J Inf Process Syst, Vol.14, No.5, pp.1254~1271, October 2018
https://doi.org/10.3745/JIPS.04.0091

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Dependency Graph-Based Keyphrase Extraction Method Using Anti-patterns

Khuyagbaatar Batsuren*, Erdenebileg Batbaatar**, Tsendsuren Munkhdalai***, Meijing Li****, Oyun-Erdene Namsrai*****, and Keun Ho Ryu**

Abstract

Keyphrase extraction is one of fundamental natural language processing (NLP) tools to improve many text-mining applications such as document summarization and clustering. In this paper, we propose to use two novel techniques on the top of the state-of-the-art keyphrase extraction methods. First is the anti-patterns that aim to recognize non-keyphrase candidates. The state-of-the-art methods often used the rich feature set to identify keyphrases while those rich feature set cover only some of all keyphrases because keyphrases share very few similar patterns and stylistic features while non-keyphrase candidates often share many similar patterns and stylistic features. Second one is to use the dependency graph instead of the word co-occurrence graph that could not connect two words that are syntactically related and placed far from each other in a sentence while the dependency graph can do so. In experiments, we have compared the performances with different settings of the graphs (co-occurrence and dependency), and with the existing method results. Finally, we discovered that the combination method of dependency graph and anti-patterns outperform the state-of-the-art performances.

Keywords

Dependency Graph, Keyphrase Extraction

1. Introduction

An overwhelming amount of textual document has been published on semantic web, with numbers increasing each year exponentially. This huge growth has brought on needs to improve the document processing applications including document understanding, text summarization, and information retrieval. In those applications, keywords are considered as a brief summary of a text document so that readers and machines can easily analyze and decide whether a document is relevant for their purposes or not. However, there are many documents that have been publishing without any keywords and topics over the world. Consequently, automatic keyword extraction from a textual document has been very important task to many applications.

Manuscript received April 7, 2015; first revision February 1, 2017; accepted February 9, 2017.
Corresponding Author: Keun Ho Ryu (khryu@dblab.chungbuk.ac.kr)
* Doctoral School of Information and Communication Technology, University of Trento, Italy (k.batsuren@unitn.it)
** Database and Bioinformatics Laboratory, School of Electrical and Computer Engineering, Chungbuk National University, Cheongju, Korea ({eegii, khryu}@dblab.chungbuk.ac.kr)
*** Dept. of Quantitative Health Sciences University, University of Massachusetts Medical School, Worcester, MA, USA (tsendsuren.munkhdalai@umassmed.edu)
**** College of Information Engineering, Shanghai Maritime University, Shanghai, China (mjli@shmtu.edu.cn)
***** School of Engineering and Applied Science, National University of Mongolia, Ulaanbaatar, Mongolia (oyunerdene@seas.num.edu.mn)

The purpose of the task of keyword extraction is to automatically identify a set of keyphrases that appear in the document. Despite of the fact that keyphrase extraction has been extensively studied and discussed in a number of literatures, the performances of them are not perfect as other tasks in natural language processing (NLP) like the POS Tagger, Named Entity Recognition (NER). NER is a crucial component of biomedical NLP [1-3], enabling keyword extraction and ultimately reasoning over and knowledge discovery from textual document. There has been a surge of interest of applying machine learning [4], deep learning [5] techniques in common biomedical keyword and information extraction. Therefore, the task of keyphrase extraction is still very crucial to be improved.

The state-of-the-art methods can be generally categorized into two groups: supervised and unsupervised methods. Both groups of methods have same first step that is to generate all possible noun phrases from a document as a candidate set of keyphrases.

Supervised methods [6-10] extract syntactic, stylistic and statistic features for each candidate phrase, and then the famous machine learning algorithms (e.g., Naive Bayes, bagged decision tree, and neural network) have been employed to train a model on the extracted features. Finally, the models classify each candidate noun phrase whether it is a keyphrase or not.

The state-of-the-art unsupervised methods construct a graph between words of a document by exploiting the co-occurrence data of the words, and then PageRank algorithm run on the word co-occurrence graph to extract the importance weight of each term in the document. By using this important weights, the candidate set of keyphrases are ranked and top 5 or 10 candidates are chosen as keyphrases. This method has been first introduced by Mihalcea and Tarau [11] and called TextRank. Many other unsupervised methods extended its core idea, and developed different variations of TextRank.

According to the report of SemEval-2010 task [12] that is the world-wide competition on the task, the 19 teams participated this contest, and most of these systems used same syntactic, stylistic, and statistic features, and similar candidate ranking techniques while the two leading teams, HUMB and KP-Miner, removed the candidates with English stop words from the candidate set. Probably, it could be their secret to win the contest, and maybe it is due to the fact that just two percentages of all possible candidate phrases of the dataset are keyphrases. In this result, it raises the research question whether there are words as stop word that appear frequently in non-keyphrase candidates and are not included in any keyphrase. The reason why it should be necessarily studied is that many non-keyphrase candidates share such words. For instance, suppose that the keyphrases are being extracted from the scientific article documents, and then in this domain all candidates with “paper” term can be easily recognized as non-keyphrases while the unsupervised state-of-the-art methods could prioritize the candidates with “paper” term as keyphrase just because the “paper” term appear more frequently than other important terms in the documents. In this paper, we call these types of stop words as “anti-pattern”.

By being motivated from it, we propose a graph-based keyphrase extraction method using anti-patterns. For candidate ranking technique, we select the state-of-the-art graph-based ranking algorithm on TextRank [11]. Other novel idea is to use a dependency graph, introduced in [13], instead of using a word co-occurrence relation-based graph. We also modify this graph and construct the modified dependency graph that embed its syntactic features with properties of some statistic and stylistic features of the document.

The method we put forward has a number of desirable advantages:

(1) It generates anti-patterns which aim to filter out non-keyphrase candidates from the candidate set, and are thus able to be significantly useful in any keyphrase extraction method.
(2) It is robust to non-keyphrase candidates which share commonly stylistic and syntactical features.
(3) It uses an efficient and scalable graph that is based on dependency graph and adopts statistical and syntactical features of terms.

The rest of this paper is organized as follows. In Section 2, we will review some related works. Section 3 contains more details about the novel ideas, proposed in this method: dependency graph and anti-pattern. In Section 4, the dependency graph-based keyphrase extraction method using anti-patterns is explained. Section 5 provides the experimental analysis, evaluation and the related results followed by the discussion of our results. Finally, Section 6 concludes our paper.

2. Related Work

The keyphrase extraction methods are grouped into two categories: unsupervised or supervised keyphrase extractions. The state-of-the-art methods in both groups have two common steps: first is to generate all possible candidate phrases, and the second step is either to rank or to classify candidates.

Unsupervised Method

The unsupervised state-of-art method is the graph-based ranking methods. Mihalcea and Tarau [11] first applied a graph-based ranking algorithm (TextRank) for keyword extraction. TextRank was inspired by PageRank by using the ranking algorithm for a text and builds a graph representing a text. Every node V_i corresponds to a lexical unit. The goal is to calculate the score of each node $WS(V_i)$ which reflects its importance, and then adopt the words types that correspond to the highest-scored vertices to form keyphrases for a given text. $WS(V_i)$ is initialized with a default value and computed in an iterative manner as follows a recursive formula.

$$WS(V_i) = (1 - \mathrm{d}) + \mathrm{d} \sum_{V_j \in in(V_i)} \frac{w_{ij}}{\sum_{V_k \in out(V_j)} w_{jk}} * WS(V_j) \quad (1)$$

where w_{ij} is the weight of direct edge (V_j, V_i), $In(V_i)$ is the set of vertices that point to vertex V_i , and $Out(V_j)$ is the set of vertices that vertex V_j points. d is the damping factor usually set to 0.85, as in the PageRank algorithm.

The core idea of TextRank has been extensively exploited in many unsupervised systems. For instance, ExpandRank [14] is a combination of TextRank, k-nearest neighbors (KNN), and Tfidf. This method first uses a small number of nearest neighbor documents to provide more knowledge to improve keyphrase extraction. After finding KNN of the document using Tfidf and cosine similarity, the graph for the document is built using their similarities and co-occurrence statistics. The rest of the procedure can be similarly performed as TextRank.

A topic decomposition framework was proposed by Liu et al. [15]. It first recognize a topic distribution from the dataset using Latent Dirichlet Allocation (LDA) model [16]. Extracted topic distribution has a number of topics, each of which related to a group of words. In their work, multiple

random walks are performed for the topics instead of the traditional single random walk through the graph. Later on, other topical keyphrase extraction methods [17,18] have been applied to Twitter dataset.

The graph-based unsupervised methods give the less importance for words that are placed far away from each other in a document while the words are syntactically related in the sentence. To solve this issue, the modified dependency graph has been proposed in this paper. This graph has a number of desirable advantages:

- it contains the well-known term frequency features.
- in this graph, verb nodes only vote to noun and adjective nodes as important, and noun and adjective nodes vote to each other because keyphrases are composed of adjective and noun words.
- it can connect related words, placed far away from each other.

Supervised Method

A number of supervised methods in keyphrase extraction have been proposed for different types of classification methods. For instance, the Bayesian classifier, called KEA, is used by the system by Witten et al. [9]. Then, this system was improved as KEA++ by Medelyan and Witten [10] by using semantic information on terms and phrases gleaned from a domain specific thesaurus. The neural network based approaches [7,8] have also been studied widely.

Before training a model of all above supervised methods, the features are extracted from the training data set. Moreover, the most important features are the frequency and location of the phrase in the document. More linguistic knowledge has been explored by Hulth [19]. Nguyen and Kan [20] presented keyphrase extraction in scientific articles by using features that capture the logical position and additional morphological characteristics of scientific keywords. Naïve-Bayes based method has been applied to the medical domain, which has been tested on a small set of 25 documents. These studies have been utilized in many different domains such as medical domain, computer science articles, web pages, and news.

Given the fact that at least ninety percentages of all possible candidates are non-keyphrases in different datasets including medical domain, scientific article, and news, all supervised state-of-the-art methods try to find a rich feature set to recognize keyphrases while the features are not even able to cover many keyphrases of the training data due to fact that many keyphrases often share no common pattern or feature with each other while non-keyphrase candidates share common words and features. Therefore, the proposed method exploits this idea and removes non-keyphrases accurately from the candidate set by using anti-patterns.

By combining the two solutions for each group, we propose a dependency graph-based keyphrase extraction using anti-patterns.

3. Dependency Graph and Anti-pattern

In this section we describe the dependency graph and the anti-patterns that our proposed method is based on. We first define what the basic dependency graph is and how it can be modified for keyphrase extraction task. Then we explain what the anti-patterns are, categorize the different types of the anti-patterns, and how it can be measured.

3.1 Basic Dependency Graph

The dependency graph represents a text document and interconnects words with grammatical relations. This graph is first introduced in [13]. A text is divided into tokens or words, each of which represents a node of the graph. After parsing all typed dependency relations from all sentences, the nodes are connected with weighted and directed edges based on typed dependency relations. A value of the edge is a sum of values of the dependency relations between its nodes. A process of building dependency graph consists of following main steps:

1. Parse typed dependency relations for each sentence.
2. Identify each distinct term that appear in a document and add them as vertices in the graph.
3. Draw edges between vertices in the graph using these relations. Edges are directed and weighted.

The Stanford dependency parser (SDP) is used to detect grammatical relations between words in a text. It provides a representation of grammatical relations between words in a sentence. The current representation contains approximately 50 grammatical relations. Moreover, SDP has four kinds of models. In our research approximately 30 collapsed dependencies are utilized because the collapsed dependency summarizes two basic dependencies, and some dependencies are unnecessary for our task. Therefore, these unnecessary dependencies such as det, predet, aux, and advmod are not used. In Fig. 1, the sample dependency graph for an abstract from Inspec dataset is shown. The sample abstract is "*Compatibility of systems of linear constraints over the set of natural numbers. Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types systems and systems of mixed types.*" that same as the example abstract on the study of Mihalcea and Tarau [11].

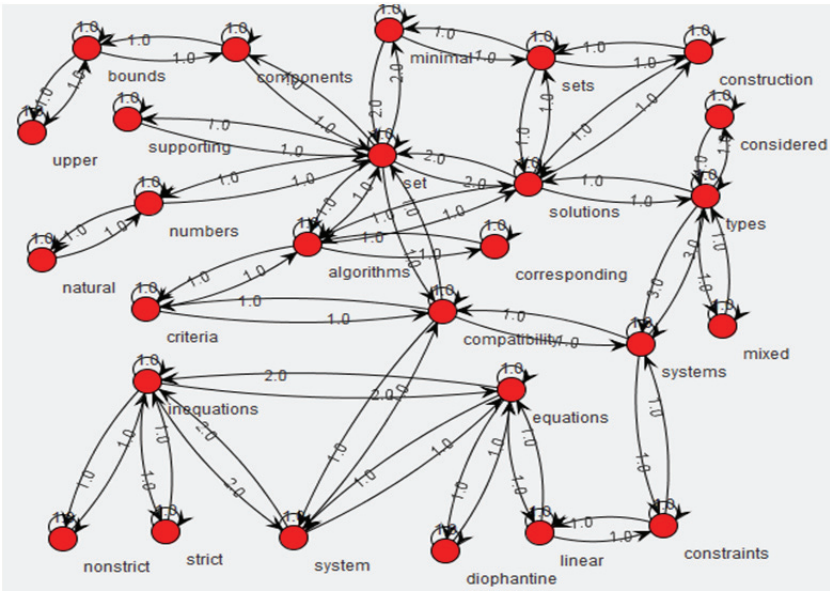


Fig. 1. The basic dependency graph for the sample abstract of Inspec dataset.

3.2 Modified Dependency Graph

The basic dependency graph can be modified for the task of keyword extraction. In order to improve the performances of our method, we modified the basic dependency graph. The vertices of the graph are categorized into a noun, adverb, adjective, adverb, and verb vertices. Moreover, nouns and adjectives are more important than determiners, adverbs and verbs due to keyphrase consists of only adjectives and nouns. As we mentioned, term frequency feature is very useful for this task. Therefore, the purpose of our modification is to consider the noun and adjective nodes as more important than other types of nodes and to make the graph which takes term frequency feature. This modification consists of following steps:

(1) Assign the unique ID for every verb and adverb in each sentence before constructing a graph, and assume each verb and adverb in different sentences are different than each other although verbs with the same value and meaning are used in two sentences. Finally, add them as different vertices in the graph.

(2) Set the value of term frequency to the value of direct edge which connect a node to itself.

(3) Draw edges between vertices in the graph using these relations. Edges are directed and weighted.

After processing first step, the importance values of the adverb and verb nodes in the graph would be decreased and it helps to increase an importance of the noun nodes. At the second step, the noun and adjective nodes have their term frequency a number of which is higher than 1. But, all adverb and verb nodes have the term frequencies of 1 due to every adverbs and verbs in the document is different from one another. By doing step 2, it increases the potential of nodes of frequent noun words while TextRank algorithm is running. By doing step 3, the noun vertices only recommend other noun and adjective vertices as an important, and the adjective vertices only recommend noun vertices because of adjective

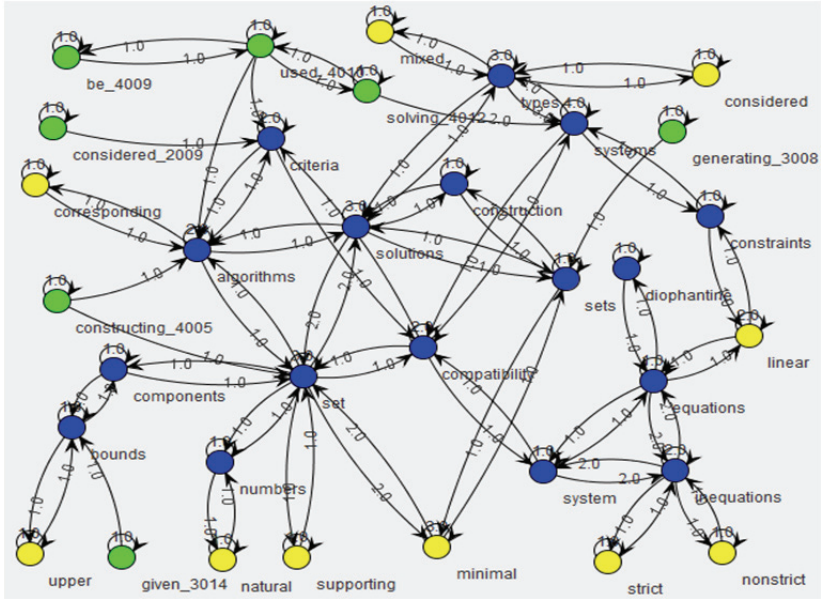


Fig. 2. The modified dependency graph for the sample abstract of Inspec dataset.

words can only connect noun words in a dependency tree. After this modification, the importance values of the noun and adjective nodes would increase and the importance values of other types would decrease. The example of the modified dependency graph is shown in Fig. 2.

3.3 Anti-pattern

Many candidates with English stop-words are found after generating the candidate phrases from the dataset [12]. The candidates are not keyphrases, and could be easily filtered out with English stop-words. However, there are many terms, not included in English stop-word list, can also be used to remove non-keyphrase candidates from the candidate set. We call those terms as anti-pattern.

Definition: Anti-pattern is a word that often appears in non-keyphrase candidates.

Let us consider the following sample of the training dataset in Table 1 that lists the 10 candidate phrases with their keyphrase class labels and two types of candidates where the candidate phrases that are composed of two or more words are called compound while the candidates with only one word is called a single word.

Table 1. Training dataset for anti-pattern extraction

ID	Candidate phrase	Type	Keyphrase label
1	Corresponding keyphrase	Compound	No
2	Keyphrase extraction	Compound	Yes
3	Linear constraints	Compound	Yes
4	Set	Single word	No
5	New systems	Compound	No
6	New keyphrase extraction	Compound	No
7	Set	Single word	No
8	Linear corresponding algorithms	Compound	No
9	Experimental study	Compound	No
10	Important study	Compound	No

As described in Table 1, same word often has different meanings on different location of compound candidate phrase. For example, the candidates in Table 1 with the ending word, “keyphrase”, should be removed, but other candidates with the starting word, “keyphrase”, should be kept. Therefore, to handle those situations (e.g., compound vs single word, and same word in different locations), we create the four types of anti-patterns: head word, tail word, single word, and anti-word. More detailed information of these types is shown in Table 2.

Table 2. Anti-pattern

Type	Description	Support count	Example
Head word	First word of candidate phrase	5	New
Tail word	Last word of candidate phrase	5	Study
Single word	Candidate phrase with one length	2	Set
Anti-word	Any word of candidate phrase	8	Corresponding

Head word, tail word, and anti-word anti-patterns are depending on where those anti-patterns appear in candidate. However, single word anti-pattern is relevant in candidates with one length. Many words which are candidate with one length are recommended as keyphrases, but most of those words are non-keyphrase candidates.

There is one key issue that needs to be addressed when anti-patterns are being extracted from the training dataset. Some of these discovered patterns are potentially spurious because they may happen simply by chance. The strength of anti-pattern can be measured in its support count and confidence which are used in association analysis. We follow the definition of these two metrics in [21]. First metric support count, σ(*X*), is a number of candidates that match anti-pattern *X*. The formal definition of confidence metric is as follows:

$$Confidence\ (X) = \frac{\sigma(Keyphrase\ l\ abel\ = "Yes" \bigcap X)}{\sigma(X)} \tag{2}$$

where *X* is a pattern, σ(*X*) is representing a number of candidate noun phrases which matches *X*. And c is a threshold. If *Confidence*(*X*) is closer to a zero, it is a strong pattern. Of course, a larger training dataset is required to extract the anti-patterns with higher confidences. Although the anti-patterns are only covering a training domain, some strongest patterns can help to prune non-keyphrases of other domains.

4. Proposed Method

The purpose of our study is to improve the state-of-art graph-based keyphrase extraction by using anti-patterns. Our proposed method consists of three primary components: anti-pattern generation, a candidate phrase filtering, and a graph-based keyphrase extraction. Fig. 3 shows the overview of our method. A goal of each component is explained shortly as follows:

1. All the generated candidates for document dataset I are utilized to extract anti-patterns.
2. All the generated candidates for document dataset II are filtered by using anti-patterns. Candidates which match anti-patterns are eliminated from the candidate set.
3. To get importance score of each term for a document, TextRank is run on a dependency graph, representing a document. Finally, keywords are extracted by combining survived candidates and importance score of each term for a document.

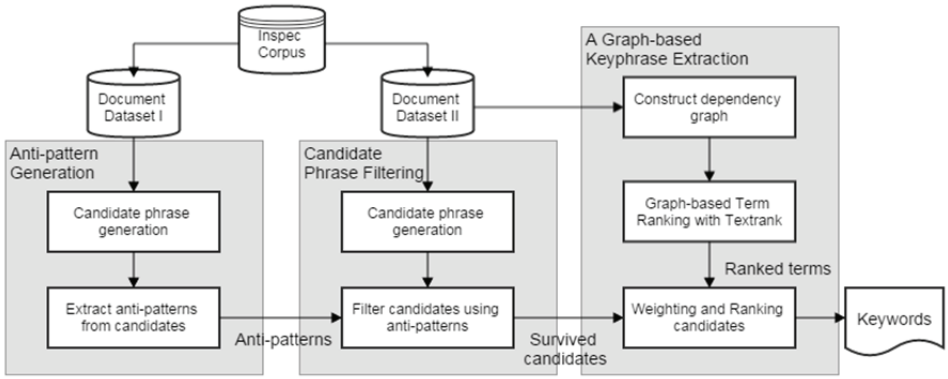


Fig. 3. The overview of our proposed method.

4.1 Anti-pattern Generation

For this component, most important work is the candidate phrase generation. We used Stanford log-linear POS tagger to obtain the part-of-speech tags of all documents. First step, candidate noun phrases are generated by matching sequential words with the pattern (adjective)*(noun)+, which represents zero or more adjectives followed by one or more nouns. After all possible candidate phrases are generated for document dataset I, each candidate phrase for every document is assigned by a label that is representing whether a candidate phrase is a keyphrase or not. Then, anti-pattern which satisfies minimum support count and maximum confidence are extracted from candidates for the training data set. In order to avoid overfitting, the max confidence is chosen as 0.09, and minimum support count for each kind of anti-patterns is shown in Table 2. Extracted anti-patterns are moved into next component to be used.

4.2 Candidate Phrase Filtering

All possible candidates for document dataset II are generated by matching a sequence of words with the pattern (adjective)*(noun)+ as same as mentioned in the first component. The generated candidates are filtered by using anti-patterns which are generated in first component. If a candidate matched any anti-pattern during the filtering, this candidate is directly eliminated from the candidate list. In this process, every candidate is checked by each anti-pattern whether it matches a candidate or not. Finally, all survived candidates in document dataset II are moved into next component.

4.3 Graph-Based Keyphrase Extraction

For a preprocessing of this component, the LingPipe sentence extractor are first used to detect sentences from a text document. Then SDP is used to extract the dependency relations from each sentence separately in a document. Even though the SDP has its own library to tokenize sentences of a text, this tool is an unsupervised traditional approach, so we preferred to use LingPipe. Moreover, the LingPipe sentence extractor is a supervised method based on the large training corpus. Before extracting relations, a sentence structure and proper grammar of a document are very important. Therefore, some preprocessing techniques such as tokenization, stemming, and removing stop words must not be used in a document. After extracting dependency relations, the dependency graphs are constructed as described in the dependency graph section. Then, the score associated with each node is set to an initial value of 1, and graph-based term ranking algorithm as described in Eq. (1) is run on the dependency graph until convergences. After all the final scores of nodes are converged, the method runs post-processing component.

Remember that just before post-processing phase, all candidate phrases are processed by anti-pattern filters, and the matched candidates are removed from the pool of all candidates. Then, only the survived candidate phrases are ranked by Eq. (3). The score of a candidate phrase p_i is computed by summing importance scores of words contained in the phrase.

$$PhraseScore(x) = \sum_{v_j \in Words(x)} WordScore(v_j) \tag{3}$$

All survived candidate phrases in the document are ranked in decreasing order of the phrases scores and top ranked k phrases are selected as the keywords. This parameter k ranges from 1 to 25.

5. Experimental Results and Evaluations

5.1 Dataset and Evaluation Metrics

In this experiment we have used the Inspec corpus that is a collection of 2,000 abstract documents with titles, extracted from journal papers published in Computer Science and Information Technology. Each abstract has two different sets of keyphrases assigned by the indexers: the first set is the controlled keyphrases which appear in the Inpsec thesaurus, and the second set is the uncontrolled keyphrases which do not necessarily appear in the thesaurus. The corpus is one of very popular datasets for automatic keyphrase extraction task, as the usage of several famous researchers as first used by Hulth [19], and later by Mihalcea and Tarau [11] and Rafiqul Islam and Rakibu Islam [22]. In their experiments, the 2,000 abstracts were divided into 1,000 for training, 500 for development, and 500 for the test. Since our system is a supervised method that needs to train a model to extract the anti-patterns, we used 1,000 abstracts for training and 500 for test to compare with previous existing systems. In the evaluation, we chose the standard metrics of the precision (p), recall (r) and the f-measure (F) as follows:

$$Precision = \frac{c_{correct}}{c_{extract}} \tag{4}$$

$$Recall = \frac{c_{correct}}{c_{normal}} \tag{5}$$

$$F - measure = \frac{2*precision*recall}{precision+recall} \tag{6}$$

where $c_{correct}$ is a number of correct keyphrases detected by a method, $c_{extract}$ is a number of all keyphrases detected by a method, and c_{normal} is a number of human-labeled keyphrases provided from the corpus.

In a rest of this section, we discuss the experimental results. First, we investigate whether the types of anti-patterns are efficient to be used for this task. Second, we evaluate the two types of dependency graph with the word co-occurrence graph. Finally, we compare the results of our method with the best results of previous works.

5.2 Results of Candidate Phrase Generation

The generated candidate set for training documents contains 28,228 candidate phrases while the method should find the 9,788 keyphrases from the set. As mentioned before, only some keyphrases assigned by indexers appear in documents. Therefore, 5,081 of total 9,788 keyphrases are only included in the training candidate set. Also some keyphrases that even appear in the documents could be lost due to a misclassification error of the POS tagger we used in the method. Fig. 4 shows histograms by a number of words in the candidates for the training set. As can be seen clearly, the percentage of keyphrases is relatively smaller by comparing with total number of the generated candidates for each histogram.

5.3 Results for Anti-patterns

In this part of experiments, we wanted to investigate which type of the anti-pattern is efficient in the

certain conditions, so that we compared the performances of the baseline method with each type separately. As the baseline method, we chose the modified dependency graph unsupervised method. Fig. 5 shows the results as well as the combined performance with all four types. As shown in the figure, until a number of keyphrases increases to 12, the f-measures of all methods increase where the anti-word pattern beat other three types. However, when the number is greater than 12, all method performances are dropping where the single-word anti-pattern has the best performance because the one-length candidates have been often ranked as later cases in the priority. Otherwise, a number of candidate phrase with only one word increases dramatically.

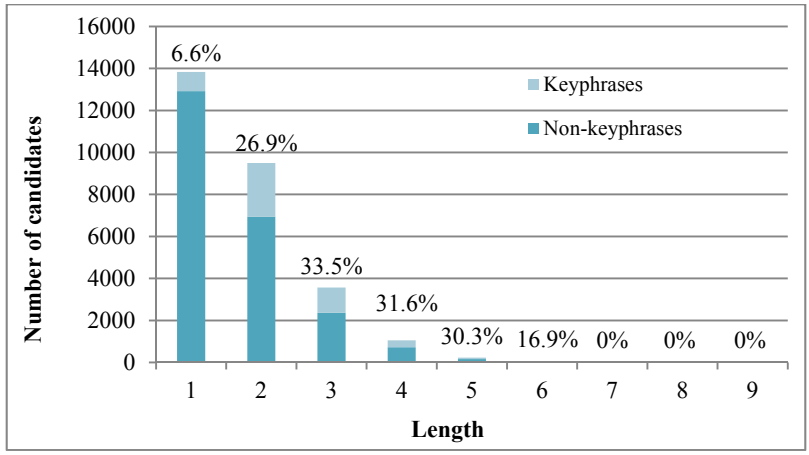


Fig. 4. The histograms of generated candidate phrases for training dataset.

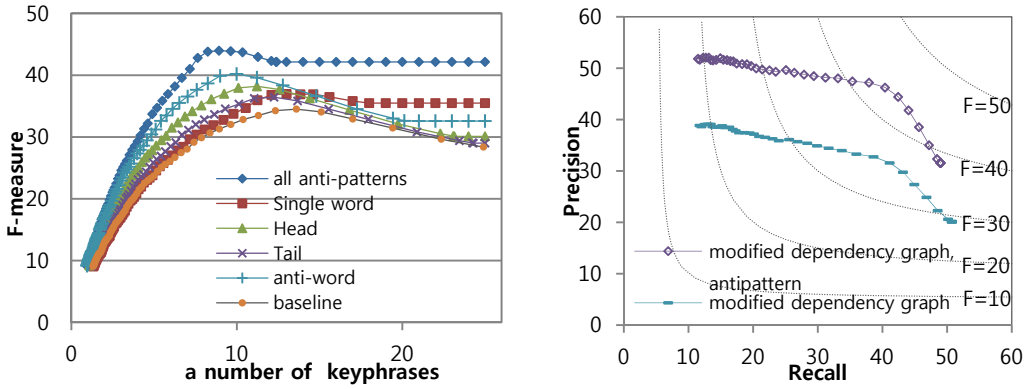


Fig. 5. Results of anti-patterns. (a) F-measures of four types of anti-patterns. (b) Influences of anti-patterns.

By applying the anti-patterns, the precisions increased highly while the f-measures were slightly dropped as can be seen from Fig. 5(b).

When the support counts for the anti-patterns are set lower than the default values, the precision and recall of our method drop with together from the best result. Thus, when threshold parameters are set higher than the default values, the recall of our method increases while precision and f-measure drop than the best result.

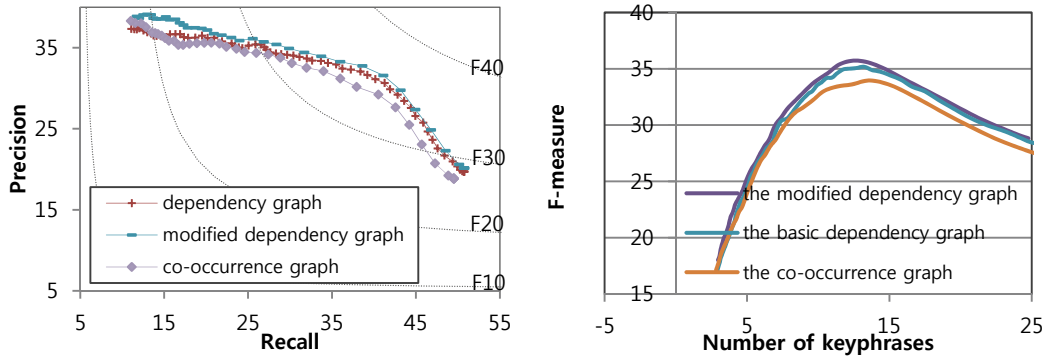
Table 3 lists ten examples for each type of anti-patterns with numbers representing how many times the corresponding term appear in both training and testing candidate sets. For instance, the anti-word "results" eliminated 3 keyphrases and 84 non-keyphrases from the candidate set for training dataset while it filtered out 1 keyphrases and 64 non-keyphrases from the testing candidate set.

Table 3. Ten samples for four kinds of anti-patterns

Anti-word			Single word			Head word			Tail word		
Term	**Train**	**Test**	**Term**	**Train**	**Test**	**Term**	**Train**	**Test**	**Term**	**Train**	**Test**
paper	0/157	0/103	method	0/35	0/47	various	0/26	0/24	types	0/12	0/13
results	3/84	1/64	system	0/45	0/46	important	0/12	0/23	ways	0/13	0/12
article	0/19	0/47	model	0/27	0/34	more	0/27	0/19	range	1/15	0/12
different	0/56	0/53	use	0/33	0/33	good	0/11	0/19	use	0/8	0/10
number	0/49	0/51	problem	0/34	0/32	major	1/11	0/18	example	0/6	0/10
important	0/29	0/35	time	0/29	0/30	additional	0/8	0/17	one	0/8	0/9
study	0/53	0/33	systems	0/21	0/29	novel	0/13	0/17	years	0/11	0/9
experimental	1/32	1/39	data	0/21	0/29	previous	0/12	0/16	due	0/8	0/9
application	0/58	1/34	approach	0/25	0/29	recent	0/14	0/16	power	1/12	0/8
available	0/15	1/33	order	0/24	0/25	certain	0/11	0/15	role	0/8	0/8

5.4 Graph Comparison

In this evaluation, we aimed to demonstrate the performance differences between all three kinds of graphs: co-occurrence graph, the basic and modified dependency graphs. In Fig. 6, the curves of the graphs are plotted to express the performances of precision, recall, and f-measure.



(a) (b)

Fig. 6. Comparison between the dependency graph and the word co-occurrence graph. (a) Precision and recall curves. (b) F-measure curves.

In general, the modified dependency graph is the best performing graph for all performances. However, the word co-occurrence graph beats the modified dependency graph when a number of

candidate keyphrases is smaller than 5 for supervised method. As can be seen from Fig. 6(a), the modified version of the dependency graph clearly outperformed other graphs for unsupervised method. According to the results of Tfidf and TextRank on previous several studies in [23,24], these works proved that Tfidf is the best performing unsupervised method although Tfidf is the simplest method. From so far what we understood, the modified dependency graph exploits the main characteristics of Tfidf. Therefore, as shown in Fig. 6(b), the modified dependency graph is the graph with the best performances.

5.5 Results Compared with Previous Methods

In order to verify the efficiency of a novel-graph based keyword extraction using anti-pattern, in Table 4, we compared its performances with several previous works on the Inspec corpus. For each method, Table 4 lists the total number of keywords assigned, the mean number of keywords per abstracts, total number of correct keywords, as evaluated against the set of keywords assigned by professional indexers, and the mean number of correct keywords.

Table 4. Comparing proposed method with previous methods

Method	Assigned		Correct		Precision	Recall	F-measure
	Total	Mean	Total	Mean			
TextRank [11]	6,784	13.7	2,116	4.2	31.2	43.1	36.2
Hulth [19]	7,815	15.6	1,973	3.9	25.2	51.7	33.9
Rafiqul Islam and Rakibu Islam [22]	6,114	12.23	2,386	4.8	39.1	48.7	43.4
Our method	5,984	12.0	1,946	3.9	32.5	39.6	35.7
Our method + anti-pattern	4,446	8.9	2,095	4.2	47.0	42.5	44.7

The table also lists precision, recall, and f-measure metrics.

5.6 Discussions

As described above, the system we developed for keyword extraction is based on the dependency graph-based TextRank and anti-patterns. The anti-patterns filtered out some unwanted candidate phrases from the candidate phrase set, and then a novel graph-based TextRank ranks candidate phrases. We evaluated the proposed method on the Inspec dataset. While constructing the dependency graphs, various models [25-27] of SDP, i.e., the PCFG model, the factored model, the PCFG caseless model were used for the experiments. By comparing the performances of those models, we chose the PCFG model that has the best result than others. If we use the other dependency parser except Stanford, it is possible to outperform the current best result of our system. We also evaluated the effects of four types of anti-patterns: anti-word, head word, tail word and single word. As shown in Fig. 6(a), when a number of keyphrase is smaller than 12, except for a single word, each kind of anti-pattern had a significant effect. While a number of keyphrase increases than 12, only single-word anti-pattern had a significant effect. This conundrum is related to Inspec dataset that contains relatively a small number of candidate phrases than other datasets. The overall time in the Inspec dataset to train the anti-patterns and extract keyphrases from the test dataset was about 2 hours.

In order to reach the current best result of our system, we developed some additional works that aren't described above. After the score of each candidate, phrase is calculated by using Eq. (3). We filtered out some candidate phrases, the score of which is smaller than a value of its length. Even though this work is not mentioned in our experimental work, we think it has a significant effect.

A models are trained for a specific dataset using machine learning techniques, i.e., SVM, neural network, decision tree, and it is only used to recognize whether a candidate phrase is a keyphrase or not. Moreover, if these models are utilized for another domain to identify correctly keyphrase, it reaches a very poor result. Because the trained model only depends on the domain it is trained on. Therefore, it can't identify keyphrases for another domain. However, some rules and parts of those models can be used to recognize correctly some non-keyphrases for another domain although those rules and part could recognize keyphrases for another domain.

6. Conclusions

Due to the fact that a volume of text data on Internet has been increasing dramatically for past decades, the keyphrase extraction has been a leveraging method of text-mining applications such as document summarization and understanding, and information retrieval. Previous studies for supervised keyphrase extraction are mostly focused on extracting a rich feature set to identify only keyphrases. However, in regarding to aspects of keyphrases on documents, it shared no similar patterns and relations as other NLP tasks such as NER. Therefore, those rich feature set often is unable to cover all keyphrases. Nevertheless, non-keyphrase candidates often share similar patterns and relations.

In this paper, we proposed a dependency graph-based keyphrase extraction method using anti-patterns. The widely used word co-occurrence graph of a graph-based keyphrase extraction has relations, limited by a window size and its words in a window are fully connected. However, it has no syntactic relation, and does not adopt enough statistical and stylistic features of words. Instead of traditional co-occurrence graph, a novel graph that is based on dependency graph is proposed to solve such problems. However, there are still no studies about comparison between dependency and co-occurrence graphs in keyphrase extraction. This paper proposed to use anti-patterns in order to filter out some unwanted phrases from the candidate phrase set. The contribution is that we have proved anti-patterns can be efficiently used to filter the candidate set and it had very significant effects.

The experimental results showed that our proposed method outperformed the state-of-the-art methods on Inspect dataset. While using the anti-patterns, the one interesting evidence we found is that the precision is increased significantly while the recall is reduced slightly. Also, the experiments showed that the modified dependency graph provided clearly the best performances by comparing with other traditional graphs including co-occurrence graph and basic dependency graph. Therefore, the modifications on the basic dependency graph has a significant effect.

As a future work, we will try to collect the anti-patterns for various domains. If we have many powerful anti-patterns, we can prove that the combination of anti-pattern and the graph-based term ranking model is the state-of-art method for keyphrase extraction area. Thus, we plan to investigate precisely the performances of the dependency graph and the co-occurrence graph for applying it to the classification task as [28].

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. 2017R1A2B4010826) and also was supported by the National Natural Science Foundation of China (No. 61702324) in People's Republic of China.

References

[1] T. Munkhdalai, M. Li, K. Batsuren, H. A. Park, N. H. Choi, and K. H. Ryu, "Incorporating domain knowledge in chemical and biomedical named entity recognition with word representations," *Journal of Cheminformatics*, vol. 7(Suppl 1), article no. S9, 2015.

[2] K. H. Ryu, M. Li, and I. Musa, "Biomedical text mining: an overview and an examplary application," in *Proceedings of International Conference on Information and Convergence Technology for Smart Society (ICICTS)*, Bangkok, Thailand, 2015.

[3] T. Munkhdalai, M. Li, K. Batsuren, and K. H. Ryu, "Towards a unified named entity recognition system," on *Proceedings of the International Joint Conference on Biomedical Engineering Systems and Technologies*, Lisbon, Portugal, 2015, pp. 251-255.

[4] M. Li, T. Munkhdalai, X. Yu, and K. H. Ryu, "A novel approach for protein-named entity recognition and protein-protein interaction extraction," *Mathematical Problems in Engineering*, vol. 2015, article no. 942435, 2015.

[5] E. Batbaatar, T. Munkhdalai, A. Nasridinov, O. E. Namsrai, and K. H. Ryu, "Incorporating domain knowledge in chemical named entity recognition using deep learning," in *Proceedings of 2016 International Conference on Information, System and Convergence Application*, 2016.

[6] P. D. Turney, "Learning algorithms for keyphrase extraction," *Information Retrieval*, vol. 2, no. 4, pp. 303-336, 2000.

[7] K. Sarkar, M. Nasipuri, and S. Ghose, "Machine learning based keyphrase extraction: comparing decision trees, naïve Bayes, and artificial neural networks," *Journal of Information Processing Systems*, vol. 8, no. 4, pp. 693-712, 2012.

[8] J. Wang, H. Peng, and J. S. Hu, "Automatic keyphrases extraction from document using neural network," in *Advances in Machine Learning and Cybernetics*. Heidelberg: Springer, 2006, pp. 633-641.

[9] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "KEA: practical automated keyphrase extraction," in *Proceedings of the 4th ACM Conference on Digital Libraries*, Berkeley, CA, 1999, pp. 254-255.

[10] O. Medelyan and I. H. Witten, "Thesaurus based automatic keyphrase indexing," in *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries*, Chapel Hill, NC, 2006, pp. 296-297.

[11] R. Mihalcea and P. Tarau, "Textrank: bringing order into text," in *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain, 2004, pp. 404-411.

[12] S. N. Kim, O. Medelyan, M. Y. Kan, T. Baldwin, and L. P. Pingar, "SemEval-2010 Task 5: automatic keyphrase extraction from scientific," in *Proceedings of the 5th International Workshop on Semantic Evaluation*, Uppsala, Sweden, 2010, pp. 21-26.

[13] J. Liu and J. Wang, "Keyword extraction using language network," in *Proceedings of International Conference on Natural Language Processing and Knowledge Engineering*, Beijing, China, 2007 pp. 129-134.

[14] X. Wan and J. Xiao, "Single document keyphrase extraction using neighborhood knowledge," in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence,* Chicago, IL, 2008, pp. 855-860.

[15] Z. Liu, W. Huang, Y. Zheng, and M. Sun, "Automatic keyphrase extraction via topic decomposition, in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Cambridge, MA, 2010, pp. 366-376.

[16] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993-1022, 2003.

[17] W. X. Zhao, J. Jiang, J. He, Y. Song, P. Achananuparp, E. P. Lim, and X. Li, "Topical keyphrase extraction from twitter," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Portland, OR, 2011, pp. 379-388.

[18] A. Bellaachia and M. Al-Dhelaan, "Ne-rank: a novel graph-based keyphrase extraction in twitter," in *Proceedings of the 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology*, Macau, China, 2012, pp. 372-379.

[19] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, Sapporo, Japan, 2003, pp. 216-223.

[20] T. D. Nguyen and M. Y. Kan, "Keyphrase extraction in scientific publications," in *Proceedings of International Conference on Asian Digital Libraries*, Hanoi, Vietnam, 2007, pp. 317-326.

[21] J. Hipp, U. Guntzer, and G. Nakhaeizadeh, "Algorithms for association rule mining: a general survey and comparison," *ACM SIGKDD Explorations Newsletter*, vol. 2, no. 1, pp. 58-64, 2000.

[22] M. Rafiqul Islam and M. Rakibu Islam, "An improved keyword extraction method using graph based random walk model," in *Proceedings of the 11th International Conference on Computer and Information Technology,* Khulna, Bangladesh, 2008, pp. 225-229). IEEE.

[23] K. S. Hasan and V. Ng, "Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, Beijing, China, 2010, pp. 365-373.

[24] Z. Zhu, M. Li, L. Chen, Z. Yang, and S. Chen, "Combination of unsupervised keyphrase extraction algorithms," in *Proceedings of 2013 International Conference on Asian Language Processing (IALP),* Urumqi, China, 2013, pp. 33-36.

[25] D. Klein and C. D. Manning, "Accurate unlexicalized parsing," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, Sapporo, Japan, 2003, pp. 423-430.

[26] D. Klein and C. D. Manning, "Fast exact inference with a factored model for natural language parsing," in *Advances in Neural Information Processing Systems*, vol. 15, pp. 3-10, 2003.

[27] De Marneffe, B. MacCartney, and C. D. Manning, "Generating typed dependency parses from phrase structure parses," *Proceedings of LREC*, vol. 6, pp. 449-454, 2006.

[28] S. Hassan and C. Banea, "Random walk term weighting for improved text classification," in *Proceedings of 2006 Workshop on Graph-based Methods for Natural Language Processing*, New York, NY, 2006, pp. 53-60.

Khuyagbaatar Batsuren https://orcid.org/0000-0002-6819-5444

He received a M.S. degree at Database and Bioinformatics Laboratory, Chungbuk National University, Cheongju, Korea in 2015. He received the B.S. degree in Computer Science from National University of Mongolia in 2012. Currently, He is a PhD student at the Doctoral School of Information and Communication Technology, University of Trento, Italy, since 2015. His major research interests focus on language diversity, computational lexical semantics, and language acquisition.

Erdenebileg Batbaatar https://orcid.org/0000-0002-9724-8955

He received the B.S. degree in Software Engineering from National University of Mongolia, in 2013. He recently received an M.S. degree at Database and Bioinformatics Laboratory, Chungbuk National University, Cheongju, Korea. Currently, he is a candidate for the PhD degrees in computer science at Chungbuk National University. His research interests include databases, data mining, bioinformatics and biomedical science, unusually, chemical text mining, epigenetic and deep learning.

Tsendsuren Munkhdalai https://orcid.org/0000-0002-8783-4993

He is a postdoctoral associate at BioNLP group at University of Massachusetts, USA. He recently received his Ph.D. in biomedical information extraction and natural language processing from the Department of Computer Science at Chungbuk National University, Korea under the excellent supervision of Prof. Keun Ho Ryu. He is currently working at Department of Quantitative Health Sciences University of Massachusetts Medical School in USA as a Post-Doc. Research Associate. His research interest includes semi-supervised learning, representation learning, meta-learning and deep learning with applications to natural language understanding and (clinical/biomedical) information extraction.

Meijing Li https://orcid.org/0000-0003-3931-7905

She received the M.S. degree in Bio Information Technology and the Ph.D. degree in computer science from Chungbuk National University, Cheongju, Korea in 2010 and 2015. She received the B.S. degree in computer science from Dalian University, Dalian, China, in 2007. She worked at Chungbuk National University as Post-Doc. She is currently an Assistant Professor in College of Information Engineering, Shanghai Maritime University, Shanghai, China. Her research interests include data mining, information retrieval, database systems, bioinformatics, and biomedicine.

Oyun-Erdene Namsrai https://orcid.org/0000-0001-6895-6904

She received the Ph.D. degree from Chungbuk National University, Cheongju, South Korea in 2008. She is working as a professor at Department of Information and Computer Science, School of Engineering and Applied Sciences, National University of Mongolia. Her research interests include temporal database, data mining, data ware housing, advanced algorithm issues, and advanced database applications. She is also a Software Developer with good experience on the Database and Visual programming. She has worked for a project of NUM as a consultant/software developer of Library Automation System (LAS), and able to work on own initiative or as part of a team and can deal with administrative duties competently. She also worked BIT project of Chungbuk National University of Korea from the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning as a researcher/software developer.

Keun Ho Ryu https://orcid.org/0000-0003-0394-9054

He is a professor at Chungbuk National University and a leader of database and bioinformatics laboratory in Korea since 1986. He is also a vice president of Personalized Tumor Engineering Research Center. He received the Ph.D. in Computer Science/Engineering from Yonsei University, Korea in 1988. He was a captain in Korean Army for four years as ROTC. He worked at University of Arizona as Post-doc as well as research scientist in USA. He has also worked at Electronics & Telecommunications Research Institute in Korea, as senior researcher. He has served on numerous program committees including a demonstration co-chair of the VLDB, a panel and tutorial co-chair of the APWeb, a general co-chair of the FITAT/ISPM. He has published over 1,000 referred technical articles in various journals, international conferences, and books. His research interests are included in temporal databases, spatiotemporal database, temporal GIS, ubiquitous computing and stream data processing, knowledgebase information retrieval, database security, data mining, bioinformatics and biomedical science. He is a member of the IEEE as well as a member of the ACM since 1983.