (excluding stopwords, such as, “then” and “how”) in the document collection. Back-of-book indexes tend to be much smaller, since they index only important occurrences of interesting words and phrases.

Back-of-book indexes are more relevant to keyphrase extraction for human browsing than search engine indexes. Leung and Kan [33] address the problem of learning to assign index terms from a controlled vocabulary. It involves building a statistical model for each index term in the controlled vocabulary. The statistical model attempts to capture the syntactic properties that distinguish documents where the given index term is appropriate from documents where it is

inappropriate. Their results are interesting, but the use of a controlled vocabulary makes it difficult to compare their work with the algorithms we examine here. Turney [1] presented a study with a small sample of controlled index terms in the INSPEC database. He found that very few of these terms appear in the bodies of the corresponding documents. It seems that algorithms that are suitable for automatically generating controlled index terms are substantially different from algorithms that are suitable for automatically extracting keyphrases.

2.2 Algorithms for Keyphrase Extraction

Some of keyphrase extraction algorithms have been put into practical applications. They are keyphrase extraction algorithms of Microsoft Word97, Part-of-Speech Tagger – frequent noun phrases, Verity's Search 97, Kea and GenEx.

2.2.1 Microsoft Word97

Microsoft's Word97 is a complete word processing software package. We are only concerned with its AutoSummarize feature, because of its keyword extraction ability. When a user opens a Word document and selects the AutoSummarize feature from the Tools menu, Word97 will ask the user to select Type of Summary. It then displays the summary of the text based on the selection. The identified sentences can be highlighted or separated from the remainder of the text. Also, the user can specify a target percentage of the text for AutoSummarize to condense. Next, the user selects Properties from the File menu and corresponding to the AutoSummarize process just completed, Word97 will list the keyphrases of

the text. The function of AutoSummarize always generates exactly five keyphrases and the keyphrases are always single words. So, they are called Keywords in Word97, not Keyphrases. For the convenience of later discussion, we treat "keyword" and "one-word keyphrase" interchangeably.

The keyphrase extraction of Word97 is probably the most convenient of all the algorithms to use because the document to be extracted was written in Word and the keyword extractor was embedded in Word. However, the users of Word97 can not customize the AutoSummarize feature. For example, it is not possible for the user to ask Word for six 1-word keyphrases instead of five or to ask for 2-word or 3-word keyphrases.

2.2.2 Part-of-Speech Tagger: frequent noun phrases

A second algorithm for keyphrase extraction is Part-of-Speech Tagger: Frequent Noun Phrases [34]. This algorithm claims that mostly all of the target keyphrases are noun phrases. Keyphrases often correspond to frequently occurring noun phrases in a text. The following is the essential algorithm of Part-of-Speech Tagger for automatic keyphrase extraction [34]:

- *Tag each word in the given document by its part-of-speech*
- *Look for all sequences of tagged words that match the pattern*
- *Count the frequency of each matching sequence*
- *Output the N most frequent matching sequences*

There exist some difficulties in using this algorithm. Turney implemented experiments on the keyphrase extraction algorithm [35]. In using Eric Brill's part-

of-speech Tagger for the experiment, he found that there were two problems with the algorithm. First, the N most frequent matching sequences tend to be almost entirely phrases consisting of a single word. If we want to extract multi-word keyphrases, then the algorithm needs some adjustment. Second, the algorithm does not recognize phrases that are minor variants of each other, such as *pattern?*, *patterns?* and *patterning?* [1].

2.2.3 Verity's Search 97

Verity's Search 97 is a complete text retrieval system [36]. Its Summarize feature enables the system to display summaries of documents, along with the document title. A summary in Search 97 consists of a list of sentences, with highlighted keyphrases embedded in the sentences.

The user can control the summary output, such as, the number of sentences, by either specifying the number of sentences desired or the percentage of the source document desired. The problem is that the user cannot directly isolate the embedded keyphrases from the original text.

2.2.4 Kea

Kea [37] is a supervised learning algorithm to automatically extract keyphrases from a text. The algorithm first goes through the preliminary steps, i.e., tokenizing and pre-processing of each word to get candidate keyphrases with attribute values. Then the Naïve Bayes machine-learning algorithm is used to compare attribute values for the known keyphrases with those for all the other candidates, and a model is built for determining whether or not a candidate is a

keyphrase. The Naïve Bayes scheme learns two sets of numeric weights from the attribute values. One set applying to positive ("is a keyphrase") examples and the other to negative ("is not a keyphrase") ones. Using these values, the model predicts whether a candidate is or is not a keyphrase.

The basic part of the Kea algorithm follows the same idea as GenEx algorithm. The experimental data by Turney indicate that Kea and GenEx have statistically equivalent levels of performance [1]. In section 2.3, we go deep into GenEx algorithm.

2.3 GenEx - a Supervised Machine Learning Algorithm

The GenEx algorithm [1] is geared to develop a system that finds a way to embed the specialized procedural domain knowledge into the keyphrase extraction algorithm. GenEx treats keyphrase extraction as a supervised learning problem and it uses a machine-learning mechanism to predict which candidate phrases are keyphrases.

2.3.1 Keyphrase extraction procedures of GenEx

From the viewpoint of software architecture, GenEx includes two components, Extractor algorithm for keyphrase extraction [1] and Genitor algorithm for extraction performance tuning of Extractor [38]. In Extractor, a group of parameters are introduced and the value ranges of the parameters reflect the factors of keyphrase locations. Then the parameters are optimized using Genitor genetic algorithm [38], so as to improve the extraction performance. Genitor is

only used to tune Extractor. Once the training process is complete, Genitor is no longer needed. GenEx takes a document as input and produces a list of keyphrases as output.

Figure 2.2 Illustration of GenEx Keyphrase Extraction

The keyphrase extraction of GenEx algorithm can be simplified as following three stages, as illustrated in Figure 2.2.

1. Model building: create a model Extractor for identifying keyphrases.
2. Supervised machine learning: using training corpora with author provided keyphrases to train Extractor with Genitor algorithm and make Extractor "learn" the feature of training corpora, so to form GenEx.
3. Extraction: extract keyphrases from testing corpora using GenEx extractor.

2.3.2 Processing procedures of Extractor in GenEx

In GenEx, Extractor algorithm includes ten steps, as illustrated in Figure 2.3 [1].

Figure 2.3 Conceptual Diagram of GenEx Algorithm

- i) **Find single stems:** Make a list of all of the words in the input text. Drop words with less than three characters. Drop stop words using a given stop word list. Convert all remaining words to lower case. Stem the words by truncating them at predefined length of stem characters.
- ii) **Score single stems:** For each unique stem, count how often the stem appears in the text and note when it first appears, then score each stems using predefined parameters.
- iii) **Select top single stems:** Rank the stems in order of decreasing score and make a list of a predefined number of top-ranked single stems.

- iv) **Find stem phrases with different lengths:** Make a list of all the phrases in the input text. A phrase is defined as a sequence of one, two, or three words that appear consecutively.
- v) **Score stem phrases:** For each stem phrase, count how often the stem phrase appears in the text and note when it first appears. Assign a score to each stem phrase, as in step 2.
- vi) **Expand single stems:** For each stem in the list of the top predefined number of single working stems, find the highest scoring stem phrase of one, two, or three stems that contains the given single stem. The result is a list of the number of predefined number of stem phrases. Keep this list ordered by the scores calculated in step 2.
- vii) **Drop duplicates:** The list of the top number of single working stem phrases may contain duplicates. For example, two single stems may expand to the same two-word stem phrase. Delete duplicates from the ranked list of the number of single working stem phrases, preserving the highest ranked phrase.
- viii) **Add suffixes:** For each of the remaining stem phrases, find the most frequent corresponding whole phrase in the input text.
- ix) **Add capitals:** For each of the whole phrases with suffixes added, find the best capitalization.
- x) **Final output:** add suffixes on the ordered phrase list. The list is ranked in the order of the scores calculated in step 2. That is, the score for each whole

phrase is based on the score of the highest scoring single stem that appears in the phrase.

2.3.3 Supervised learning of GenEx algorithm

Supervised machine learning is a mechanism for a system to "learn" something new toward the solution of a problem with given conditions and then the system uses the newly obtained knowledge to get optimized results. For GenEx algorithm, a set of binary strings is used to represent the conditions to obtain optimal extraction results. The set of binary strings works as a set of parameters that reflect the syntactic features of corpora, but their values are not predefined; they are the value ranges. In GenEx, Genitor algorithm works as a trainer to carry out the system learning process. It runs with different parameter values and number of trials. Each trial consists of running Extractor with the parameter settings specified in the given binary string and processing the entire training corpora. Genitor itself updates the parameter value one individual at a time, resulting in continuously changing parameters, with no distinct generations. Finally an optimized set of parameters are found. Extractor uses them to get most matching between author provided keyphrase set and system extracted keyphrase set. [1]

We will give more detailed discussion about supervised machine learning in Chapter 5.

2.4 Summary

We have reviewed the background literatures about keyphrase extraction and surveyed five different keyphrase extraction algorithms that have been practically used. The focus is put on GenEx, a supervised learning algorithm, which will be adapted to pattern-specific extraction application. The reason is lying that it can extract multi-word keyphrases, which meets extraction requirements of pattern articles, and it provides a mechanism for user to adjust the extraction performance based on the corpora to be extracted.

In the next chapter, we will discuss the characteristics of a software design pattern and related issues. Based on the GenEx algorithm, proper design patterns will be selected for Object-Oriented modeling and design of the tool.

Chapter 3 Software Design Pattern and Application in the Design of Pattern-Specific Extraction Tool

This chapter analyzes the specific characteristics of software design patterns, aiming to capture the syntactic characteristics that can distinguish pattern articles from non-pattern articles. Software design patterns in the thesis will work two folds: firstly, they will be used for the adaptation from the wide category, domain independent GenEx algorithm to a pattern-specific extraction algorithm. Secondly, they will be selected for the design and implementation of the pattern-specific extraction tool.

3.1 Structure of Software Design Patterns

Software design patterns aroused a lot of interests in software community due to their characteristics as introduced in the previous sections. They attempt to describe successful solutions to common software problems and have been proven

useful at helping people reuse successful practices. Not only do patterns teach useful techniques, but also they help people communicate better. Design patterns can be viewed as approaches to encapsulate "good practices" in Object-Oriented programming, where the patterns indicate to the programmer how carefully engineered collections of objects can be used to develop applications. In fact, design patterns add another reuse dimension to Object-Oriented programming, since they present a way to repeatedly use the experience of the best developers in structuring applications.

There exist different forms of descriptions for design patterns: Modern Alexander Form, Gang of Four Form, POSA Form and Canonical Form, but the components of each form constitute the very similar framework and the components of each form play very similar functions. A typical form of design pattern consists of the components in terms of NAME, PROBLEM, CONTEXT, FORCE, and SOLUTION [39].

- **Name**

A design pattern must have a meaningful and self-explanatory name. This allows people to use a single word or short phrase to refer to the pattern and the knowledge structure it describes intuitively. It would be very unwise to describe or even summarize the pattern every time people use it in a discussion. A good pattern name forms a vocabulary for discussing conceptual abstractions and facilitates the categorization of pattern collection.

- **Problem**

A statement of the problem which describes its intent, i.e., the goals and objectives it wants to reach within the given context and forces. The statement of a problem should be concise and clear.

- **Context**

The pre-conditions under which the problem and its solution seem to recur, and for which the solution is desirable, tell us the pattern's applicability. It can be thought of as the initial configuration of the system before the pattern is applied to it.

- **Forces**

Forces are interactions/conflicts within related factors that we wish to achieve. They follow a concrete scenario that serves as the motivation for the pattern. Forces reveal the intricacies of a problem and define the kinds of tradeoffs that must be considered in the presence of the tension or dissonance they create.

- **Solution**

It is the approach about how to realize the desired outcome based on the static relationships and dynamic rules described. This is often equivalent to giving instructions that describe how to construct the necessary work products. The description of a solution may encompass pictures, diagrams and prose that identify the pattern's structure, its participants, and their collaborations, in order to show how the problem is solved. The solution should describe not only static structure but also dynamic behavior.

Parallel to these basic concepts, are also two derived concepts:

- **Resulting Context**

The state or configuration of the system after the pattern has been applied, including the consequences (both good and bad) of applying the pattern, and other problems and patterns that may arise from the new context. It describes the post-conditions and side effects of the pattern. This is sometimes called resolution of forces because it describes which forces have been resolved, which ones remain unresolved, and which patterns may now be applicable.

- **Rationale**

It is another justifying explanation of steps or rules in the pattern, and also of the pattern as a whole in terms of how and why it resolves its forces in a particular way to be in alignment with desired goals, principles, and philosophies. It explains how the forces and constraints are played together. This tells us how the pattern actually works, why it works, and why it is "good".

Most of software design patterns have the following general form [40]:

IF you find yourself in CONTEXT
with PROBLEM,
entailing FORCES
THEN for some REASONS,
apply design form and/or rule to construct SOLUTION
leading to NEW CONTEXT and OTHER PATTERNS

The typical form of design patterns makes the pattern articles possess special format. Obviously, these components generate relatively clear cuts between the different writing compartments, which are the major feature of the adaptation of GenEx to the keyphrase extraction of pattern articles.

3.2 Pattern Languages

The other important concept for design patterns worth mentioning is Pattern Languages. The guest editorial of the Communications of the ACM expounded the concepts of pattern languages [41, 3]: pattern languages are not formal languages, but a collection of interrelated patterns. Pattern languages provide vocabulary for talking about particular problems and define the rules to combine them into architectural styles. In other words, pattern languages describe software families of related systems. People can not only use a pattern individually with a specified solution, but can also combine different patterns together to realize new functionality while meeting the requirements of context and forces.

D. C. Schmidt, R. E. Johnson and M. Fayad have given another clearer description about the integration of design patterns to form pattern language [42]. Much of the existing literature on patterns is organized as design pattern catalogs. These catalogs present a collection of relatively independent solutions to common design problems. As more experience is gained using these patterns, developers and authors will increasingly integrate groups of related patterns to form pattern languages. These pattern languages will encompass a family of

related patterns that cover particular domains and disciplines ranging from concurrency, distribution, organizational design, software reuse, real-time systems, business and electronic commerce, and human interface design [42].

The concept of pattern languages is not just limited to those solutions for individually specific problems. It can help for integrated uses of design patterns in complicated problems, which extends the applicability of design patterns.

3.3 Selection of Design Patterns for Development of the Extraction Tool

As the advantages of design patterns in software development, the extraction tool will be design and implemented with design patterns. To select proper design patterns for the system design modeling, we need to deeply investigate the extraction procedures from the viewpoint of software engineering.

3.3.1 Extraction procedures

We first treat the extraction tool as a black box. Using the tool, users extract keyphrases from pattern articles under given constraints, such as the length of keyphrases list, and keyphrase lengths (1-word and/or 2-word and/or 3-word keyphrases). The whole extraction is an incremental process on input data stream and output list as selected strings (keyphrases). Each word and multi-word phrase will go through the similar steps, even though the processing methods are different. The adaptation will not bring about much change on major processing

flow. So proper design patterns can be selected and applied for the data stream process.

3.3.2 Different mechanisms of keyphrase extraction

We can implement the extraction tool as a single component, but such a single component system is too big; it is not feasible. Based on GenEx algorithm described in last chapter, the whole task of the extraction process can be divided into several sequential processing steps, such as data collection, data preprocessing, finding candidate phrases, scoring phrases, and finally determining keyphrases. These decomposed extraction processing steps are connected by the data flow through the system – the output data of one step is the input to the subsequent step. Each processing step can be treated as a filtering component, and whole system can be abstracted as a number of chained filters as shown in Figure 3.1. Each filter carries on part of the processes and manipulation of the data stream.

Figure 3.1 Structure of Filters

Another factor to be considered is that there are three different input files: the text file to be extracted, the stopwords file and the common verbs file. During extraction process, a text file to be extracted first goes through pre-process, in which those stopwords and common verbs listed in the last two files are removed, and then the data flow enters different process paths. All the outputs from

different paths converge and get further processed. This is a system with multiple input and one output. The filtering structure can be depicted in Figure 3.2.

Figure 3.2 Multi-inputs Filter Data Flow

3.3.3 Modeling design and specifications

ISO9126 lists the concrete requirements on software quality, which include functionality, reliability, usability, maintainability and portability. Among them, functionality and reliability are basic requirements; special attention should be paid to other requirements too. Specifically speaking, the extraction tool should possess such features:

- a) Each filtering component could be functionally independent, and can be easily plugged in and removed or reordered,
- b) Filtering component object management functionality should be included,
- c) Each filtering component can be changed or configured dynamically,
- d) Non-adjacent processing steps can share information,
- e) Source code can be shared among similar processes, and

f) No repetitive constructor calling during the execution and redundant coding

We categorize the extraction tool into three functional components:

Data sources: The beginning stage of data originates, which includes the text file to be extracted, the stopwords file and the common verb file.

Data sinks: Final store of the extraction results.

Data filters: The intermediate processing stages. Besides performing the operations of preprocessing and statistical calculations, the filtering components also manipulate and analyze data.

Incremental process of data flow, multi-inputs, single output, branched processes and intermediate information sharing among non-adjacent processes are the main consideration for the selection of proper design patterns to realize the above features.

3.3.4 Selection of patterns for design and implementation

The features of Filter Pattern and Pipes & Filters Pattern for data stream process are listed in Table 3.1. Both patterns can be used to implement sequential process; they do not support the hierarchical combination of filters that are the composition of other filters. Of the two patterns, the more general and simpler form is Filter Pattern. Each processing step can be encapsulated in a filtering component, but does not include functions of Pipes.

We also need the patterns that can support the functions of inheritance, dynamic link and management of the filtering components. Table 3.2 lists three design patterns for the requirements.

Table 3.1 Two design patterns that process data streams

Filter Pattern	
Usage	Feature
Provide a structure for systems that process a stream of data. Each processing step is encapsulated in a filter component.	Does not support the integration and composition of those filters in a hierarchical fashion. Also, it does not have inheritance architecture.
Pipes and Filters Pattern	
Usage	Feature
Provides the function of data stream processing. It allows objects that perform different transformations and computations on streams of data and have compatible interfaces to be dynamically connected in order to perform combinations of operations on streams of data.	It can be used to implement sequential calls of filters. It gives explicit support for sources and sinks and simplifies the sequential combination of filters. However, this design pattern does not support hierarchical combination of filters that are the composition of other filters.

Table 3.2 Some design patterns that have hierarchical structure

Strategy Pattern	
Usage	Feature
Allows the algorithm to be used to vary the context independently and offer a choice of implementations	Each strategy implements a specific part of the behavior of the component. The concrete strategies delegate the executions of a specific task. They can vary the implementations at runtime and adapt it to different needs arising in varying contexts.
Composite Pattern	
Usage	Feature
Defines class hierarchies consisting of leaf objects and composite objects	Allows building complex objects by recursively composing similar objects in a tree-like manner. Allow the objects in the tree to be manipulated in a consistent manner, by requiring all of the objects in the tree to have a common super-class or interface. Composite objects are treated the same way as leaf objects
Composite Filter Pattern	
Usage	Feature
Proper for those applications with various types of filtering functions.	Different types of filters can be hooked into the filter subsystem. A filter controller can integrate the features of the Filter Pattern, Strategy Pattern and Composite Pattern together

3.3.5 Determination of the design patterns to be used

We will use Filter Pattern for the basic data flow process, together with the patterns with filtering component management.

All of the previous three structural patterns in Table 3.2 provide the interfaces of an interchangeable family of algorithms. They hide algorithm implementation details of derived classes with the major classes working as "Containers", and they can use polymorphism to delegate their children.

The Strategy Pattern is a suitable option. Clients of the algorithms couple themselves strictly to the interface. The common implementations and data can be easily shared among similar process, but it needs some modifications in accordance with our extraction requirements. The Composite Pattern is characterized by its tree structure; all of the concrete implementations are constructed in a leaf class. We do not need tree structured filtering system for our design, but its "one to many" relationship between the abstract class and managerial class can be introduced to our design.

The Composite Filter Pattern has mixed features of Filter Pattern, Strategy pattern and Composite Pattern; it provides a common interface to the various filtering mechanisms. It is a "filter-all" pattern and can be used for our extraction tool, but looks like overkill.

During processing of each filtering component, different data formats may be encountered. For example, we use a long string in one filtering stage and use a vector of strings in another filtering stage. So, we add an interface to these

patterns. We combine Filter Pattern and Strategy Pattern together and modify them to form a combined pattern, as shown in Figure 3.3. The Combined Pattern possesses the following features:

- Any filters can be dynamically replaced at run-time and reordered.
- Facilitate the management of parallel filtering components.
- It has high reusability.
- Make the update and reconfiguration easier.

3.4 Summary

The menu style of design pattern articles is characterized by relatively clear separates between the different writing compartments, which is the major feature to be used for the adaptation of GenEx algorithm to pattern-specific keyphrase extraction. The properly selected design patterns for the system design and implementation will improve the software engineering in the development of the pattern-specific extraction tool.

Chapter 4 Important Concepts in the Design of Extraction Tool

Occurrence frequency, Stopwords and Stemming are the important concepts that are commonly used in natural language process. These concepts are used in GenEx algorithm too. This chapter uses a design pattern article as an analytical corpus to discuss these concepts thoroughly and to clearly reveal the functions that they play in keyphrase extraction of pattern articles.

4.1 Phrase Occurring Frequency

Occurrence frequency of phrases in a text reflects the most direct emphasis of a language expression. This is everyone's intuitive experience. The following are the fine resolution of occurrence frequency that describe the weights of phrases in main topic discussion of a text [47]:

- *Term frequency*

It is a within-document frequency of a phrase. The more often a phrase occurs in a document, the more likely it is to be an important one for that document.

Then can we weigh a term proportionally to its term frequency like $(t * f)$, where f is the occurrence frequency and t is some factor that reflects the degree to which occurrence frequency acts on the importance of the term? The answer is NO. A word that appears three times more often than the other does not mean that it is three times more important than that word in the text. Bird's study on Computational Linguistics [48] introduces the weight scales of occurrence frequency on the importance of a phrase in a text. He uses such a relationship $(1 + \log(t * f))$ or $\sqrt{t * f}$ to express the importance of a phrase in a text via occurrence frequency.

- *Collection frequency*

Phrases that occur in only a few documents are often more important than ones that occur in many documents. We can use the contradiction method to prove it. If many articles use the same phrase many times, then it means that the phrase is not special. Bird's also introduces a relationship to weigh the importance of a phrase with occurrence frequency in multi-text: $(1 + \log(t * f)) * \log(N/(d*f))$, where f is the frequency, d is documents, and N is the number of articles [48].

- *Document length*

The third factor that affects the weight of a phrase is the length of a document. If a phrase occurs the same times in a short document and as in a long one, the phrase is likely to be more valuable for the short document.

Obviously, occurrence frequency should not be used just as it stands as the only weighting factor, but must be related to some other source of information about documents. It is confirmed that occurrence frequency of a phrase plays an important role in helping analyze the syntactic function of the phrase [47]. We investigate what phrases with top occurrence frequencies look like in a pattern article.

4.2 Observations through Experimentation

To give a direct view, an extraction experiment on a design pattern article "Pattern density and role modeling of an object transport service" is conducted. The article is from the 4th Pattern Languages of Programming Conference (PLoP) [49]; it will work as an analytical corpus in the thesis. We specially developed a utility program for this experiment.

At first, a pre-process on the given article is made. Remove all the symbols; such as, all non-alphabetic characters, then change them to lower case, and find all phrases with different number of consecutive words, i.e., 1-word, 2-word and 3-word phrases. In this way, we can list all of the potential phrases with different number of words. Some are phrases; some are not; some are important, while others are not important. Then the occurrence frequencies of all phrases are calculated and ranked. Top twenty phrases are selected and listed in Table 4.1, Table 4.2 and Table 4.3, which include the phrases with one, two and three words respectively.

Table 4.1 Occurring frequencies of 1-word phrases

One-word phrase	Frequency	One-word phrase	Frequency
The	160	Framework	28
Of	66	Is	24
And	63	Class	21
A	62	An	20
Role	51	Classes	20
Pattern	46	Which	20
Object	33	As	19
To	33	Patterns	19
Design	28	Roles	19
In	28	Objects	18

Table 4.2 Occurring frequencies of 2-word phrases

Two-word phrase	Frequency	Two-word phrase	Frequency
of the	23	Role model	6
The ots	14	Object transport	6
Role models	13	And the	6
Plays the	11	Design of	6
An object	10	The service	6
The geo-transporter	10	With the	6
The framework	9	The observer	6
The design	8	Instance of	6
Pattern the	7	Of an	5
Pattern applications	7	In a	5

Table 4.3 Occurring frequencies of 3-word phrases

3-word phrase	frequency	3-word phrase	Frequenc y
Pattern the ots	8	Design of the	4
The ots plays	7	of the framework	4
Ots plays the	7	Gamma et al	3
Of an object	5	et al 1995	3
The design of	5	High pattern density	3
A role model	5	The observer pattern	3
Object transport service	4	The objecttransportservice class	3
Based on the	4	The otsclient class	3
As role models	4	one instance of	3
The geo-transporter framework	4	Instance of the	3

Following features are observed:

- a) Not all phrases of different lengths are noun phrases; they include conjunctive, adverbs or adjectives.
- b) For the top twenty phrases with different lengths, fewer-word phrases occur much more frequently than more-word phrases.
- c) Very few verbs appear in different lengths of phrase lists.
- d) Some single/plural forms of the same nouns appear. For example, "object"/"objects", "pattern"/"patterns", "role"/"roles" and "class"/"classes".
- e) Most of the phrases are not content bearing; i.e. can be eliminated safely from the different lengths of phrase lists.
- f) Not just noun phrases are the candidate phrases. Adjective can perfectly constitute a phrase; for example, "high pattern density".
- g) Some special phrases, such as "ots" and "objecttransportservice" need to be investigated from the original text to determine the importance.

The observation in e) involves in the following concept, stopwords.

4.3 Stopwords

The top common words in a text are found with the top occurring frequencies from the text. We noticed that within all phrases in above three tables, the words have different types, i.e., some are pronouns, or prepositions, or conjunctions or adverbs, and some are nouns. Stopwords are those very short and/or common words and include articles; such as, "the", "a", and "an", adverbs; such as, "since",

"then", pronouns, like "this," "that" and "those" and conjunctions; such as, "when" and "how" [50]. Stopwords do not play significant roles in the formation of phrases. Removing these commonly occurring words from text can reduce the number of words to be processed, significantly improve query response time without affecting accuracy. It has been traditional in setting up information systems to discard the commonest words of a language that are less meaningful during indexing and querying. The removal of stop words can greatly assist searching for phrases. In the implementation of our extraction tool, we will eliminate all the stop words in corpus.

Let's still take the sample article "Pattern Density and Role Modeling of an Object Transport Service" as the experiment corpus and remove all the stop words. Then re-calculate the frequency for each phrase. Another three tables Table 4.4, Table 4.5 and Table 4.6 are created on the extraction results. This time, much more content-bearing phrases are listed out. It shows the following features:

- a) There are more meaningful phrases.

Table 4.4 Occurring frequencies of 1-word phrases after removing stopwords

One-word phrase	Frequency	One-word phrase	Frequency
Role	51	Different	14
Pattern	46	Ots	14
Object	33	Models	13
Design	28	Service	12
Framework	28	Using	12
Class	21	Plays	12
Classes	20	Communicator	11
Patterns	19	Based	10
Roles	19	Geo-transporter	10
Objects	18	Observer	10

Table 4.5 Occurring frequencies of 2-word phrases after removing stopwords

Two-word phrase	Frequency	Two-word phrase	Frequency
Role models	13	Observer pattern	5
Pattern ots	8	Riehle d	5
Pattern applications	7	Transport service	4
Role model	7	Using role	4
Ots plays	7	Role-model based	4
Pattern density	6	Geo-transporter framework	4
Role modeling	6	Objecttransportser-vice class	4
Object transport	6	Object-oriented frameworks	3
Design patterns	5	Design pattern	3
Object collaboration	5	Like object	3

Table 4.6 Occurring frequencies of 3-word phrases after removing stopwords

3-word phrase	Frequency	3-word phrase	Frequency
Pattern ots plays	7	Remote request execution	2
Object transport service	4	Designed implemented object-oriented	2
High pattern density	3	Implemented object-oriented framework	2
Ots plays client	3	Patterns role models	2
Object-oriented frameworks exhibit	2	Using role models	2
Design pattern applications	2	Role models design	2
Services like object	2	Particular aspect object	2
Like object migration	2	May play roles	2
Object migration remote	2	Configures service instantiating	2
Migration remote request	2	Service instantiating appropriate	2

- b) There are still some words in the phrases with both singular and plural forms of the same nouns being listed. They are "object"/"objects", "pattern"/"patterns", "role"/"roles", "class"/"classes" and "Role model"/"Role models". If we fold them to singular form, we can get even more unique meaningful phrases.

- c) Some adjectives, verbs, past participle, present participle appear in the phrases. Verbs, even very few, are not useful for the phrases, so they can be eliminated. The sequence of words consisting of a string of nouns and adjectives with a final noun or gerund could be candidate phrases. "high pattern density", for example.

Special stop words also needed to be considered; such as, the irregular form of strings something like this,

"" , /* null string */

"am/is/are/be/being/been/" , /* BE */

"have/has/having/had/" , /* HAD */

"do/does/doing/did/" , /* DID */

..., /* multi-line string */

These phrases are mapped to the null strings.

The stopwords list contains 474 words in different syntactic type [50]. It consists of most of the stopwords that occurred in pattern articles.

4.4 Stemming

In English, there are many words derived from the same roots with the addition of affixes, for example, model/modeling and frame/framework. Usually, the root of a word is the most content bearing. If more words with same root appear in a text, it means that the text talks more about what is reflected from the root. The other affixes are used to change the type of words or generate close-meaning new words.

Stemming is a typical language processing method [1]. It has been used to reduce variant word forms to common roots in information processing system and thereby improve the ability of the system to match query and document vocabulary. Based on English linguistics, the variety in word forms comes from both inflectional and derivational morphology. They can be used to distinguish between inflectional morphology like plurals, and derivational morphology that can change a word's syntactic category [51].

Usually the approach for stemming is to drop suffixes. The two most popular algorithms for stemming English words are given by Porter [52] and Lovins [53]. It is found that aggressive stemming is better for keyphrase extraction than conservative stemming. Turney found that the iterated Lovins stemming algorithm is better for keyphrase extraction than conservative stemming [1]. The algorithm repeatedly applies the Lovins stemmer, until the word stops changing. Iterating in this manner will necessarily increase (or leave unchanged) the aggressiveness of any stemmer. But the Iterated Lovins stemming algorithm has limitations; for example, it could not solve the duplication of “probability” and “probabilities” [1].

The most common approach to this problem is to identify more elaborate patterns over character sequences that reliably separate tokens so as to get their root forms [54]. A broad range of such patterns can be defined in terms of a context-sensitive transformation grammar. For example, we specially process those words ending with "IES".

Some other specific cases also should be considered, for example, the special ending of nouns. Even in the simple case of plural endings, it is not as simple as removing S's. We cannot assume that every time there is an ending "S" we can remove it. Consider the following words, stemming *PROCESS* and *ADDRESS* to *PROCES* and *ADDRES* would make no sense. In our extraction tool, we also have treatment on the words ending with "SSES" and "SS". Still, those cases including the specific forms of plural nouns, such as *DATUM/DATA* CORPUS/ CORPORA and *LEAF/LEAVES* need separate treatments.

We also need to distinguish words like "hypothesis". For the process on different cases, we follow such rules:

- If a word has both verb and noun functions, treat it as a noun.
- For the case, such as "model's", we will process it after matching
- Adjust the word pre-processing order; such as, removing verbs first
- Some words are still hard to process, such as "appropriate", which can be adjective or verb, keep them.

Usually, an author never suggests keyphrases that map to the same stemmed phrase, so an extractor should be so designed that it never suggest keyphrases to be mapped to the same stemmed phrase. When calculating the number of matches, it is allowed at most one extractor extracted phrase to match an author-provided phrase. Thus, if the extractor does suggest two keyphrases that map to the same stemmed phrase (contrary to our preceding assumption), then at most one of these two keyphrases can count as a match.

After using the stemming method to process the text, those words with the same roots are pulled to a more outstanding position and more likely chosen as the candidate keyphrases.

4.5 Summary

The pattern specific extraction tool to be developed is based on GenEx algorithm. This chapter discusses the three important concepts that are used in general processes of GenEx, namely calculation of occurrence frequency, removal of stopwords and stemming. In our adapted version of GenEx, these concepts are also used in the process of a text to be extracted.

Chapter 5

Adaptation of GenEx to Pattern-Specific Extraction

Chapter 2 introduced GenEx extraction algorithm. Chapter 3 reviewed the features of design patterns. This chapter studies how to get the characteristics of pattern articles to be reflected in the extraction mechanism of GenEx algorithm, so as to derive an pattern-specific, adapted version of GenEx algorithm and obtain a higher extraction performance.

5.1 Process of Supervised Learning of GenEx Algorithm

GenEx is a hybrid algorithm [61], as the discussion in Chapter 2. Its central component is Extractor, together with a genetic Genitor algorithm. GenEx provides a mechanism and 12 parameters for Extractor to "learn" the characteristics of training corpora by means of Genitor algorithm. The parameters are not a set of predefined values, but a set of value ranges, which reflect the syntactic features of keyphrase locations in training corpora. Genitor carries on

the value settings and adjustments of parameters through supervised machine learning.

Genitor algorithm may be viewed as a method for optimizing a string of bits. Total 72 bits are used to represent ten of the twelve parameters (another two parameters are set manually). Genitor is run with a given population size for number of trials. Each trial consists of running Extractor with the parameter setting specified in the given binary string and processing the entire training set. The performance measure for the binary string is based on the average precision for the whole training set. The final output of Genitor is the highest scoring binary string, which can be used to extract the maximum matches between the output of Extractor and the target keyphrase lists. The performance of the learning process is measured by examining the match using the testing data [1].

5.2 Parameters of the Extractor

The descriptions of the 12 parameters of GenEx algorithm are list in Table 5.1:

Table 5.1 Parameter definition of Extractor

Name of Parameter	Description
NUM_PHRASES	Length of final keyphrase list
NUM_WORKING	Length of working list.
FACTOR_TWO_ONE	Factor for expanding to two words
FACTOR_THREE_ONE	Factor for expanding to three words
MIN_LENGTH_LOW_RANK	Low rank words must be longer than this
MIN_RANK_LOW_LENGTH	Short words must rank higher than this
FIRST_LOW_THRESH	Definition of "early" occurrence
FIRST_HIGH_THRESH	Definition of "late" occurrence
FIRST_LOW_FACTOR	Reward for "early" occurrence
FIRST_HIGH_FACTOR	Penalty for "late" occurrence
STEM_LENGTH	Maximum characters for fixed length stem
SUPRESS_PROPER	Flag for suppressing proper nouns

To get more information about the parameters, reader can refer to paper about keyphrase extraction with GenEx algorithm [1]. In GenEx algorithm, the types and sample ranges of the parameters are listed are as Table 5.2 [1].

Table 5.2 Parameters of Extractor

Parameter Number	Parameter name	Parameter type	Parameter range
3	FACTOR_TWO_ONE	Real	[1, 3]
4	FACTOR_THREE_ONE	Real	[1, 5]
5	MIN_LENGTH_LOW_RANK	Real	[0.3, 3.0]
6	MIN_RANK_LOW_LENGTH	Integer	[1, 20]
7	FIRST_LOW_THRESH	Integer	[1, 15]
8	FIRST_HIGH_THRESH	Integer	[0.01, 1.0]
9	FIRST_LOW_FACTOR	Real	[1, 15]
10	FIRST_HIGH_FACTOR	Real	[0.01, 1.0]
11	STEM_LENGTH	Integer	[1, 10]
12	SUPRESS_PROPER	Boolean	[0, 1]

5.3 Adaptation of GenEx to Pattern Specific Extraction

5.3.1 Observation of keyphrase extraction on pattern articles with GenEx

All pattern articles generally follow basically the same format, as we discussed in Chapter 3. Each pattern article declares a problem with the given context and related issues, then describes a solution. The writing compartments of pattern articles are clearly separated.

To find the more detailed characteristics on pattern articles so as to find the approach for adaptation, extraction experiments are carried out with GenEx demo program on training corpora, which includes a group of 20 pattern articles. The following phenomena has been found:

- i) A typical pattern article contains five components, namely NAME, PROBLEM, CONTEXT, FORCE and SOLUTION. It is found that the distributions of keyphrases in design pattern articles are most likely located in the components of PROBLEM, FORCE and SOLUTION. It means that the locations of keyphrases in pattern articles are most likely associated with these writing compartments.
- ii) The extraction results from the group of training corpora with GenEx online demo program shows that on average, the extraction rate for 2-word and 3-word keyphrases is lower than that for the keyphrase lists provided by authors, comparing 27.3% and 5.0% to 34.1% and 14.0% respectively.
- iii) The average extraction rate difference between 1-word and 2-word keyphrases specified by authors are relatively smaller than that extracted with GenEx extractor, comparing with 51.9% to 34.1% for author's results and 67.7% to 27.3% for GenEx' results.

Table 5.3 and Table 5.4 list the statistical average percentages of 1-word, 2-word and 3-word keyphrases extracted with GenEx extractor and with the author provided keyphrase lists, based on the same training corpora.

Table 5.3 Distributions of extracted 1-, 2- and 3-word keyphrase - GenEx extractor

File No.	Total Keyphrases	1-word	2-word	3-word
101	7	6	1	0
102	7	4	3	0
103	7	6	1	0
104	7	6	1	0
105	7	6	1	0
106	7	5	2	0

107	7	4	2	1
108	6	5	1	0
109	7	4	2	1
110	7	6	1	0
111	7	2	4	1
112	7	4	3	0
113	7	4	2	1
114	7	2	3	2
115	7	5	2	0
116	7	5	2	0
117	7	4	3	0
118	7	5	1	1
119	7	7	0	0
120	7	4	3	0
Total	139	94	38	7
Percentage	100%	67.7%	27.3%	5.0%

Table 5.4 Distributions of 1-, 2- and 3-word keyphrase - author provided

File No.	Total Keyphrases	1-word	2-word	3-word
101	11	3	7	1
102	4	3	1	0
103	5	2	1	2
104	6	4	1	1
105	3	1	2	0
106	4	2	2	0
107	4	3	1	0
108	2	2	0	0
109	3	0	2	1
110	4	3	1	0
111	3	3	0	0
112	3	0	1	2
113	3	3	0	0
114	3	1	1	1
115	3	1	2	0
116	5	4	1	0
117	3	3	0	0
118	4	0	2	2
119	3	3	0	0
120	3	0	2	1
Total	79	41	27	11
Percentage	100%	51.9%	34.1%	14.0%

iv) Among the 12 parameters of GenEx algorithm, two sets of parameters of FIRST_LOW_THRESH, FIRST_LOW_FACTOR and FIRST_HIGH_THRESH, FIRST_HIGH_FACTOR play more functions in determining the "location" of keywords/keyphrase (Table 5.2). Compared with other parameters, they can provide more adjustment space based on their "early" or "late" occurrence than other parameters.

5.3.2 Adaptation with parameters

Based on the characteristics of design pattern articles, described in 3.1 and the above observations, we make following changes on the parameters:

- **NUM_PHRASES**

The aim to extract keyphrases is to provide more clues to readers under the main titles of the articles and to help index pattern articles. If there are too many keyphrases, they appear verbose and if too few, they show less reference. Unless an article is talking about a brand new concept or process, it needs enough keyphrases for the reader to distinguish one article from the others that discuss the same topic, even though the number of keyphrases is not absolutely strict.

It is found from the available pattern articles that the number of author-provided keywords/ keyphrases is between 3 and 11. We select 10.

- **NUM_WORKING**

The GenEx algorithm sets the working number of phrases as 50 to decrease the process of candidate phrases. In our implementation, we modify the algorithm by folding those plural form words to singular form, which decrease large number of redundant words. So, we set NUM_ WORKING to be 10. In fact, the experiments on extractions of pattern articles show that there is not much change on extraction results with the different settings of 10, 20 and 50, but the process speed is much faster with setting of 10.

- FACTOR_TWO_ONE and FACTOR_THREE_ONE

In GenEx, the two parameters are introduced to increase the weight scales of 2-word and 3-word phrases to increase the likelihood of 2-word and 3-word phrases to be selected as a keyphrase. Based on the distributions of 2- and 3-word keyphrases on pattern articles listed in Table 5.3 and Table 5.4 – 34.1% and 14.0% provided by authors, and 27.3% and 5.0% extracted with GenEx. We increase the values of two settings further so as to adapt to the characteristics.

- STEM_LENGTH

It is defined as the number of characters for fixed length stem used for stemming each word in the phrase, so as to improve the ability of the system to match a query. Adjusting STEM_LENGTH can tune the aggressiveness of the stemming. When STEM_LENGTH is low, stemming by truncation is more aggressive, i.e., more likely to map two words to the same stem [1]. In GenEx, the sample setting value is 5; we found that a setting of 4 for the parameter can give better mapping.

- To adapt the above point i) and iv), we introduce 6 new parameters to replace the original 4 parameters: FIRST_LOW_THRESH, FIRST_HIGH_THRESH, FIRST_LOW_FACTOR and FIRST_HIGH_FACTOR. The six new parameters are PROBLEM_THRESH, FORCE_THRESH and SOLUTION_THRESH, associated with the respective weight scale parameters namely PROBLEM_FACTOR, FORCE_FACTOR and SOLUTION_FACTOR.
- We also found that the use of the parameters of MIN_LENGTH_LOW_RANK and MIN_RANK_LOW_LENGTH even lower the extraction performance for pattern articles, so we removed the two parameters and related process.

The adapted parameters we used are described in Table 5.5.

Table 5.5 Parameter definition of adapted algorithm

Parameter Name	Description
NUM_PHRASES	Length of final keyphrase list
NUM_WORKING	Length of working list.
FACTOR_TWO_ONE	Factor for expanding to two words
FACTOR_THREE_ONE	Factor for expanding to three words
PROBLEM_THRESH	Definition of "problem" occurrence
FORCE_THRESH	Definition of "force" occurrence
SOLUTION_THRESH	Definition of "solution" occurrence
PROBLEM_FACTOR	Reward for "problem" occurrence
FORCE_FACTOR	Reward for "force" occurrence
SOLUTION_FACTOR	Penalty for "solution" occurrence
STEM_LENGTH	Maximum characters for fixed length stem
SUPRESS_PROPER	Flag for suppressing proper nouns

Table 5.6 Parameter type and value ranges of adapted algorithm

Parameter Number	Parameter name	Parameter type	Parameter range
3	FACTOR_TWO_ONE	Real	[2, 4]
4	FACTOR_THREE_ONE	Real	[3, 6]
5	PROBLEM_THRESH	Real	[5, 10]
6	FORCE_THRESH	Integer	[15, 20]
7	SOLUTION_THRESH	Integer	[1.0, 2.0]
8	PROBLEM_FACTOR	Integer	[1, 2.0]

9	FORCE_FACTOR	Real	[0.50, 1.0]
10	SOLUTION_FACTOR	Real	[0.80, 1.0]
11	STEM_LENGTH	Integer	4
12	SUPRESS_PROPER	Boolean	0

The ten new parameters of Extractor are turned manually with the listed types and value ranges, which are listed in Table 5.6.

5.3.3 *Adaptation with other factors*

- Our extraction tool also considers those sequence of words consisting of a string of adjective-noun-noun and adjective-noun-gerund as candidate phrases and consider whether they are keyphrases. For example, "probabilistic machine learning" contains an adjective followed by a noun and then a gerund.
- Plural form of nouns in articles may cause redundant keyphrase selection. To avoid it, we fold plural forms of words to their original forms, which increases the effective selection of keyphrases.

5.3.4 *From supervised machine learning to manual tuning*

GenEx algorithm uses the supervised machine learning mechanism to get the optimal parameter values with Genitor algorithm, so as to obtain the maximum number of matches between machine extracted keyphrases and author provided keyphrases. We did not carry out the "machine supervised learning". Instead, after completely analyzing both characteristics of GenEx algorithm and pattern articles, we found the "optimal" values of the parameters by means of repetitive manual tuning and analyzing the number of matches and F-measures between

extracted results and author provided keyphrases. As soon as the "highest" extraction rates are obtained with the manual tuning through training corpora, the parameters are fixed. Then the optimized extraction tool is used on testing corpora, so as to get a practical extraction evaluation. Here the words "optimal" and "highest" are based on the manual tuning, because it is not possible to do the extraction on training corpora with such the fine resolution of the parameters one-by-one as Genitor does.

5.4 Summary

The adaptations of GenEx to pattern-specific extraction are:

- Replace some original parameters with ones reflecting the pattern-specific features.
- Remove original less-valued parameters.
- Fold those plural forms of words to their original form.
- Decrease the parameter value of NUMBER_WORKING.
- Select smaller STEM_LENGTH for more words mapping to the same roots.
- Tune the parameters through repetitive extractions with the tool and comparisons of the extracted results with author provided keyphrase lists.

Chapter 6 Evaluation of Performance of the Extraction Tool

This chapter discusses the selection of evaluation corpora and sets the evaluation scheme to examine the performance of extraction tool. The evaluation is carried out with three different approaches. Firstly, average extraction rate is used to give intuitive view on overall performance; secondly, three common measures, i.e., Precision, Recall and F-measure are used to make quantitative analysis. Finally, the F-measures relative to the sizes of the extracted pattern articles are analyzed.

6.1 Selection of Extractors and Evaluation Corpora

To carry out the evaluation, we need to determine the performance references and the testing corpora. The performance references come from the extraction results with other extractors on the same testing corpora.

6.1.1 Three extractors joining the evaluations

Chapter 2 has introduced the 5 extraction algorithms. Among them only GenEx online extraction demo program and Microsoft Word97 available for practical extraction. They are to be used, together with the developed pattern-specific extraction tool, for the extraction evaluations on the same evaluation corpora.

6.1.2 Selection of evaluation corpora

In linguistics, a corpus may be defined as "a collection of texts assumed to be representative of a given language, dialect, or other subset of a language, to be used for linguistic analysis" [56]. The advantage of using a corpus for research is to provide ready-to-use examples of word usage that can be put in a dictionary entry or cited in a study of a particular word.

There is not a ready-to-use design pattern corpus collection. We searched design pattern home pages and collected a group of 250 pattern articles for the corpora data pool. Among them, keyphrases specified by authors only appear in 32 articles. Since we have to use author provided keyphrase lists as the standard keyphrase lists for the evaluation reference, only this type of pattern articles is our concern. 20 of the 32 articles randomly are selected as the training corpora and used to tune the extraction tool.

Twenty pattern articles as the training corpora seems not enough, but the authors of Kea, the another keyphrase extractor, give their statement based on the experiments [37]. In their discussion on the selection of the number of training corpora sets, they varied the size of the training set from 1 to 130 documents and

tested Kea's performance with each set. The results show that the extraction performance improves steadily up to a training set of about 20 documents and very small gains are made until the training set holds 50 documents. As that described in Chapter 2, the algorithm of Kea is mainly based on the GenEx algorithm [1]. So, it is considered that the conclusion of Kea to be suitable for the selection of training corpora for our developed extraction tool. Figure 6.1 illustrates the results of the extraction performance of the Kea experiment against the number of training corpora [37].

Figure 6.1 Performance against the number of training documents

6.1.3 Evaluation scheme

The evaluation scheme is divided into three parts:

- Average extraction rate will be first used to examine the overall effectiveness of the extraction tool. This is an intuitive method that only needs to directly count the number of matched keyphrases and compare it with total number of

author provided keyphrases. Using this value, users can directly tell the degree of their satisfaction with the extraction tool.

- Precision, Recall and F-measure are to be use to examine the extraction performance, so as to compensate the difference of keyphrase number (NUM_PHRASES) in the final keyphrase lists of the 3 different extractors.
- The extraction effects relative to different sizes of corpora are investigated.

6.2 Results of Evaluation of Three Extractors

The extraction results with developed pattern-specific extraction tool, GenEx online extraction demo program and Microsoft Word97 on both training corpora and testing corpora are presented here.

6.2.1 *Average extraction rates*

The average extraction rates are based on the training corpora with 20 pattern articles, and testing corpora with 12 pattern articles. For training corpora, the total number of 79 author provided keyphrases work as the standard keyphrase reference; the total number of keyphrases extracted with the developed pattern-specific extraction tool, GenEx extractor and with Microsoft Word97 are 31, 25 and 16 respectively.

The average extraction rates are as follows:

- Average extraction rate with our extraction tool = $31 / 79 = 0.393$
- Average extraction rate with GenEx online demo program =

$$25 / 79 = 0.316$$

- Average extraction rate with Word97 = $16 / 79 = 0.203$

For testing corpora, the numbers of the keyphrases extracted with the three extractors are 14, 9 and 8 and corresponding to the average extraction rates 0.359, 0.231 and 0.205 respectively, based on the author provided total 39 keyphrases in this group.

Figure 6.2 illustrates the average extraction rates using three different keyphrase extractors on both training and testing corpora.

Figure 6.2 Average extraction rates for 3 extractors on 2 types of corpora

6.2.2 Three measures

If we only take the total number of extracted keyphrases into account, the performance of our extraction tool is obvious. But we noticed that each extractor's settings for the keyphrase number is different from the others: the number of keyphrase setting is 10 with the developed extraction tool, 7 with GenEx

extractor, and 5 with Microsoft Word97. So, we use commonly used three measures, i.e., Precision, Recall and F-measures to give another view for the evaluation, which is the second part of the evaluation scheme.

Table 6.1, 6.2 and 6.3 list the results of the Precision, Recall and F-measure with our 3 extractors on the same training corpora. The formulas to calculate the 3 measures have been described in Chapter 2.

Table 6.1 Measures of extraction on training corpora with our extraction tool

Average Precision:
$(0.600 + 0.400 + 0.200 + 0.100 + 0.200 + 0.100 + 0.100 + 0.100 + 0.100 + 0.100 + 0 + 0 + 0.200 + 0.100 + 0.200 + 0.200 + 0.100 + 0 + 0 + 0.200) / 20 = 0.150$
Average Recall:
$(0.575 + 1.000 + 0.400 + 0.167 + 0.667 + 0.250 + 0.250 + 0.500 + 0.333 + 0.250 + 0 + 0 + 0.667 + 0.333 + 0.667 + 0.400 + 0.333 + 0 + 0 + 0.667) / 20 = 7.209 / 20 = 0.361$
Average F-measure:
$(0.571 + 0.571 + 0.267 + 0.125 + 0.308 + 0.143 + 0.143 + 0.167 + 0.154 + 0.143 + 0 + 0 + 0.308 + 0.154 + 0.308 + 0.267 + 0.154 + 0.308 + 0 + 0) / 20 = 4.091 / 20 = 0.205$

Table 6.2 Measures of extraction on training corpora with GenEx demo program

Average Precision:
$(0.428 + 0.428 + 0.288 + 0.143 + 0.143 + 0.288 + 0.143 + 0.143 + 0.143 + 0.143 + 0 + 0 + 0 + 0.143 + 0.288 + 0.288 + 0 + 0.143 + 0 + 0) / 20 = 3.152 / 20 = 0.158$
Average Recall:
$(0.273 + 0.750 + 0.400 + 0.167 + 0.333 + 0.500 + 0.250 + 0.500 + 0.333 + 0.250 + 0 + 0 + 0 + 0.333 + 0.667 + 0.333 + 0.400 + 0.250 + 0 + 0) / 20 = 5.739 / 20 = 0.287$
Average F-measure:
$(0.333 + 0.545 + 0.333 + 0.154 + 0.200 + 0.364 + 0.182 + 0.222 + 0.200 + 0.182 + 0 + 0 + 0 + 0.222 + 0.400 + 0.333 + 0 + 0.182 + 0 + 0) / 20 = 3.852 / 20 = 0.193$

Table 6.3 Measures of extraction on training corpora with Word97

Average Precision:
$(0.600 + 0.400 + 0 + 0.400 + 0.200 + 0 + 0 + 0.200 + 0 + 0.200 + 0 + 0 + 0.200 + 0.200 + 0.200 + 0.400 + 0 + 0 + 0 + 0) / 20 = 0.160$
Average Recall:
$(0.273 + 0.500 + 0 + 0.333 + 0.333 + 0 + 0 + 0.500 + 0 + 0.250 + 0 + 0 + 0.333 + 0.333 + 0.333 + 0.200 + 0.667 + 0 + 0 + 0 + 0) / 20 = 4.055 / 20 = 0.203$
Average F-measure:
$(0.375 + 0.444 + 0 + 0.364 + 0.250 + 0 + 0 + 0.222 + 0 + 0.222 + 0 + 0 + 0.250 + 0.250 + 0.250 + 0.200 + 0.500 + 0 + 0 + 0 + 0) / 20 = 3.327 / 20 = 0.166$

Average extraction rate and 3 measures will also be used to evaluate the extraction effects on testing corpora with author provided keyphrases after tuning.

Table 6.4 Measures of extraction on testing corpora with our extraction tool

Average Precision:
$(0.100 + 0.100 + 0.100 + 0 + 0.100 + 0 + 0.100 + 0.300 + 0 + 0.200 + 0.200 + 0.200) / 12 = 0.117$
Average Recall:
$(0.333 + 0.333 + 0.333 + 0 + 0.333 + 0 + 0.250 + 0.500 + 0 + 0.667 + 0.400 + 0.667) / 12 = 0.318$
Average F-measure:
$(0.154 + 0.154 + 0.154 + 0 + 0.154 + 0 + 0.154 + 0.375 + 0 + 0.308 + 0.267 + 0.308) / 12 = 0.269 + 0.375 + 0.2667 + 0.615) / 12 = 0.169$

Table 6.5 Measures of extraction on testing corpora with GenEx demo program

Average Precision:
$(0 + 0 + 0.143 + 0.143 + 0.143 + 0.288 + 0.143 + 0.143 + 0 + 0.143 + 0 + 0.143) / 12 = 3.152 / 12 = 0.107$
Average Recall:
$(0 + 0 + 0.333 + 0.500 + 0.333 + 0.667 + 0.333 + 0.167 + 0 + 0.333 + 0 + 0.333) / 12 = 0.250$
Average F-measure:
$(0 + 0 + 0.200 + 0.222 + 0.200 + 0.400 + 0.200 + 0.154 + 0 + 0.200 + 0 + 0.200) / 12 = 0.148$

Table 6.6 Measures of extraction on testing corpora with Word97

Average Precision:
$(0 + 0 + 0 + 0 + 0.200 + 0 + 0 + 0.600 + 0 + 0.200 + 0.400 + 0.200) / 12 = 8 / 12 = 0.133$
Average Recall:
$(0 + 0 + 0 + 0 + 0.333 + 0 + 0 + 0.500 + 0 + 0.333 + 0.400 + 0.333) / 12 = 0.158$
Average F-measure:
$(0 + 0 + 0 + 0 + 0.125 + 0 + 0 + 0.545 + 0 + 0.250 + 0.400 + 0.25) / 12 = 0.131$

Table 6.4, 6.5 and 6.6 list the results of the Precision, Recall and F-measure with three extractors on the testing corpora. Average extraction rate and the three

measures will be used to compare the extraction effects on testing corpora with the extraction results from other two extractors, GenEx online demo program and Word 97.

Figure 6.3 illustrates the F-measure values with 3 different extractors, on training corpora and testing corpora.

Figure 6.3 Average F-measures for 3 extractors on 2 types of corpora

Appendix 1 lists the extracted keyphrases from both training corpora and testing corpora, with our developed extraction tool, GenEx extractor and Word97.

6.2.3 *Extraction results with different sizes of articles*

We now check the influence of pattern article lengths on F-measures, following the third evaluation scheme. Before starting it, we first arrange all of the pattern articles of both training and testing corpora based on their sizes, which are listed in Appendix 3. The sizes of the 32 pattern articles range from 7k to 63k on the horizontal axis. The corresponding F-measure performance of each extraction

result is shown on the vertical axis. Figure 6.4 depicts the F-measures corresponding to each pattern article in the order of sizes.

Figure 6.4 F-measures with size-incremented corpora

6.3 Summary

We discussed the selection of the number of training corpora. Only 32 pattern articles with author provided keyphrases are found. Among them, 20 of the 32 articles are randomly selected as the training corpora and the remaining for testing corpora. For the evaluation of our extraction tool on testing corpora (remaining 12 articles), it is carried out after the completion of tuning on testing corpora. It includes the comparisons between the extraction results with our developed extraction tool and the standard keyphrase sets provided by authors and the parallel comparisons between the extraction results with our developed extraction tool and the extraction results with the other two extractors. In Chapter 7, we will discuss the extraction performance.

Chapter 7 Performance Analysis

Chapter 6 records all the extraction results with the three different extractors on both training corpora and testing corpora, including average extraction rates, 3-measures (Precision, Recall and F-measure) and the depicted relation of F-measures corresponding to size-ordered pattern articles. Following are the discussion on the significance of each measurement.

7.1 Discussion on average extraction rates

Average extraction rate, as an intuitive measure, reflects individual extraction performance. In the extractions on both training corpora and testing corpora, the highest average extraction rate are obtained with our developed extraction tool. As the discussion in Chapter 6, average extraction rate can not only be used for the comparison of the performance among the different extractors, since the size settings of the number of keyphrases for each extractor are different.

7.2 Discussion on Precision, Recall and F-measure

As introduced in Chapter 2, Precision measures are the ratio of the correctly extracted information against all the extracted information. Recall measure is the ratio of the correct information extracted from the texts against all the available information. *F*-measure is the harmonic mean of Precision and Recall and is used when a single evaluation value is required.

Based on the extraction data in Chapter 6, we rank the extraction performance of the three different extractors on the training corpora as the follows:

Precision: Word 97, GenEx Demo Program, our extraction tool

Recall: Our extraction tool, GenEx Demo Program, Word 97

F-measure: Our extraction tool, GenEx Demo Program, Word 97

Same performance is obtained with the testing corpora, except that the rank of Precision with our developed extraction tool is promoted, from third rank to second rank. The detailed data is shown in Table 7.1.

Table 7.1 F-measure of 3 different extraction algorithms on both corpora

Extractor \ 3-measure	Our Extraction Tool		GenEx on-line Extractor		Microsoft Word97	
	Training	Testing	Training	Testing	Training	Testing
Precision	0.150	0.117	0.158	0.107	0.160	0.133
Recall	0.361	0.318	0.287	0.249	0.203	0.158
F-measure	0.205	0.169	0.193	0.148	0.166	0.131

7.2.1 F-measure

Among the three measures, F-measure provides subject evaluations from both point of views of authors and extractors, which is discussed in Chapter 2. It is a

most valuable measure that reflects extraction performance. Table 7.1 shows that our developed extraction tool obtained the highest F-measure values, either on training corpora, or on testing corpora.

7.2.2 Recall

Recall of the extraction performance reflects the degree of satisfaction of authors. It is the ratio of the number of keyphrases agreed by both author and extractor, and the number of phrases identified as keyphrases by the author. The data in Table 7.1 also shows that our developed extraction tool obtained the highest Recall value, which means that the author has highest degree of satisfaction to the extraction result, compared with the results provided by another two extractors.

7.2.3 Precision

It is noticed that the Precision with our extraction tool is lower than that with other two extractors on training corpora. The highest setting of the number of keyphrase with 10 among the three extractors may contribute to the result. Since Precision reflects the degree of satisfaction of an extractor and it is the ratio of the number of keyphrases agreed by both author and extractor against all the extracted information by an extractor. The bigger the setting of the number of keyphrase, the more the extracted information, so as the lower the Precision.

7.2.4 Difference of 3-measure values with training corpora and testing corpora

We also observed another data changes: all the 3-measure values with three different extractors on the testing corpora are lower than that extracted from the

training corpora. We think that the difference of the number of evaluated pattern articles between the training corpora and the testing corpora may contribute to the cause. The testing corpora of 12 articles are not enough to make a comparable statistical analysis with the training corpora of 20 articles. Due to the limitation of the pattern articles with author provided keyphrases, we could not carry out further analysis on this point.

The different distribution of keyphrases (such as the relative number of 1-, 2- and 3-word keyphrase and the occurrence degree of author provided keyphrases practically appearing in the pattern articles) may cause such the changes too. We introduce 2 new concepts (zero-measure-value and common-zero-measure-value) to give more detailed discussion.

7.3 Zero Measure Values and the Effect on Extraction

During the extraction with three different extractors on two types of corpora, special cases were found on some extractions where keyphrases extracted with any one of the three extractors do not match any of the keyphrases provided by the authors. The instances are displayed in bold font in Table 6.1, 6.2 and 6.3 for training corpora and in Table 6.4, 6.5 and 6.6 for testing corpora. As a result, each of the three calculated measures of Precision, Recall and F-measure are all zeros. This phenomenon has not been described in literature, to our knowledge. Now we like to give the case a name "common-zero-measure-value" to describe the common non-matching with any extractors. We also use "zero-measure-value" to describe the non-matching of a specific extractor. They are very significant, we think, when

making comparison and discussion of the extraction performances with different extractors on the same corpora. Based on the Table 6.1 to Table 6.6, with 3 different extractors, Table 7.2 is used to summarize the number of such the cases from training corpora and testing corpora.

Table 7.2 Zero-measure-values of the extractors with both corpora

Extractor Corpora type	Our Extraction tool	GenEx Extractor	MS Word 97
Training corpora	4	5	9
Testing corpora	3	4	7

The calculated percentile of total zero-measure-values against the total corpora size for each extractor is illustrated in Figure 7.1.

Figure 7.1 Average of zero-measure-values

There are two aspects to the occurrence of zero-measure-values. Firstly, the authors provided keyphrases do not show up in the original articles and secondly, the keyphrases both listed and included in the original articles are not extracted. Checking the original articles with the common-zero-measure-value, we found the

cause: not all keyphrases provided by the authors are showing up in the original articles. This confirmed the statement, made by the authors of Kea algorithm, that only about 80% of keyphrases specified by authors actually occur in the original article [59].

From Figure 7.1, It can be seen that the zero-measure-values in the extraction of testing corpora are all higher. In fact, among all the factors that influence F-measure, zero-measure-value causes largest impact. So, larger average number of zero-measure-value in testing corpora is the main cause of the lower average F-measures. Among the results from three extractors, our extractor obtained the lowest zero-measure-value, which reflects that the adaptation in our extractor makes the contribution to the decrease of zero-measure-values, from the other point of view.

In contrast, Word 97 contains most zero-measure-values. The reason is that it can only extract one-word keyphrases (or called keywords), which provide very small proportion of information in pattern articles, relative to that specified by the authors. As a result, it appears insufficient as representatives of the main topics.

7.4 Effect of the Article Size on Performance

Figure 6.3 depicts the F-measures corresponding to each pattern article in a size-incremented order. The total number of ordered pattern articles is 32; their sizes range from 7k to 63k. F-measures vary from 0 to 0.57 and are randomly distributed along the size ranges. Similar variations of F-measures occur with all other three extractors. Since the number of articles in each size group is small and

some size groups contain more zero-measure-value cases, it makes the curve fluctuated up and down wildly.

It can be image that if there are large number of testing corpora, say, 1000 testing articles, the variation of F-measures will be decreased and a better view of the relation between F-measures and corpora can be obtained.

7.5 Extraction on other Articles

Now we use the same three extractors to extract the keyphrases from another 20 pattern articles. These articles are randomly selected from the different years' PLoP conference paper collection [60]. The authors of these articles did not provide any keyphrases.

Since we do not have the author provided keyphrases as evaluation references, we could not give any evaluation on that. But readers can read original articles to make their own judgements based on these extraction results. The extraction results are listed in Appendix 2.

7.6 Summary

We have evaluated our developed extraction tools on both training corpora and testing corpora, compared with the extraction results from the other two extractors. We summarize the results as following:

- a) Our extraction tool obtained the highest F-measure value.
- b) Our extraction tool obtained the highest extraction rate.
- c) Our extraction tool also obtained the fewest zero-measure-values.

d) We found that zero-measure-value can seriously deteriorate the extraction performance.

Based on the results, we can claim that the adaptation of GenEx to pattern articles is successful.

Chapter 8 Conclusions

8.1 Summary of the Thesis

This thesis systematically reviewed the research work on the keyphrase extraction and related work, investigated the characteristics of software design pattern articles, and analyzed the important design concepts for the development of keyphrase extraction tool. Then we adapted GenEx, a wide category and domain independent algorithm, to the characteristics of design pattern articles, then selected the proper design patterns to develop the pattern-specific keyphrase extraction tool. After that, the tool was repetitively tuned with the parameters that reflect the characteristics of pattern articles, and the comparisons of the extracted keyphrases with author provided keyphrases, until the satisfactory results were obtained. The performance of the developed pattern-specific extraction tool was investigated on both training and testing corpora with the preset evaluation scheme, and compared with that of the other two extractors. The evaluation results were analyzed from different points of view.

The developed pattern-specific extraction tool shown the best performance with the highest average extraction rate, highest F-measure and lowest zero-measure-values, among the 3 investigated extractors. It confirmed that the adaptation of GenEx to the characteristics of pattern articles was successful.

The variations of the extraction performances on the training corpora and testing corpora and on the size of articles were analyzed.

8.2 Future Work

Some work can be done to further improve the extraction performance.

- a) If more design pattern articles with author provided keyphrases can be obtained, further extractions analysis with our tool can be carried out, such as, investigating the effects of article size and length of keyphrases on extraction performance, and then adjusting the values of parameters to get even higher performance.
- b) String operation, such as string comparison and stemming, was very time consuming, which was the heaviest work in keyphrase extraction. It is worth introducing binary string comparison to see if it can improve extraction performance.
- c) Using sentence boundary to get all potential phrases and implementing the rest processes can remove those cross-boundary phrases (2-word and 3-word phrases), so as to decrease the non-meaningful phrase processes.
- d) Removing all of the phrases with occurrence frequencies of one. This can speed up the following processes and decrease the extraction time.

Bibliography

- [1] Turney, P.D. (1999). *Learning to Extract Keyphrases from Text*. National Research Council, Institute for Information Technology, Technical Report ERB-1057.
- [2] W. F. Tichy, A Catalogue of General-Purpose Software Design Patterns. University of Karlsruhe. www.ipd.ira.uka.de/~tichy/patterns/overview.html
- [3] J. O. Coplien, "A Pattern Definication", Bell Laboratories. <http://hillside.net/patterns/definition.html>
- [4] PLoP conference pages: <http://st-www.cs.uiuc.edu/~plop/>
- [5] European Pattern Languages of Programs <http://www.hillside.net/patterns/EuroPLoP/>
- [6] KoalaPLoPTM is one of the series of conferences in the PloP: <http://www.cs.rmit.edu.au/KoalaPLoP/>
- [7] Annual ACM Conference on Object-Oriented Programming, Systems, <http://oopsla.acm.org/>
- [8] Annual European Conference on Object-Oriented Programming. <http://www.ecoop.org/>
- [9] Pattern Home Pages: <http://hillside.net/patterns/>
- [10] Pattern web site: <http://www.patterndepot.com/pages/>
- [11] Software development methodologies: http://dmoz.org/Computers/Programming/Methodologies/Patterns_and_Anti-Patterns/
- [12] Pattern Almanac Demo Web Interface Page: <http://slappy.cs.uiuc.edu/fall01/team4/mock/default.htm>

- [13] The Interaction design pattern pages:
http://www.pliant.org/personal/Tom_Erickson/InteractionPatterns.html
- [14] Email communication with D. Roberts, organizer of 1997's PLoP (2000)
- [15] J. Y. Chai, "Learning and Generalization in the Creation of Information Extraction Systems", (AAAI'99), pp. 431-438, July, 1999
- [16] Luhn, H.P. (1958). The automatic creation of literature abstracts. *I.B.M. Journal of Research and Development*, 2 (2), 159-165.
- [17] Edmundson, H.P. (1969). New methods in automatic extracting. *Journal of the Association for Computing Machinery*, 16 (2), 264-285.
- [18] Marsh, E., Hamburger, H., and Grishman, R. (1984). A production rule system for message summarization. In *AAAI-84, Proceedings of the American Association for Artificial Intelligence*, pp. 243-246. Cambridge, MA: AAAI Press/MIT Press.
- [19] Paice, C.D. (1990). Constructing literature abstracts by computer: Techniques and prospects. *Information Processing and Management*, 26 (1), 171-186.
- [20] Johnson, F.C., Paice, C.D., Black, W.J., and Neal, A.P. (1993). The application of linguistic processing to automatic abstract generation. *Journal of Document and Text Management*, 1, 215-241.
- [21] Salton, G., Allan, J., Buckley, C., and Singhal, A. (1994). Automatic analysis, theme generation, and summarization of machine-readable texts. *Science*, 264, 1421-1426.
- [22] Kupiec, J., Pedersen, J., and Chen, F. (1995). A trainable document summarizer. In E.A. Fox, P. Ingwersen, and R. Fidel, editors, *SIGIR-95: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 68-73, New York: ACM.
- [23] Brandow, R., Mitze, K., and Rau, L.R. (1995). The automatic condensation of electronic publications by sentence selection. *Information Processing and Management*, 31 (5), 675-685.
- [24] Jang, D. H., and Myaeng, S.H. (1997). Development of a document summarization system for effective information services. *RIAO 97 Conference Proceedings: Computer-Assisted Information Searching on Internet*, pp. 101-111. Montreal, Canada.
- [25] MUC-5. (1993). Proceedings of the Fifth Message Understanding Conference. California: Morgan Kaufmann.
- [26] Soderland, S., and Lehnert, W. (1994). Wrap-Up: A trainable discourse module for information extraction. *Journal of Artificial Intelligence Research*, 2, 131-158.

- [27] Muñoz, A. (1996). Compound key word generation from document databases using a hierarchical clustering ART model. *Intelligent Data Analysis*, 1 (1), Amsterdam: Elsevier.
- [28] Nakagawa, H. (1997). Extraction of index words from manuals. *RIAO 97 Conference Proceedings: Computer-Assisted Information Searching on Internet*, pp. 598-611. Montreal, Canada.
- [29] Krulwich, B., and Burkey, C. (1996). Learning user information interests through the extraction of semantically significant phrases. In M. Hearst and H. Hirsh, editors, *AAAI 1996 Spring Symposium on Machine Learning in I* Ginsberg, A. (1993). A unified approach to automatic indexing and information retrieval. *IEEE Expert*, 8, 46-56.
- [30] Steier, A. M., and Belew, R. K. (1993). Exporting phrases: A statistical analysis of topical language. In R. Casey and B. Croft, editors, *Second Symposium on Document Analysis and Information Retrieval*, pp. 179-190.
- [31] Fagan, J.L. (1987). Experiments in Automatic Phrase Indexing for Document Retrieval: A Comparison of Syntactic and Non-Syntactic Methods. Ph.D. Dissertation, Department of Computer Science, Cornell University, Report #87-868, Ithaca, New York.
- [32] Ginsberg, A. (1993). A unified approach to automatic indexing and information retrieval. *IEEE Expert*, 8, 46-56.
- [33] Leung, C.-H., and Kan, W.-K. (1997). A statistical learning approach to automatic indexing of controlled index terms. *Journal of the American Society for Information Science*, 48 (1), 55-66.
- [34] E. Brill, Some Advances in Rule-based Part of Speech Tagging, in: Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI Press, 1994). <http://www.cs.jhu.edu/~brill/>.
- [35] P. Turney, "Extraction of Keyphrases from Text: Evaluation of Four Algorithms". National Research Council, Institute for Information Technology, Technical Report ERB-1051.
- [36] Verity's 97, <http://www.verity.com/press/1998/980316.html>
- [37] Witten, I.H., G. Paynter, E. Frank, C. Gutwin and C. Nevill-Manning. 1999. KEA: Practical automatic keyphrase extraction. In Proceedings of DL-99, pp. 254-256.
- [38] Whitley, D. (1989). The GENITOR algorithm and selective pressure. *Proceedings of the Third International Conference on Genetic Algorithms (ICGA-89)*, pp. 116-121. California: Morgan Kaufmann.
- [39] A Pattern Definition: <http://hillside.net/patterns/>
- [40] Patterns-discussion FAQ, maintained by Doug Lea.

- <http://g.oswego.edu/dl/pd-FAQ/pd-FAQ.html>
- [41] The guest editorial for the Communications of the ACM, Special Issue on Patterns and Pattern Languages, Vol. 39, No. 10, October 1996.
 - [42] <http://www.cs.wustl.edu/~schmidt/CACM-editorial.html>
 - [43] Sherif M. Yacoub, "Composite Filter Pattern: Tech Report." HPL-2001-131. www.hpl.hp.com/techreports/2001/HPL-2001-131.html
 - [44] UML Toolkit Hans-Erik Eriksson, Magnus Penker, ISBN: 0-471-19161-2
 - [45] C. Bono, Course notes: <http://www-scf.usc.edu/~csci201/notes/CRC.pdf>
 - [46] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley, 1995.
 - [47] U. Pfeifer, T. Huynh, "Weighing". Course notes: <http://ls6-www.informatik.uni-dortmund.de/ir/projects/freeWAIS-sf/old/node27.html>
 - [48] S. Bird, course notes, www.cis.upenn.edu/~cis530/materials/CL11.pdf
 - [49] The 4th Pattern Languages of Programming Conference <http://st-www.cs.uiuc.edu/users/hanmer/PLoP-97/Workshops.html>
 - [50] The SuperJournal Project, "Stopword list". <http://www.mimas.ac.uk/sj/application/demo/stopword.htm#TOP> a
and TechTarget, "Stopwords". http://searchwebmanagement.techtarget.com/sDefinition/0,,sid27_gci798881,00.html
 - [51] R.K. Belew, Finding Out About Cambridge Univ. Press, 2000 Stemming and other morphological processing <http://www.cs.ucsd.edu/~rik/foa/l2h/foa-2-3-1.html>
 - [52] Porter, M.F. (1980). An algorithm for suffix stripping. *Program; Automated Library and Information Systems*, 14 (3), 130-137.
 - [53] Lovins, J.B. (1968). Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11, 22-31.
 - [54] R. Lazarinis, *Combining Information Retrieval with Information Extraction for Efficient Retrieval of Calls for Papers*. www.cs.strath.ac.uk/~mdd/research/publications/98lazarinis.pdf
 - [55] MUC-4. (1992). Proceedings of the Fourth Message Understanding Conference. California: Morgan Kaufmann.

- [56] W. Nelson Francis, Problems of Assembling and Computerizing Large Corpora, In Computer Corpora in English Language Research, Stig Johansson (ed.), Norwegian Computing Centre for the Humanities, Bergen, 1992
- [57] Web page for the Pattern Languages of Programs (PLoP) Conference:
<http://jerry.cs.uiuc.edu/~plop>
- [58] D. Turney, online extractor demo program
http://extractor.iit.nrc.ca/upload_demo.html
- [59] C. Gutwin, G. Paynter, I. Witten, C. Nevill-Manning, E. Frank, "Improving Browsing in Digital Libraries with Keyphrase Indexes".
<http://citeseer.nj.nec.com/gutwin98improving.html>
- [60] Appendix 2: testfile140 - testfile159

Appendix 1

The following are the extraction results using three different extractors on the training corpora (20 design pattern articles: testfile101 to testfile120), and the testing corpora (12 pattern articles: testfile121 to testfile132). The articles come from the PLOP conference 1997, www.plop97.html.

A1.1 Extraction Results with Training Corpora

Testfile101	Pattern Density and Role Modeling of an Object Transport Service		
Author's keyphrases	Framework, pattern, pattern density, pattern application, pattern composition, role, role model, role model composition, object transport, object migration, inter-process communication.		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Role, pattern, object, class, framework, role model, design pattern, pattern density, object transport, pattern density role	pattern, framework, design, role models, plays, communicator, composition	role, pattern, object, class, framework
Precision	0.600	0.429	0.600
Recall	0.545	0.273	0.273
F-measure	0.571	0.333	0.375

Testfile102	Describing Process Patterns		
Author's keyphrases	Process pattern, process, environments, patterns		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Pattern, description, process, environment, concept, description pattern, process pattern, describing pattern, procedure pattern, describing process pattern	pattern, process patterns, environment, standard process, proposals, natural language, understanding	pattern, process, description, problem, solution
Precision	0.400	0.428	0.400
Recall	1.000	0.750	0.500
F-measure	0.571	0.545	0.444

Testfile103	Improving the Representation of Configuration Design Patterns		
Author's keyphrases	Configuration design patterns, pattern representation, UML, new implementation techniques, frameworks.		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Pattern, design, diagram, component, UML, design pattern, class diagram, configuration design, representation configuration, configuration design pattern	patterns, framework, design patterns, instantiation, diagrams, representation, UML	pattern, framework, class design, diagram,
Precision	0.200	0.288	0
Recall	0.400	0.400	0
F-measure	0.267	0.333	0

Testfile104	Software Construction: Patterns for Building Software		
Author's keyphrases	Version Control, Patterns, Software Configuration Management, Replication, Repository, Build		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Version, software, pattern, context, SCM, software system, resulting context, related pattern, version repository, older version software	pattern, context, software system, control, environment, SCM, configuration	version, file, software, pattern, build
Precision	0.100	0.143	0.400
Recall	0.167	0.167	0.333
F-measure	0.125	0.154	0.364

Testfile105	Domain-Driven Framework Layering in Large Systems		
Author's keyphrases	Framework, framework layering, domain modeling		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Framework, business, layer, domain, application, business domain, role object, application system, framework layering, system abstract framework	framework, business, layer, layering, customer, desktop layer, investment	layer, framework, business, domain, section
Precision	0.200	0.143	0.200
Recall	0.667	0.333	0.333
F-measure	0.308	0.200	0.250

Testfile106	Customer Interaction Patterns		
Author's keyphrases	Customer interaction, communication, negotiation, personal relationships		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Customer, pattern, developer, interaction, context, customer interaction, active listening, interaction pattern, aware boundary, customer interaction pattern	customer, patterns, customer interaction, developers, trust, pattern language, communication	pattern, class, framework, design, figure
Precision	0.100	0.288	0
Recall	0.250	0.500	0
F-measure	0.143	0.364	0

Testfile107	A Data Flow Pattern Language		
Author's keyphrases	Data flow, payloads, module-data protocol, partitions		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Data, module, flow, application, different, data flow, module data, output port, inter-module communication, data flow architecture	data flow, filter, data flow architecture, payloads, pattern, communication, priority levels	Pattern, class, framework, design, figure
Precision	0.100	0.143	0
Recall	0.250	0.250	0
F-measure	0.143	0.182	0

Testfile108	Big Ball of Mud		
Author's keyphrases	Architecture, Evolution		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	System, code, architecture, pattern, software, pattern language, language program, throwaway code, software architecture,	architecture, programmers, patterns, design, emerge, throwaway code	system, code, architecture, pattern, work
Precision	0.100	0.143	0.200
Recall	0.500	0.500	0.500
F-measure	0.167	0.222	0.222

Testfile109	Client/Server Architectures for Business Information Systems		
Author's keyphrases	Gartner Styles, Client/Server cuts, Business Information Systems		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Architecture, database, distributed, system, client, information system, business information, distributed database, architecture business, business information system	distribution, architecture, information systems, server, client, business information systems, client/server architectures	application, architecture, server, database, client
Precision	0.100	0.143	0
Recall	0.333	0.333	0
F-measure	0.154	0.200	0

Testfile110	Patterns for GIS Applications Design		
Author's keyphrases	GIS, Decorator, Roles, Geographic Position		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	System, object, application, pattern, GIS	Reference system, pattern, o/oo, design, GIS, applications, abstraction	system, reference, object, role, measure
Precision	0.100	0.143	0.200
Recall	0.250	0.250	0.250
F-measure	0.143	0.182	0.222

Testfile111	The Role Object Pattern		
Author's keyphrases	Role modeling, context adaptation, framework integration		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Role, object, pattern, class, customer, role object, object pattern, component interface, component' interface role object pattern	role object, role object pattern, customer, interface, core object, key abstraction, object identity	role, object, core, pattern, customer
Precision	0	0	0
Recall	0	0	0
F-measure	0	0	0

Testfile112	Pattern Systems for Hypermedia		
Author's keyphrases	Hypermedia systems, Hypermedia navigational design, Hypermedia interface design		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	link, node, pattern, hypermedia, application, hypermedia application, related pattern, pattern system, node link, hypermedia design methodology	patterns, navigation, hypermedia, hypermedia applications, context, hypermedia systems, design methodologies	node, link, pattern, hypermedia, application
Precision	0	0	0
Recall	0	0	0
F-measure	0	0	0

Testfile113	Patterns for Team Dispersal		
Author's keyphrases	Team, Organization, Redistribution		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Team, pattern, success, organization, development, team member, successful team, communication system, team dispersion transplanted team member	patterns, successful teams, team members, architecture, success, self-directed work team, CONTEXT	team, member, work, architecture, pattern
Precision	0.20	0	0.200
Recall	0.667	0	0.333
F-measure	0.308	0	0.250

Testfile114	Dependent Demand -- Business Pattern for Balancing Supply and Demand		
Author's keyphrases	Patterns, Business patterns, Supply chain management		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Dependent, pattern, demand, supply, product, dependent demand, supply chain, chain management, component product, pattern supply chain	dependent demand, supply chain, pattern, independent demand, manufacturing, dependent demand network, supply chain management	Demand, dependent, supply, pattern, product
Precision	0.100	0.143	0.200
Recall	0.333	0.333	0.333
F-measure	0.154	0.222	0.250

Testfile115	Patterns for Conducting Process Improvement		
Author's keyphrases	Patterns, Process Improvement, Software Process		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Improvement, project, pattern, process, effort, improvement effort, process improvement, development project, product development, dedicated improvement processor	process improvement, patterns, PIT, improvement efforts, SPI, practitioners, SEPG	process, improvement, pit, project, pattern
Precision	0.200	0.288	0.200
Recall	0.667	0.667	0.333
F-measure	0.308	0.400	0.250

Testfile116	Organizational Patterns For Software Architecture		
Author's keyphrases	Software Architecture, Reuse, Organization, Management, Risk		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Component, organization, pattern, manager, software, component owner, software architecture, organization pattern, organization management, organizational pattern software	owners, pattern, architecture, managers, risks, informal network, software architecture	component, organization, owner, pattern, decision
Precision	0.200	0.288	0.200
Recall	0.400	0.400	0.200
F-measure	0.267	0.333	0.200

Testfile117	Managing Change to Reusable Software		
Author's keyphrases	Reuse, Management, Organization		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Platform, organization, application, software, project, configuration management, application development, platform identity, reusable software, reuse platform identity	platform, reuse, patterns, support, shared platform, reusable components, improvement efforts	reuse, platform, component, organization, change
Precision	0.100	0.143	0.400
Recall	0.333	0.333	0.667
F-measure	0.154	0.200	0.500

Testfile118	The Reliable Hybrid Pattern - A Generalized Software Fault Tolerant Design Pattern		
Author's keyphrases	Reliable Hybrid pattern, fault tolerant software, N-Version Programming, Recovery Block		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Pattern, software, hybrid, fault, design, reliable hybrid, hybrid pattern, fault tolerance, design pattern, Fault tolerance software	pattern, reliable hybrid pattern, fault tolerant, design, adjudication, acceptance, voting	Pattern, software, fault, version, hybrid
Precision	0.100	0.143	0
Recall	0.250	0.250	0
F-measure	0.143	0.182	0

Testfile119	Assembler		
Author's keyphrases	Grouping, builder, visitor		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Object, assembler, collection, groupingcriteria, criteria, collection object, assembled object, object groupingcriteria	Assembler, pattern, GroupingCriteria, isopleth, CalculationAlgorithm, criteria, Composite	Object, assembler, group, collection, pattern
Precision	0	0	0
Recall	0	0	0
F-measure	0	0	0

Testfile120	Distributed Proxy: A Design Pattern for Distributed Object Communication		
Author's keyphrases	Distributed Communication, Incremental Software Development, Design Patterns		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Distributed, proxy, object, pattern, communication, distributed proxy, distributed communication, distributed object, design pattern, distributed object communication	Distribution, proxy, communication, pattern, agenda manager, object references, physical layer	distribute, name, object, proxy, pattern
Precision	0.200	0	0
Recall	0.667	0	0
F-measure	0.308	0	0

A1.2 Extraction Results with Testing Corpora

Title 121	Roundabout, A Pattern Language for Recursive Programming		
Author's keyphrases	recursion, functional programming, novice programming		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Procedure, pattern, structure, data, recursion, helper procedure, structural recursion, structure data, pattern language, using structural recursion	Patterns, inductive data, syntax procedure, language, lambda, local procedure, mutual recursion	Procedure, helper, symbol, slist, replace
Precision	0.100	0	0
Recall	0.333	0	0
F-measure	0.154	0	0

Title 122	Two complementary patterns to build multi-expert systems		
Author's keyphrases	Architectural pattern, blackboard systems, dynamic control		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Knowledge, source, system, blackboard, pattern, knowledge source, blackboard system, problem solving, complementary pattern, Complementary pattern multi-expert	Control, Knowledge sources, Blackboard, Control plan, Pattern, Control component, enabled knowledge, source	Control, knowledge, source, system, blackboard
Precision	0.100	0	0
Recall	0.333	0	0
F-measure	0.154	0	0

Title 123	Lock Server		
Author's keyphrases	Lock Server, Resource Integration, Concurrency Control		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Resource, client, server, lock, system, lock server, resource client, distribution architecture, information system, resource lock server	resource, client, lock server, shared resources, concurrency, distribution, architecture	resource, client, lock, server, share
Precision	0.100	0.143	0
Recall	0.333	0.333	0
F-measure	0.154	0.200	0

Title 124	Patterns of Input Processing Software		
Author's keyphrases	Parsing, user interface		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	input, language, pattern, processing, parser, parsing processing, input processing, programming language, mainline process, input processing software	pattern, language, parsing, interface, context, programming languages, separate parsing,	Input, pattern, language, program, process
Precision	0	0.143	0
Recall	0	0.500	0
F-measure	0	0.222	0

Title 125	Locality of Reference		
Author's keyphrases	Caching, Locality of Reference, Efficiency		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	data, cache, memory, locality, reference, data cache, locality reference, secondary memory, datasourceref cache, principle locality reference	cache, requester, data source, memory, reference, locality, cache manager	Datum, cache, addr, source, memory
Precision	0.100	0.143	0.200
Recall	0.333	0.333	0.333
F-measure	0.154	0.200	0.250

Title 126	Design Patterns in Garbage Collection		
Author's keyphrases	memory management, garbage collection, tricolour rootset		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	object, collector, garbage, root, pattern, tricolor, garbage collector, design pattern, source root, pattern garbage collector	Garbage collection, garbage collector, Iterator, heap, patterns, design pattern, memory management	Object, lambda, collector, garbage, collection
Precision	0	0.288	0
Recall	0	0.667	0
F-measure	0	0.400	0

Title 127	Distributed Proxy: A Design Pattern for Distributed Object		
Author's keyphrases	Distributed Communication, Incremental Software Development, Design Patterns		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	distributed, proxy, object, pattern, communication, distributed proxy, distributed object, agenda manager, distributed communication, distributed object communication	Distribution, Communication, Proxy, pattern, agenda manager, design pattern, communication mechanisms	Distribute, name, object, proxy, server
Precision	0.100	0.143	0
Recall	0.250	0.333	0
F-measure	0.154	0.200	0

Title 128	Arguments and Results		
Author's keyphrases	Message, arguments, selector, results, lazy, future		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	object, argument, result, pattern, message, result object, argument object, message object, object protocol, protocol argument object	Patterns, protocol, server, client, lazy, Curried object, Programmers	Object, argument, result, pattern, message
Precision	0.300	0.143	0.600
Recall	0.500	0.167	0.500
F-measure	0.375	0.154	0.545

Title 129	Private Interface		
Author's keyphrases	Encapsulation, Reducing Visibility		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	class, member, interface, function, savehandler, member function, document class, savehandler class, private interface, savedata member function	Interface, member function, private, document class, saveHandler, saveInterface, saveData	Class, savehandler, document, member, interface
Precision	0	0	0
Recall	0	0	0
F-measure	0	0	0

Title 130	Branch Removal - A Pattern For Code Efficiency		
Author's keyphrases	Branch, speed, optimize		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	code, branch, performance, speed, compiler, compiler optimization, removal pattern, pattern code, pipelining performance, branch removal pattern	Instruction, pipeline, speeds, Complexity, int, CPU, pattern	Branch, int, if, code, triangle
Precision	0.200	0.143	0.200
Recall	0.667	0.333	0.333
F-measure	0.308	0.200	0.250

Title 131	Proactor, An Object Behavioral Pattern for Demultiplexing and Dispatching Handlers for Asynchronous Events		
Author's keyphrases	Proactive, Asynchronous, Concurrent, Reactive, Thread		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	operation, asynchronous, thread, pattern, HTTP, asynchronous operation, proactor pattern, object behavioral pattern	Asynchronous operation, handler, proactor pattern, client, request, HTTP, operating system	Operation, completion, asynchronous, thread, handler
Precision	0.200	0	0.400
Recall	0.400	0	0.400
F-measure	0.267	0	0.400

Title 132	The Layered Agent Pattern Language		
Author's keyphrases	Pattern Language, Agents, Mobile Agents		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Agent, layer, pattern, society, collaboration, layered agent, pattern language, agent society, layer agent, layered agent pattern	Agent, pattern, plan, collaboration, interface, layered agent, architecture	Agent, layer, pattern, object, plan
Precision	0.200	0.143	0.200
Recall	0.667	0.333	0.333
F-measure	0.308	0.200	0.250

Appendix 2

The following tables list the extraction results using three different extractors from 20 design pattern related articles. The authors of these articles did not provide the keyphrases. The articles come from the PLOP conference of 1998 to 2000.

Title 140	Stock Manager: An Analysis Pattern for Inventories		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Inventory, pattern, quantity, component, stock, component product, keep track, analysis pattern, inventory system, inventory keep track	Inventory, pattern Stock, distribution, Manufacturing, Analysis pattern, stock manager	inventory, pattern, stock, item, qty

Title 141	The Object Filter and Access Control Framework		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	client, object, filter, data, authentication, filter policy, concrete client, access control, object filter, object filter access	client, filter, authentication, request, server, filter policy, access rights	object, client, policy, filter, request

Title 142	A Pattern for Managing Distributed Workflows		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	workflow, pattern, distributed, manager, system, distributed workflow, managedistributedworkflow pattern, workflow management, distribution workflow, managing distributed workflow	Workflow, workflow server, workflow process, task managers, distributed workflow, workflow engine, workflow management	workflow, manager, task, process, server

Title 143	A pattern language for security models		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Pattern, model, system, security, authorization, Authorization pattern, operating system, pattern language, security model, pattern language security	pattern, security, authorization, access control, RBAC, pattern language, architectural level	pattern, role, access, user, model

Title 144	Fundamental Class Patterns in Java		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	class, method, pattern, interface, name, abstract superclass, primitive method, inheritance interface, simple class, simple class pattern	pattern, inheritance interface, abstract superclass, implementations, primitive method, narrow inheritance interface, complex methods	name, class, method, pattern, interface

Title 145	Groundswell: Two Patterns of Technical Leadership		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	team, practice, pattern, development, programming, programming guideline, development organization, following practice, pattern technique, pattern article experience	practice, patterns,shared practice, development organization, support, programming guidelines, programming environment	Team, practice, solution, pattern, share

Title 146	Implied Interface		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	interface, class, algorithm, implied, object, implied interface, explicit interface, functional polymorphism, concrete class, implied interface pattern	implied Interface, interface, implied interface algorithm, functional polymorphism, pattern, language, explicit interfaces	Interface, class, algorithm, method, implement

Title 147	Patterns for Internet shops		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	pattern, product, customer, shopping, catalog, shopping cart, catalog pattern, product customer, pattern infrastructure, shopping process pattern	pattern, customers, shopping, shopping process, web shops, catalog, selection	pattern, product, shop, customer, process

Title 148	Introducing Technology into the Workplace		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	technology, pattern, dedicated, champion, organization, dedicated champion, local leader, technology workplace, communication system,, introducing technology workplace	technology, patterns, dedicated champion, tecnologico para informatica, corporate angel, local leader, grass rotes	Technology, pattern, introduce, context, dedicate

Title 149	Workflow Patterns - A pattern language for workflow systems		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	workproduct, system, pattern, architecture, requirement, workflow pattern, workflow system, workflow requirement, pattern language, hard ensure workproduct	workflow, subsystem, patterns, architecture, workproduct, workflow requirements, responsibility	workproduct, workflow, pattern, system, state

Title 150	Workshop on non-software examples of software design patterns		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	example, pattern, software, workshop, iterator, design pattern, non-software example, software design, workshop participant, example software design	pattern, communication, software design, iterator, design patterns, non-software, channel	pattern software participant bread iterator

Title 151	Understanding and Using Patterns in Software Development		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	pattern, design, software, context, tool, design pattern, software design, conceptual pattern, software development, software design pattern	pattern, design patterns, materials, software development, context, pattern handbook, conceptual patterns	pattern, design, software, form, context

Title 152	An Analysis Pattern for Reservation and Use of Reusable Entities		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	reservation, pattern, entity, class, room, reservation reusable entity, reservation reusable entity	reservation, pattern, analysis pattern, diagram, customer, cancel, availability	entity, reservation, pattern, class, type

Title 153	Pattern-Oriented Software Architecture: A System of Patterns		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Pattern, chapter, text, software, design, design pattern, pattern text, software architecture, Pattern-oriented software, architecture system pattern	patterns, design patterns, software architecture, template, review, reading, reader	Pattern, book, chapter, system, text

Title 154	Notification Server		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	Notification, client, server, resource, database, notification server, client notification, passive resource, client notification server	notification server, clients, resource, shared resource, register, database, passive resource	notification, server, client, resource, share

Title 155	The Phrasebook Pattern		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	phrasebook, language, code, foreign, application,, foreign language, generic phrase, application code, phrasebook pattern	phrasebook, language,foreign language,SQL statements,pattern, lookup key	phrasebook, language, phrase, code, foreign

Title 156	Convenience Methods		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	method, parameter, default, convenience, client, convenience method, default parameter, client code, parameter convenience, parameter convenience method	convenience methods, client, interface, languages, client code, smalltalk, signature	method, parameter, default, change, convenience

Title 157	SCRUM: An extension pattern language for hyperproductive software development		
Name of extractor	My extraction tool	GenEx extractor	Microsoft Word97
Extraction results	scrum, sprint, software, development, pattern, scrum meeting, software development, backlog sprint, pattern language, daily scrum meeting	backlog, software development, patterns, developers, sprint, architecture, environment	Scrum, work, sprint, team, backlog

Title 158	"Identify the Champion" - An Organisational Pattern for Programme Committees		
Name of extractor	Our extraction tool	GenEx extractor	Microsoft Word97
Extraction results	paper, champion, meeting, review, member, champion detractor, paper review, member paper, programme committee, champion detractor paper	champion, meeting, review, members, detractor, PCC, acceptance	paper, pc, champion, review, meet

Title 159	Customer Interaction Patterns		
Name of extractor	My extraction tool	GenEx extractor	Microsoft Word97
Extraction results	customer, pattern, relationship, product, interaction, relationship sale, customer interaction, linda rising, team member, customer interaction pattern	customer, pattern, customer interaction, team members, context, developers, pattern language	Lease, resource, service, user, object

Appendix 3

In the appendix, we list all the text files in training corpora and testing corpora in the order of sizes. The sizes of the text files in the corpora range from 7k to 63k.

Training / Testing Corpora	Size (byte)
Testfile119	7,673
Testfile105	9,290
Testfile125	9,829
Testfile123	10,467
Testfile130	12,917
Testfile101	15,130
Testfile129	16,671
Testfile118	18,410

Testfile106	19,667
Testfile110	20,776
Testfile113	21,438
Testfile114	23,222
Testfile103	23,747
Testfile121	24,208
Testfile104	24,647
Testfile122	25,958
Testfile127	27,843
Testfile124	28,774
Testfile117	28,890
Testfile120	29,895
Testfile102	30,379
Testfile111	34,254
Testfile116	35,119
Testfile112	39,241
Testfile128	42,870
Testfile126	47,058
Testfile109	48,108
Testfile132	52,799
Testfile108	59,113
Testfile131	59,635
Testfile107	61,054
Testfile115	63,043