Automatic Documents Annotation by Keyphrase Extraction in Digital Libraries using Taxonomy

Iram Fatima, Asad Masood Khattak, Young-Koo Lee, Sungyoung Lee

Department of Computer Engineering, Kyung Hee University, Korea
{iram.fatima, asad.masood, sylee}@oslab.khu.ac.kr, yklee@khu.ac.kr

Abstract. Keyphrases are useful for variety of purposes including: text clustering, classification, content-based retrieval, and automatic text summarization. A small amount of documents have author-assigned keyphrases. Manual assignment of the keyphrases to existing documents is a tedious task, therefore, automatic keyphrase extraction has been extensively used to organize documents. Existing automatic keyphrase extraction algorithms are limited in assigning semantically relevant keyphrases to documents. In this paper we have proposed a methodology to assign keyphrases to digital documents. Our approach exploits semantic relationships and hierarchical structure of the classification scheme to filter out irrelevant keyphrases suggested by Keyphrase Extraction Algorithm (KEA++). Experiments demonstrate that the refinement improves the precision of extracted keyphrases from 0.19% to 0.38% while maintains the same recall.

1 Introduction

Keyphrases precisely express the primary topics and theme of documents and are valuable for cataloging and classification [1,2]. A keyphrase is defined as a meaningful and significant expression consisting of a single word, e.g., information, or compound words, e.g., information retrieval. Manual assignment and extraction of keyphrases is resource expensive and time consuming. It requires a human indexer to read the document and select appropriate descriptors, according to defined cataloguing rules. Therefore, it stimulates the need for automatic extraction of keyphrases from digital documents in order to deliver their main contents.

Existing approaches for keyphrase generation include keyphrase assignment and keyphrase extraction [3, 4]. In keyphrase assignment keyphrases are selected from a predefined list of keyphrases, thesaurus or subject taxonomy (i.e., Wordnet, Agrovoc) [4]. While in later approach all words and phrases included in the document are potential keyphrases [5, 6]. Phrases are analyzed on the basis of intrinsic properties such as frequency, length, and other syntactic information. The quality of the generated keyphrases by the existing approaches has not been able to meet the required accuracy level of applications [7,8].

The extraction algorithm used in this paper, KEA++, applies a hybrid approach of keyphrase extraction and keyphrase assignment [7-9]. KEA++ combines advantages of both, while avoiding their shortcomings. It makes use of a domain specific taxonomy to assign relevant keyphrases to documents. Limitation of this approach is that output keyphrases contain some irrelevant information along with the relevant ones. For example, out of five keyphrases, two might fit well, while the remaining three have no semantic connection to the document (discussed later in the case study). The focus of this paper is to improve the semantic alignment procedure by exploiting different hierarchical levels of taxonomy. The proposed methodology is a novel approach of refinement, and comprises two major processes: (a) extraction and (b) refinement. KEA++ (Key Phrase Extraction Algorithm) [7-9] has been adopted for extracting keyphrases. The refinement process refines the result set of keyphrases returned by KEA++ using different levels of taxonomy. It detects the semantic keyphrases that are closer to human intuition as compared to KEA++. Experiments have been performed on dataset of 100 documents collected from the Journal of

Universal Computer Science (JUCS[1]). Experimental results show better precision (0.45) of proposed methodology in comparison to the precision (0.22) of KEA++ at the third level of the ACM Computing Classification[2] while maintaining the same recall.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 explains the proposed methodology of automatic keyphrase refinement. Results from JUCS dataset are given in Section 4. Conclusion together with possible future work discusses in section 5.

2. Related Work

Keyphrase extraction is a process to gather useful information from documents that help in describing the true content of the documents. KEA [10,11] identifies candidate phrases from textual sequences defined by orthogonal boundaries and extract relevant ones based two feature values for each candidate: the (1) TF x IDF measure, and (2) the distance from the beginning of the document to the first occurrence of a phrase. GenEx uses the genetic algorithm which is based on 12 numeric parameters and flags [12, 13]. This keyphrase extraction algorithm has two main components: (1) Genitor and (2) Extractor. Genitor is applied to determine the best parameter settings from the training data. Extractor combines a set of symbolic heuristics to create a ranked list of keyphrases.

The next approach is to use Natural Language Processing (NLP) tools in addition to machine learning, therefore the A.Hulth algorithm [14] compares different methods to extract candidate words and phrases like NP chunking, Parts of Speech (PoS) pattern matching, and trivial n-gram extraction. Candidates are filtered on the basis of four features: (1) term frequency, (2) inverse document frequency, (3) position of the first occurrence, and (4) PoS tag. In keyphrase assignment, a predefined set of keyphrases called the controlled vocabulary is used to describe the characteristics of documents in order to find the appropriate keyphrases, rather than individual phrases within them [15,16,17].

KEA++ is a hybrid of keyphrase assignment and keyphrase extraction [7-9]. It can involve taxonomy in extracting keyphrases from documents. Keyphrase selection is based on the computation of the naïve based statistical model and relations within the taxonomy. KEA++ takes a document, along with the taxonomy, as input for keyphrase extraction. KEA++ extracts terms from the documents (i.e., not explicitly mentioned in the document but existing in the taxonomy) by relating them to the terms of the taxonomy. The results of controlled indexing are highly affected by the parameter settings [18, 19]. The major parameters affecting the results are: vocabulary name, vocabulary format, vocabulary encoding, max. length of phrases, min. length of phrase, min. occurrence and no of extracted keyphrases.

The quality of the generated keyphrases by existing algorithms is inadequate, and they need to be improved for their applicability in real world applications. Some of the existing approaches use the taxonomy's hierarchy, yet it can be utilized in a more effective way. The results of KEA++ returned relevant keyphrases along with noise. In order to filter out the irrelevant information from the returned keyphrases of KEA++, there is a need for some refinement methodology that reduces the noise from the returned results of KEA++.

3. Proposed Methodology

Proposed methodology processes the returned results of KEA++ [7-9] by exploiting different hierarchical level of taxonomy. It involves two main steps: (a) extraction and (b) refinement. Extraction is a prerequisite of refinement. The refinement process is based on refinement rules. Refinement rules are applied to the set of keyphrases returned by KEA++ after the customized

[1] http://www.jucs.org/jucs_16_14

[2] http://www.acm.org/about/class/1998/

parameter settings. We set the vocabulary name parameter to the ACM computing classification in the SKOS format using UTF-8 encoding. It is used for the implementation and testing purpose of our algorithm, while our contribution is adoptable for other classification systems. The remaining refinement parameter settings of KEA++ are: *Max. Length of Phrases:* After analyzing the ACM Computing Classification, we set the value of this parameter to five words. *Min. Length of Phrase:* The minimum phrase length is one word in ACM taxonomy (i.e., hardware), which is the top level. We set the value of this parameter to two words because setting the value to one word provides many irrelevant keyphrases. *Min. Occurrence:* KEA++ recommends two words for this parameter in long documents. *No. of Extracted Keyphrases:* If the value of this parameter is less than ten words, for example four words, then KEA++ returns the top four keyphrases from the results it computes. These keyphrases might not be relevant. Other parameter settings as mentioned above can affect the results of this parameter.

3.1. Refinement Rules

These rules emphasize the importance of different levels/facts and their associated semantic relation in the training and semantic keyphrase extraction process. The basic idea filtered out semantic keyphrases according to the most related levels and available relations within different levels of taxonomy applying following rules:

Rule I: Adopting the Training Level: The training level is the hierarchical level of the taxonomy, adjusted for manually extracted keyphrases in documents. We adopt the KEA++ training level during the refinement process to extract the refined set of semantic keyphrases. The effective usage of the remaining rules depends on the accurate value of the training level of the taxonomy.

Rule II: Preserving the Training Level Keyphrases: We only preserve keyphrases aligned on the training level. KEA++ results have keyphrases that belong to different levels in the taxonomy. In addition to training level keyphrases, it might have upper level keyphrases and lower level keyphrases which do not contain information as relevant as the training level keyphrases. This rule selects the most relevant keyphrases from the resulting set of KEA++.

Rule III: Stemming the Lower Level General Keyphrases: In the ACM Computing Classification, there is the *general* category of keyphrases on each level of the hierarchy. If a keyphrase is aligned on a lower level than the training level (e.g., C.2.3.0), and associated with the general category in the lower level, then we stem the lower level keyphrase to its training level (e.g., C.2.3) keyphrases. This rule helps in extracting the maximum possible information from the lower level keyphrases in the presence of training level keyphrases.

Rule IV: Preserving the Lower Level Keyphrases: If the result set of KEA++ contains no training level keyphrases, then we preserve the lower level keyphrases from the result set of KEA++. This rule identifies the relevant keyphrases in the absence of training level keyphrases. In this case, lower level keyphrases represent the documents alignment on more accurate nodes, which belong to more specific keyphrases in the taxonomy.

Rule V: Identifying and Preserving the Training Level Equivalent Keyphrase: Different keyphrases aligned to separate categories of the ACM Computing Classification can be semantically equivalent, e.g., *Control Structures and Microprogramming* (B.1) is equivalent to *Language Classifications* (D.3.2). The procedure is carried out by first identifying the training level keyphrases from the set of upper level keyphrases. If the upper level has equivalent keyphrases of the training level, then preserve the training level keyphrase in the refined result set and discard the upper level keyphrase. But before discarding them, it helps in preserving any training level keyphrases equivalent to an upper level keyphrase.

Rule VI: Removing Redundant Keyphrases: After stemming the lower level general keyphrases and identifying and preserving the training level equivalent keyphrases, the result might contain

redundant keyphrases (i.e., C.2.3, C.2.3, D.4.5). Remove the redundant keyphrases from the set of refined keyphrases (i.e., C.2.3, D.4.5).

Input: **Training:**
(a) Set the parameters of KEA++ by keeping in view the keyphrase length in the taxonomy and documents type
(b) Documents along with their keyphrase and taxonomy
Dataset for Extraction:
(a) Documents with unknown keyphrases

Output: Set of refined keyphrases

```
1: TrainLevel ← KEA++ TrainLevel //(Rule I)
2: resultSet [] ← returned keyphrases by KEA++[]
3: resultSet [] ← level labels (Resultset [])
4: for resultSet[] <> empty do
5:     if (resultSet(training level)) then
6:         if (keyphrase level = lower level keyphrases) then
7:             processSet[] = preserving lower level keyphrases //Rule II
8:         else
9:             processSet[] ← identifying and preserving training level equivalent //Rule V
10:            processSet[] ← remove redundant keyphrases //Rule VI
11:            refineSet[] ← processSet[]
12:        end if
13:    else
14:        if (keyphrase level = training level) then
15:            refineSet[] ← processSet[]
16:        else
17:            if (keyphrase level = upper level) then
18:                processSet[] ← identifying and preserving training level
                                  equivalent keyphrases //Rule V
19:            else
20:                processSet[] ← stemming lower level general keyphrases //Rule III
21:            end if
22:            processSet[] ← remove redundant keyphrases //Rule VI
23:            refineSet[] ← processSet[]
24:        end if
25:    end if
26: end for
27: return refineSet[] //refine result set of semantic keyphrases.
```

Algorithm 1: Refinement Algorithm

3.2 Refinement Algorithm

The algorithm describes the flow of refinement rules that is illustrated in Algorithm 1. Extraction of the semantic keyphrases is the essential requirement of the refinement process. First of all parameters of the extraction algorithm KEA++ are set with respect to keyphrases' length in the taxonomy and length of the documents. Secondly train KEA++ on the set of documents using taxonomy. Then apply KEA++ on actual documents (data). Adopting the training level for the refinement rules has primary importance because it guides the remaining rules in their process. The keyphrases returned by KEA++ is processed to get its level label in the taxonomy. Indentify level labels is required before applying the refinement rules because they represent the hierarchical order of the keyphrases as described in steps 1 to 3 of Algorithm 1. If the KEA++ result has training level keyphrase then these training level keyphrases are retained in the result set as shown in steps 5 to 12 of Algorithm 1. Lower level keyphrases are stemmed to their training level keyphrases and kept in the result set if they are associated with the general category at the lower level in taxonomy. Otherwise lower level keyphrases are discarded. Upper level keyphrases are handled according to Rule-V and discarded after indentifying and preserving their equivalent keyphrases from taxonomy which belong to the same level of training level keyphrases. If the initial result does not contain any training level keyphrases then lower level keyphrases of the result are preserved and added in the final refined result. Upper level keyphrases in the initial result are handled according to Rule-V and discarded after indentifying and preserving their equivalent keyphrases from taxonomy which belong to the same level of training level keyphrases. This process is executed from steps 13 to 21 of the algorithm. Finally redundant keyphrases are removed from the final refined set of keyphrases.

3.3. Case Study

To focus more on the refinement process proposed in this paper, a case study is presented in which training models is trained on third level of ACM Computing Classification. Table 1 illustrates the information about the documents used in the evaluation (available on the web). The first column of Table 2 represents the semantic keyphrases returned by KEA++ after applying the parameters proposed in the refinement process.

Table. 1. Sample document

Title	Passive Estimation of Quality of Experience
Identification Key:	JUCS, Vol. 14, Issue 5, year 2008
Manual Alignment:	C.2.3 (Network Operations), C.4 (Performance of System)

Extracted semantic keyphrases align the document on five nodes of the ACM Computing Classification, while document is manually aligned on two nodes. Extracted keyphrases include both relevant and noise/irrelevant keyphrases.

Table 2: Results of the refinement algorithm

KEA++ Results	**Level Label**	**Refined Results**
Network Management	C.2.3.0	
Distributed Functions	G.3.2	G.3.2
Network Operations	C.2.3	C.2.3
Approximate Methods	I.4.2.1	
Data Structures	E.1	

We select the level labels of the keyphrases from the ACM Computing Classification as shown in the second column of Table 2. Keyphrases with their associated level labels show alignment of the document with different depths in the ACM Computing Classification. The refined results are calculated after applying the rules of the refinement algorithm. The results are quite improved in that they include an exact match with one relevant keyphrase. The whole process of refinement involves following steps. After identifying the level labels of keyphrases, the refinement algorithm checks whether the level labels of the keyphrases contain the training level. If it has the training level (C.3.2 and G.3.2), then it preserves these training level keyphrases. Now it identifies whether the result set has upper level keyphrases. As it has upper level keyphrases (E.1), the rule V is applicable here. The existence of low level keyphrases belongs to a general category (C.2.3.0), so it is stemmed to the training level (C.2.3) keyphrase. In the end, the algorithm removes the redundant keyphrases (C.2.3, C.2.3) and declares the result set as the final refined result set (C.2.3, G.3.2), as shown in the third column of Table 2. This case study explains that lower level keyphrase extraction is not significant when compared to training level keyphrases.

4. Results and Evaluation

In this section, the results of manual annotation, KEA++, and the proposed refinement algorithm are compared. The precision of the refinement algorithm is evaluated on various hierarchical levels as provided in the manual annotations of the dataset. Evaluation contains both (a) *keyphrase based evaluation* and (b) *document based* evaluation. We modeled the ACM Computing Classification in SKOS format[3]. Dataset[4] from (JUCS) has a key file along with the document file which contains manually assigned keyphrases. Two experiments have been performed on dataset of 100

[3] http://uclab.khu.ac.kr/ext/ACM_Computing_Classification.rar
[4] http://uclab.khu.ac.kr/ext/Dataset_Journal_of_Universal_Computer_Science(JUCS).rar

documents. Experiment I is based on 65 documents, 50 for training and 15 for testing purpose. Experiment II is performed on 100 documents, 70 documents were used for training and 30 were used for testing. To evaluate the precision of the system, documents which were used as testing documents in the first experiment were replaced with the training documents.

4.1 Keyphrase based Evaluation

It is used to estimate the performance in terms of returned keyphrases. It is further categorized as (a) keyphrases returned per average number of documents, and (b) total returned keyphrases. In the former category, we compare results among (1) manual annotation, (2) KEA++, and (3) the refinement rules. Fig. 1 (a) and (b) show that the number of KEA++ returned keyphrases lies between 0 to 7, 0 to 13, respectively. Manual annotation of the same documents varies from 1 to 4, 1 to 5, respectively. Keyphrases returned by the refinement algorithm range from 0 to 3. It shows that the refinement algorithm reduces the number of keyphrases returned against an average number of documents as compared to KEA++ and closer to the manual annotation. However, it does not affect the precision of correctly aligned documents, as shown in the next subsection.

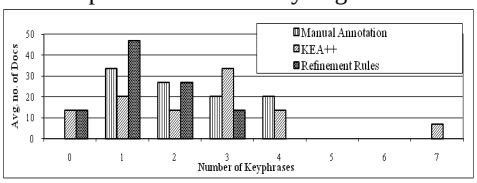

Fig. 1(a). Keyphrases returned per average no of documents of experiment I

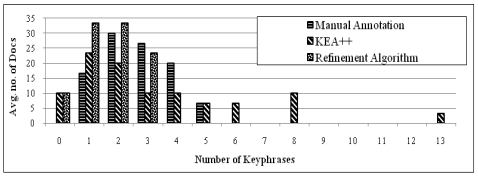

Fig.1 (b). Keyphrases returned per average no of documents of experiment II

Total returned keyphrases compares the precision, and recall of both KEA++ and the refinement rules. Fig. 2 (a) and (b) show the precision of both algorithms. In the case of KEA++, the precision 0.19 and 0.23, while the refinement algorithm shows more precise results, with values of 0.38 and 0.45, respectively. The recall comparison is illustrated in Fig. 3 (a) and (b) and the recall of KEA++ and the refinement algorithm are the same in the case of both experiments.

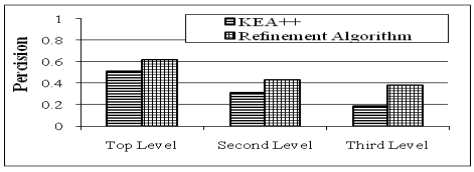

Fig. 2 (a). Precision against total keyphrases returned of experiment I

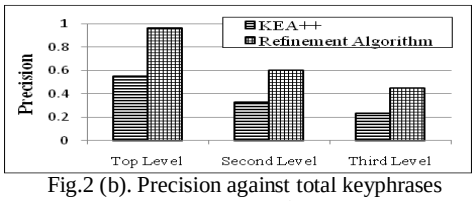

Fig.2 (b). Precision against total keyphrases returned of experiment II

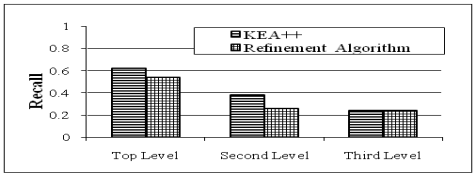

Fig. 3 (a). Recall against total keyphrases returned of experiment I

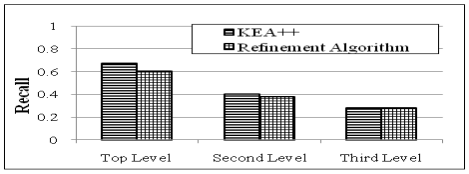

Fig. 3 (b). Recall against total keyphrases returned of experiment II

4.2 Document based evaluation

The document based evaluation verifies the performance of both algorithms against correctly aligned documents. We do not consider the recall calculation as the number of documents is the same in both cases. This evaluation criterion is further categorized as (a) the totally matched result

and (b) the approximate matched result. The totally matched result contains all of the manually annotated keyphrases of the particular document, while the approximate matched result comprises a subset of manually annotated keyphrases of the particular document.

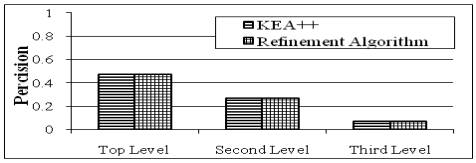

Fig. 4 (a). Precision of totally matched results of experiment I

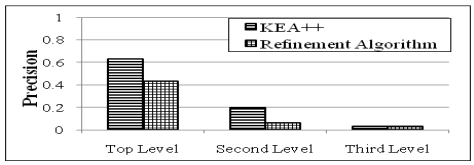

Fig. 4 (b). Precision of totally matched results of experiment II

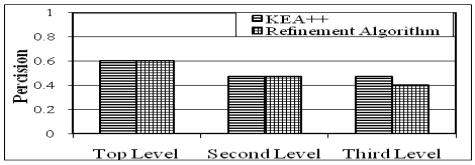

Fig. 5 (a). Precision of approximate matched results of experiment I

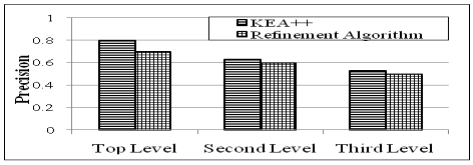

Fig. 5 (b). Precision of approximate matched results of experiment II

The totally matched result is a more conservative approach because it ignores the approximately aligned documents. Figure 4 (a) and (b) illustrate the precision for the totally matched results, the precisions is same on the third level of taxonomy. Furthermore, the refinement rules returned a reduced number of keyphrases. Figure 5 (a) and (b) show the precision of both approaches for the approximate matched results. Due to the reduced number of keyphrases per average number of documents, the precision is comparatively lower on the third level of the taxonomy.

Table 3. Precision, recall, and F-measure statistics

Experiments	**Results of [9].**	**Experiment. I**		**Experiment. II**	
Algorithms	KEA++	KEA++	Refined Results	KEA++	Refined Results
Documents	200	65		100	
Avg. # of Manual Annotation	5.4	2.27		2.35	
Precision	0.28	0.19	0.38	0.23	0.45
Recall	0.26	0.24	0.24	0.28	0.28
F-measure	0.25	0.21	0.29	0.25	0.34

Table 3 shows precision, recall, and F-measure statistics, in results of [9], the precision, recall, and F-measure of KEA++ are 0.28, 0.26, and 0.25, respectively, while the average number of manual annotation is 5.4 per document in the dataset of 200 documents. The precision, recall, and F-measure of KEA++ on our experiments are different. Obviously, the precision and recall is affected by a change in the number of documents in the dataset, and the average number of manual annotations per document in each dataset. In the case of the refinement algorithm, the precision has been improved in all performed experiments while the recall is same as shown in Table 3.

5. Conclusion and Future Work

An exponential growth of electronic documents requires extraction of keyphrases for semantic alignment. The proposed algorithm processes the results returned by KEA++ and removes irrelevant keyphrases by exploiting the hierarchical structure of taxonomy to achieve better accuracy. The refinement algorithm provides the functional flow to the refinement rules. To

evaluate the methodology, JUCS dataset is used in different experiments and results show obvious improvement in the precision as compared to KEA++, while maintaining the same recall. Currently the focus was on a single training level in applying the refinement algorithm so in future this refinement algorithm can be extended to involve more than one training level while executing the refinement algorithm in order to achieve more accurate results and to make the methodology scalable.

6. References

1. Liu, Z. Li, P. Zheng, Y. Sun, M.: Clustering to Find Exemplar Terms for Keyphrase Extraction. In: Empirical Methods on Natural Language Processing, pp. 257--266, ACL Proceedings, Singapore (2009)
2. Frank, E. G. Paynter, Witten, W. I. H. Gutwin, C. Nevill-manning, C. G.: Domain specific keyphrase extraction. In: Sixteenth International Joint Conference on Artificial Intelligence, pp. 668–673 Sweden (1999)
3. O. Roberto, D. Pinto, M. Tovar: BUAP: An Unsupervised Approach to Automatic Keyphrase Extraction. In: 23rd 5th International Workshop on Semantic Evaluation ACL, pp. 174--177, Sweden (2010)
4. Barker, K. Cornacchia, N.: Using noun phrase heads to extract document keyphrases, Conference on Artificial Intelligence, pp. 40--52, Canadian (2000)
5. Jacquemin, C. Bourigault, D.: Term extraction and automatic indexing, The Oxford Handbook of Computational Linguistics, pp. 559--616, (2003)
6. Jones, S. Paynter, G.: Automatic extraction of document keyprases for use in digital libraries: evaluation and applications. J of the American Society for Information Science and Technology, vol. 53(8), pp. 653--677, (2002)
7. Medelyan, O. Witten, H. I.: Thesaurus Based Automatic Keyphrase Indexing. Joint Conference on Digital Libraries, pp. 296--297, USA (2006)
8. Medelyan, O. Witten, H. I.: Semantically Enhanced Automatic Keyphrase Indexing, WiML, (2006)
9. Medelyan, O., Witten I. H.: Thesaurus-based index term extraction for agricultural documents. In: 6th Agricultural Ontology Service workshop at EFITA, Portugal. (2005)
10. Witten, I. H. Paynter, G. W. Frank, E. Gutwin, C. Nevill-Manning, C.G.: KEA: Practical automatic keyphrase extraction, pp. 254--256 (1999)
11. Witten, H. I. Paynter, G. W. Frank, E. Gutwin: Kea: Practical automatic keyphrase extraction, Design and Usability of Digital Libraries, pp. 129--152, (2005)
12. Turney, P. D.: Learning algorithms for keyphrase extraction. pp. 303--336, Information Retrieval, (2003)
13. Turney, P.D.: Coherent keyphrase extraction via web mining. J. of the American Society for Information Science and Technology, pp. 434--439 (2003)
14. Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge, Empirical Methods in Natural Language Processing, pp. 216--223, (2003)
15. Kim, S. N. Kan, M.Y.: Re-examining automatic keyphrase extraction approaches in scientific articles, MWE, pp. 9--16, (2009)
16. Barker K. and Cornacchia N., Using noun phrase heads to extract document keyphrases, pp. 40--52, Canadian (2000)
17. Paice, C. Black, W.: A three-pronged approach to the extraction of key terms and semantic roles, Recent Advances in Natural Language Processing, pp. 357--363 (2003)
18. I. Fatima, S. Khan, K. Latif, Refinement Methodology for Automatic Document Alignment Using Taxonomy in Digital Libraries, ICSC, pp. 281--286, USA (2009)
19. Samhaa, E.R. Ahmed, B.R. KP-Miner.: A keyphrase extraction system for English and Arabic documents, vol. 34(1), pp. 132-144, (2009)