A New Domain Independent Keyphrase Extraction System *

Nirmala Pudota, Antonina Dattolo, Andrea Baruzzo, Carlo Tasso

Artificial Intelligence Lab
Department of Mathematics and Computer Science
University of Udine, Italy
{nirmala.pudota,antonina.dattolo,andrea.baruzzo,carlo.tasso}@dimi.uniud.it

Abstract. In this paper we present a keyphrase extraction system that can extract potential phrases from a single document in an unsupervised, domain-independent way. We extract word n-grams from input document. We incorporate linguistic knowledge (i.e., part-of-speech tags), and statistical information (i.e., frequency, position, lifespan) of each n-gram in defining candidate phrases and their respective feature sets. The proposed approach can be applied to any document, however, in order to know the effectiveness of the system for digital libraries, we have carried out the evaluation on a set of scientific documents, and compared our results with current keyphrase extraction systems.

1 Introduction

A *keyphrase* is a short phrase (typically it contains one to three words) that provides a key idea of a document. A *keyphrase list* is a short list of keyphrases (typically five to fifteen phrases) that reflects the content of a single document, capturing the main topics discussed and providing a brief summary of its content. If every document is attached with keyphrases, a user can choose easily which documents to read and/or understand the relationships among documents. Document keyphrases are used successfully in *Information Retrieval (IR)* and *Natural Language Processing (NLP)* tasks, such as document indexing [9], clustering [10], classification [14], and summarization [4]. Among all of them, document indexing is one important application of automatic keyphrase generation in digital libraries, where the major part of publications usually are not associated with keyphrases. Furthermore, keyphrases are well exploited for other tasks such as thesaurus creation [13], subject metadata enrichment [25], query expansion [21], and automatic tagging [18].

Despite of having many applications, only a small percent of documents have keyphrases assigned to them. For instance, in digital libraries, authors assign keyphrases to their documents when they are instructed to do so [9], other digital content, like news or magazine articles, usually do not have keyphrases

* The authors acknowledge the financial support of the Italian Ministry of Education, University and Research (MIUR) within the FIRB project number RBIN04M8S8.

since it is neither mandatory nor necessary for the document authors to provide keyphrases. Manually assigning keyphrases to documents is tedious, time-consuming, and as well as expensive. Therefore, automatic methods that generate keyphrases for a given document are beneficial.

Witten et al. [24] defined two fundamental approaches for automatic keyphrase generation:

1. *Keyphrase assignment*: in this case, the set of possible keyphrases is limited to a predefined vocabulary of terms (e.g., subject headings, classification schemes, thesaurus). The task is to classify documents based on the content into different keyphrase classes that correspond to the terms of a pre-defined list. In this process, the document can be associated with keyphrases constituted by words (or n-grams) that are not contained in the document.
2. *Keyphrase extraction*: in contrast to the previous case, keyphrase extraction selects the most indicative phrases present in the input document. In this process, selection of keyphrases does not depend on any vocabulary and such phrases are *supposed to be available in the document itself.*

In this paper, we concentrate on the keyphrase extraction problem leaving the more general task of keyphrase assignment. The work presented here is part of a wide research project PIRATES (Personalized Intelligent tag Recommendation and Annotation TEStbed) [2, 3, 7], a framework for personalized content retrieval, annotation, and classification. Using an integrated set of tools, PIRATES framework lets the users experiment, customize, and personalize the way they retrieve, filter, and organize the large amount of information available on the Web. Furthermore, the framework undertakes a novel approach that automates typical manual tasks such as content annotation and tagging, by means of personalized tag recommendations and other forms of textual annotations (e.g., keyphrases).

The rest of this paper is organized as follows: Section 2 introduces the related work. The proposed domain independent keyphrase extraction system is described in detail in Section 3. Empirical evaluation is presented in Section 4 and finally we conclude the paper in Section 5.

2 Related Work

Keyphrase extraction methods usually work in two stages: (i) a *candidate identification* stage, identifies all possible phrases from the document and (ii) a *selection* stage, selects only few candidate phrases as keyphrases. Existing methods for keyphrase extraction can be divided into supervised and unsupervised approaches, illustrated in the following:

A. The *supervised approach* treats the problem as a classification task. In this approach, a model is constructed by using training documents, that have already keyphrases assigned (by humans) to them. This model is applied in order to select keyphrases from previously unseen documents. Turney

(developer of *Extractor*[1]) [22] is the first one who formulated keyphrase extraction as a supervised learning problem. According to him, all phrases in a document are potential keyphrases, but only phrases that match with human assigned ones are correct keyphrases. Turney uses a set of parametric heuristic rules and a genetic algorithm for extraction. Another notable keyphrase extraction system is *KEA* (Keyphrase Extraction Algorithm) [24]; it builds a classifier based on the Bayes' theorem using training documents, and then it uses the classifier to extract keyphrases from new documents. In the training and extraction, KEA analyzes the input document depending on orthographic boundaries (such as punctuation marks, newlines) in order to find candidate phrases. In KEA two features are exploited: tf×idf (term frequency × inverse document frequency) and first occurrence of the term.

Hulth [11] introduces linguistic knowledge (i.e., *part-of-speech (pos) tags*) in determining candidate sets: 56 potential *pos-patterns* are used by Hulth in identifying candidate phrases in the text. The experimentation carried out by Hulth has shown that, using a *pos tag* as a feature in candidate selection, a significant improvement of the keyphrase extraction results can be achieved. Another system that relies on linguistic features is LAKE (Learning Algorithm for Keyphrase Extraction) [8]: it exploits linguistic knowledge for candidate identification and it applies a Naive Bayes classifier in the final keyphrase selection.

All the above systems need a training data in small or large extent in order to construct an extraction system. However, acquiring training data with known keyphrases is not always feasible and human assignment is time-consuming. Furthermore, a model that is trained on a specific domain, does not always yield to better classification results in other domains.

B. The *unsupervised approach*[2] eliminates the need of training data. It selects a general set of candidate phrases from the given document, and it uses some ranking strategy to select the most important candidates as keyphrases for the document.

   Barker and Cornacchia [1] extract noun phrases from a document and ranks them by using simple heuristics based on their length, frequency, and the frequency of their head noun.

   In [5], Bracewell et al. extract noun phrases from a document, and then cluster the terms which share the same noun term. The clusters are ranked based on term and noun phrase frequencies. Finally, top-n ranked clusters are selected as keyphrases for the document.

   In [17], Liu et al. propose another unsupervised method, that extracts keyphrases by using clustering techniques which assure that the document is semantically covered by these terms. Another unsupervised method that utilizes

[1] http://www.extractor.com/

[2] Note that unsupervised approaches might use tools like POS taggers which rely on supervised approaches. However, as such tools are usually already available for most languages, we consider an approach is unsupervised if it does not make use of any training documents that have already keyphrases assigned to them.

document cluster information to extract keyphrases from a single document is presented in [23].
Employing graph-based ranking methods for keyphrase extraction is another widely used unsupervised approach, exploited for example in [16]. In such methods, a document is represented as a term graph based on term relatedness, and then a graph-based ranking model algorithm (similar to the PageRank algorithm [6]) is applied to assign scores to each term. Term relatedness is approximated in between terms that co-occur each other within a pre-defined window size.
Keyphrase extraction systems that are developed by following unsupervised approach are in general domain independent since they are not constrained by any specific training documents.

3 Domain Independent Keyphrase Extraction (DIKpE) System Description

Domain independent keyphrase extraction approach, which doesn't enforce any training data has many applications. For instance, it can be useful for a user who wants to know quickly the content of a new Web page, or who wants to know the main claim of a paper at hand. In such cases, keyphrase extraction approach that can be applied without a corpus[3] of the same kind of documents is very useful. Simple term frequency is sometimes sufficient to know the document overview; however, more powerful techniques are desirable.

Our approach is applied to any document without the need of a corpus. It is solely based on a single document. In the following, we provide a detailed description of our approach. The general workflow in DIKpE system is shown in Figure 1 and is illustrated in detail in the following subsections 3.1, 3.2, and 3.3. We follow three main steps: (i) extract candidate phrases from the document (ii) calculate feature values for candidates (iii) compute a score for each candidate phrase from its feature values and rank the candidate phrases based on their respective scores, in such a way, highest ranked phrases being assigned as keyphrases.

3.1 Step1: Candidate Phrase Extraction

This step is divided in the following substeps:

- **Format conversion**. We assume that the input document can be in any format (e.g., *pdf*), and as our approach only deals with textual input, our system first exploits document converters to extract the text from the given input document.
- **Cleaning and Sentence delimiting**. The plain text form is then processed to delimit sentences, following the assumption that no keyphrase parts are located simultaneously in two sentences. Separating sentences by inserting

[3] A collection of documents.

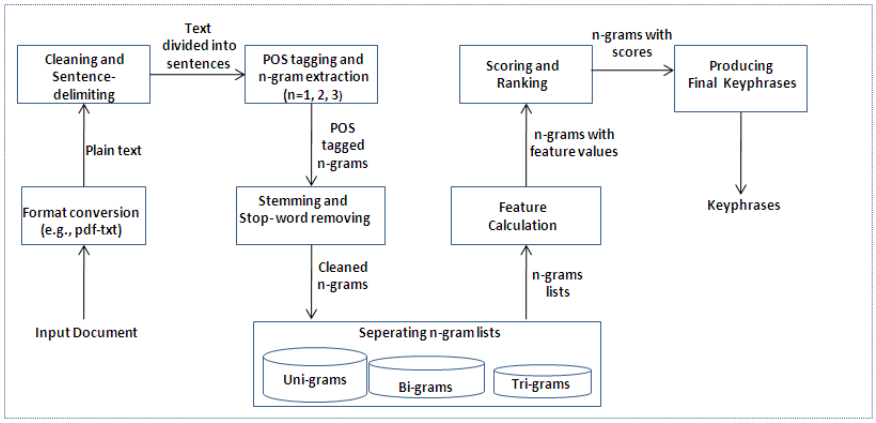

Figure 1. *Workflow in DIKpE system*

a sentence boundary is the main aim of this step. The result of this step is a set of sentences each containing a sequence of tokens, bounded by the sentence delimiter.

- **POS tagging and n-gram extraction**. We assign a pos tag (noun, adjective, verb etc.) to each token in the cleaned text, by using Stanford log-linear part-of-speech tagger[4]. The Stanford pos tagger uses 36 types[5] of pos tags (for the documents written in Italian, an Italian pos tagger developed using n-gram model trained on the La Repubblica corpus[6] is utilized.). The assigned pos tags are later utilized for filtering candidate phrases and in calculating pos value feature. The next step in our procedure is to extract n-grams. We have observed that in the dataset utilized for the experimentation, phrases that are constituted by more than 3 words are rarely assigned as keyphrases, so, in our process, we set the value of 'n' to the maximum value 3. We extract all possible subsequences of phrases up to 3 words (uni-grams, bi-grams, and tri-grams).
- **Stemming and Stopword removing**. From the extracted n-grams, we remove all phrases[7] that start and/or end with a stopword and phrases containing the sentence delimiter. Partial stemming (i.e., unifying the plural forms and singular forms which mean essentially the same thing) is performed using the first step of Porter stemmer algorithm [20]. To reduce the size of the candidate phrase set, we have filtered out some candidate phrases by using their pos tagging information. Uni-grams that are not labeled as noun, adjective, and verb are filtered out. For bi-grams and tri-grams, only

[4] http://nlp.stanford.edu/software/tagger.shtml.
[5] pos tagging follows the Penn Treebank tagging scheme.
[6] http://dev.sslmit.unibo.it/corpora/corpus.php?path=&name=Repubblica
[7] In our use of this term, we mean any n-gram (n=1,2,3) phrase.

pos-patterns defined by Justeson and Katz [12] and other patterns that include adjective and verb forms are considered.
- **Separating n-gram lists**. Generally, in a document, uni-grams are more frequent than bi-grams, and bi-grams are more frequent than tri-grams and so on. In the calculation of phrase frequency (explained in Section 3.2) feature, this shows a bias towards n-grams which are having small value of 'n'. In order to solve this problem, we have separated n-grams of different lengths (n=1, n=2, and n=3) and arranged them in three different lists. These lists are treated separately in calculation of feature sets and in final keyphrase selection. As a result of step 1, we obtain a separate list of uni-gram, bi-gram, and tri-gram candidate phrases (with corresponding pos tags) per document after the proper stemming and stopword removal explained above.

3.2 Step2: Feature Calculation

The candidate phrase extraction step is followed by a feature calculation step that characterizes each candidate phrase by statistical and linguistic properties. Five features for each candidate phrase are computed; these are: phrase frequency, pos value, phrase depth, phrase last occurrence, and phrase lifespan, illustrated in the following.

- ***phrase frequency***: this feature is same as the classical term frequency (tf) metric. But Instead of calculating it with respect to the whole length of the document, we compute it with respect to each n-gram list. With a separate list for each n-gram in hand, the phrase frequency for phrase P in a list L is:

$$frequency(P, L) = \frac{freq(P, L)}{size(L)},$$

  where:
  - $freq(P, L)$ is the number of times P occurs in L;
  - $size(L)$ is the total number of phrases included in L.
- ***pos value***: as described in [1], most author-assigned keyphrases for a document turn out to be noun phrases. For this reason, in our approach, we stress the presence of a noun in a candidate phrase while computing a pos value for the phrase. A pos value is assigned to each phrase by calculating the number of nouns (singular or plural) normalizing it by the total number of terms in the phrase. For instance, in a tri-gram phrase, if all tokens are noun forms, then the pos value of the phrase is 1, if two tokens are noun forms, then the pos value is 0.66, and if one noun is present, the value is 0.33. All remaining phrases which do not include at least one noun form are assigned the pos value 0.25. The same strategy is followed for bi-gram and uni-gram phrases.
- ***phrase depth***: this feature reflects the belief that important phrases often appear in the initial part of the document especially in news articles and scientific publications (e.g., *abstract, introduction*). We compute the position

in the document where the phrase first appears. The phrase depth value for phrase P in a document D is:

$$depth(P, D) = 1 - [\frac{first\_index(P)}{size(D)}],$$

where $first\_index(P)$ is the number of words preceding the phrase's first appearance; $size(D)$ is the total number of words in D.
The result is a number between 0 and 1. Highest values represent the presence of a phrase at the very beginning of the document. For instance, if a phrase appears at 16th position, while the whole document contains 700 words, the *phrase_depth* value is 0.97, indicating the first appearance at the beginning of the document.

- ***phrase last occurrence***: we give also importance to phrases that appear at the end of the document, since keyphrases may also appear in the last parts of a document, as in the case of scientific articles (i.e., in the conclusion and discussion parts). The last occurrence value of a phrase is calculated as the number of words preceding the last occurrence of the phrase normalized with the total number of words in the document. The last occurrence value for phrase P in a document D is:

$$last\_occurrence(P, D) = \frac{last\_index(P)}{size(D)},$$

where $last\_index(P)$ is the number of words preceding the phrase's last appearance; $size(D)$ is the total number of words in D.
For instance, if a phrase appears for the last time at 500th position last time in a document that contains 700 words, then the *phrase_last_occurrence* value is 0.71.

- ***phrase lifespan***: the span value of a phrase depends on the portion of the text that is covered by the phrase. The covered portion of the text is the distance between the first occurrence position and last occurrence position of the phrase in the document. The lifespan value is computed by calculating the difference between the *phrase last occurrence* and the *phrase first occurrence*. The lifespan value for phrase P in a document D is:

$$lifespan(P, D) = \frac{[last\_index(P) - first\_index(P)]}{size(D)},$$

where $last\_index(P)$ is the number of words preceding the phrase's last appearance; $first\_index(P)$ is the number of words preceding the phrase's first appearance; $size(D)$ is the total number of words in D.
The result is a number between 0 and 1. Highest values mean that the phrase is introduced at the beginning of the document and carried until the end of the document. Phrases that appear only once through out the document have the lifespan value 0.

As a result of step 2, we get a feature vector for each candidate phrase in the three n-gram lists.

3.3 Step3: Scoring and Ranking

In this step a score is assigned to each candidate phrase which is later exploited for the selection of the most appropriate phrases as representatives of the document. The score of each candidate phrase is calculated as a linear combination of the 5 features. We call the resulting score value *keyphraseness* of the candidate phrase. The keyphraseness of a phrase P with non empty feature set $\{f_1,f_2,...f_5\}$, with non-negative weights $\{w_1,w_2,..w_5\}$ is:

$$keyphraseness(P) = \frac{\sum_{i=1}^{5} w_i f_i}{\sum_{i=1}^{5} w_i}$$

In this initial stage of the research, we assign equal weights to all features, yielding to the computation of the average. Therefore:

$$keyphraseness(P) = \frac{1}{n}\sum_{i=1}^{n} f_i,$$

where:

- n is the total number of features (i.e., *5 in our case*);
- f_1 is the phrase frequency;
- f_2 is the phrase depth;
- f_3 is the phrase pos value;
- f_4 is the phrase last occurrence;
- f_5 is the phrase lifespan.

Producing Final Keyphrases. The scoring process produces three separate lists L_1, L_2, and L_3 containing respectively all the uni-grams, bi-grams and tri-grams with their keyphraseness values. We then select some keyphrases, which are considered to be the most important from each list. In order to produce the *'k'* final keyphrases, we have followed the same strategy that was utilized in [15]. In every list, the candidate phrases are ranked in descending order based on the keyphraseness values. Top 20% (i.e., 20% of *'k'*) keyphrases are selected from "L_3", Top 40% (i.e., 40% of *'k'*) are selected from "L_2", and remaining 40% of rest of *'k'* keyphrases are selected from "L_1'. In this way top k keyphrases for the given document are extracted.

4 Evaluation

The effectiveness and efficiency of our system has been tested on a publicly available keyphrase extraction dataset [19] which contains 215 full length documents from different computer science subjects. Each document in the dataset contains a first set of keyphrases assigned by the paper's authors and a second set of keyphrases assigned by volunteers, familiar with computer science papers. DIKpE is evaluated by computing the number of matches between the

keyphrases attached to the document and the keyphrases extracted automatically. The same partial stemming strategy exploited in candidate phrase selection (see section 3.1) is used also in matching keyphrases. For instance, given the following keyphrase sets S_1 {*component library, facet-based component retrieval, ranking algorithm, component rank, retrieval system*} and S_2 {*component library system, web search engine, component library, component ranks, retrieval systems, software components*} suggested by our system, the number of exact matches is 3: {*component library, component rank, retrieval system*}.

We have carried out two experiments in order to test our system's performance. For the first experiment, we have considered keyphrase extraction works presented by Nguyen&Kan [19] and KEA [24] as baseline systems. From the available 215 documents, Nguyen&Kan has taken 120 documents to compare these with KEA. The maximum number of keyphrases for each document (i.e., *'k'*) is set to ten in Nguyen&Kan. We have taken their results [19] as reference, and in the first experiment we have worked on 120 documents randomly selected from the 215 documents. In both the experiments, we removed the bibliography section from each document in the dataset in order to better utilize the *phrase last occurrence* feature.
Table-1 shows the average number of exact matches of three algorithms when 10 keyphrases are extracted from each document: our system significantly outperforms the other two. For the second experiment, we have extracted keyphrases

Table 1. *Overall Performances*

System	Average # of exact matches
KEA	3.03
Nguyen&Kan	3.25
DIKpE	4.75

for all 215 documents and compared our approach exclusively with the results provided by KEA. We have utilized a total of 70 documents (with keyphrases assigned by authors) extracted from the 215 documents dataset to train the KEA algorithm. For each document, we extracted 7, 15 and 20 top keyphrases using both our approach and KEA.

The results are shown in Table-2: it is clear that even though our system does not undertake any training activity, it greatly outperforms KEA performance.

Table 2. *Performance of DIKpE compared to KEA*

Keyphrases	Average number of exact matches	
Extracted	KEA	DIKpE
7	2.05	3.52
15	2.95	4.93
20	3.08	5.02

Table 3. Top seven keyphrases extracted by DIKpE system to three sample documents

Document	#26. Accelerating 3D Convolution using Graphics Hardware.	#57. Contour-based Partial Object Recognition using Symmetry in Image Databases.	#136. Measuring e-Government Impact: Existing practices and shortcomings.
keyphrases assigned by the document authors	**convolution** hardware acceleration volume visualization	**object** image **contour** recognition **symmetry**	**e-government** law interoperability architectures **measurement** evaluation
keyphrases assigned by volunteers	***3D convolution*** ***filtering*** ***visualization*** ***volume rendering***	*occlusion* ***object recognition*** ***symmetry*** *contour* *estimation*	*benchmark* ***measurement*** ***e-government*** ***public administration*** ***business process***
keyphrases assigned by DIKpE system	high pass filters **volume rendering** filter kernels **3d convolution** **convolution** **visualization** **filtering**	partial object recognition **object recognition** objects in images occlusion of objects **objects** **symmetry** **contours**	measuring e-government impact **business process** e-governmental services **public administration** **e-government** **measurement** business

A sample output of the DIKpE system for three sample documents is shown Table-3. For each document top seven keyphrases extracted by DIKpE are presented: keyphrases that are assigned by the document authors are shown in normal font, Italics indicates keyphrases that are assigned by volunteers, and boldface in DIKpE's row (third row) shows keyphrases that have been automatically extracted and matched with author or volunteer assigned keyphrases. Even if some keyphrases of DIKpE do not match with any of the keyphrases, they are still correctly related to the main theme of the document.

5 Conclusion and Future Work

In this paper, we have presented an innovative and hybrid approach to keyphrase extraction that works on a single document without any previous parameter tuning. Our current work focuses on the integration of DIKpE system with other

tools presented in the PIRATES framework, in order to exploit keyphrase extraction method for the automatic tagging task. Further work will focus on the evaluation procedure. We assumed here that a keyphrase extraction system is optimal, if it provides the same keyphrases that an author defines for his documents. However, in general there may exist many other keyphrases (different than those pre-assigned by authors) that are also appropriate for summarizing a given document. Thus, a further aspect to consider is to take into account the human subjectivity in assigning keyphrases, considering also adaptive personalization techniques for tuning the extraction process to the specific user's interests. In this paper, we evaluated DIKpE performance on scientific publications which are well structured and lengthy in general, in future, we are planning to test the effectiveness of the system on short documents such as news or blog entries. Finally, for the future work, we plan to investigate different ways to compute the coefficients of linear combination of features. We also need to concentrate on a better way to decide the number of keyphrases to be extracted by the system, instead of using a fixed number.

References

1. Barker, K., Cornacchia, N.: Using noun phrase heads to extract document keyphrases. In: Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence. pp. 40–52. Springer-Verlag, London, UK (2000)
2. Baruzzo, A., Dattolo, A., Pudota, N., Tasso, C.: A general framework for personalized text classification and annotation. In: International Workshop on Adaptation and Personalization for Web 2.0 at UMAP 2009. pp. 31–39. Trento, Italy (2009)
3. Baruzzo, A., Dattolo, A., Pudota, N., Tasso, C.: Recommending new tags using domain-ontologies. In: IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology. vol. 3, pp. 409–412. IEEE, Milan, Italy (2009)
4. Berger, A.L., Mittal, V.O.: Ocelot: a system for summarizing web pages. In: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval. pp. 144–151. ACM, New York, NY, USA (2000)
5. Bracewell, D.B., Ren, F., Kuroiwa, S.: Multilingual single document keyword extraction for information retrieval. In: Proceedings of the 2005 IEEE International Conference on Natural Language Processing and Knowledge Engineering. pp. 517–522. Wuhan (2005)
6. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks 30(1-7), 107–117 (1998)
7. Dattolo, A., Ferrara, F., Tasso, C.: Supporting personalized user concept spaces and recommendations for a publication sharing system. In: Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization. pp. 325–330. Springer-Verlag, Berlin, Heidelberg (2009)
8. D'Avanzo, E., Magnini, B., Vallin, A.: Keyphrase extraction for summarization purposes: the lake system at duc2004. In: DUC Workshop, Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting. Boston, USA (2004)

9. Frank, E., Paynter, G.W., Witten, I.H., Gutwin, C., Nevill-Manning, C.G.: Domain-specific keyphrase extraction. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence. pp. 668–673. Morgan Kaufmann Publishers, San Francisco, CA, USA (1999)
10. Hammouda, K.M., Matute, D.N., Kamel, M.S.: Corephrase: Keyphrase extraction for document clustering. In: Perner, P., Imiya, A. (eds.) MLDM. Lecture Notes in Computer Science, vol. 3587, pp. 265–274. Springer, Berlin, Heidelberg (2005)
11. Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. In: Proceedings of the 2003 conference on Empirical methods in natural language processing. pp. 216–223. Association for Computational Linguistics, Morristown, NJ, USA (2003)
12. Justeson, J., Katz, S.: Technical terminology: some linguistic properties and an algorithm for identification in text. Natural Language Engineering 1, 9–27 (1995)
13. Kosovac, B., Vanier, D.J., Froese, T.M.: Use of keyphrase extraction software for creation of an AEC/FM thesaurus. Electronic Journal of Information Technology in Construction 5, 25–36 (2000)
14. Krulwich, B., Burkey, C.: Learning user information interests through the extraction of semantically significant phrases. In: Hearst, M., Hirsh, H. (eds.) AAAI 1996 Spring Symposium on Machine Learning in Information Access. pp. 110–112. AAAI Press, California, USA (1996)
15. Kumar, N., Srinathan, K.: Automatic keyphrase extraction from scientific documents using n-gram filtration technique. In: Proceedings of the Eight ACM symposium on Document engineering. pp. 199–208. ACM, New York, USA (2008)
16. Litvak, M., Last, M.: Graph-based keyword extraction for single-document summarization. In: Proceedings of the Workshop on Multi-source Multilingual Information Extraction and Summarization. pp. 17–24. ACL, Morristown, USA (2008)
17. Liu, Z., Li, P., Zheng, Y., Sun, M.: Clustering to find exemplar terms for keyphrase extraction. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing. pp. 257–266. ACL, Singapore (2009)
18. Medelyan, O., Frank, E., Witten, I.H.: Human-competitive tagging using automatic keyphrase extraction. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing. pp. 1318–1327. ACL, Singapore (2009)
19. Nguyen, T.D., Kan, M.Y.: Keyphrase extraction in scientific publications. In: Goh, D.H.L., Cao, T.H., Sølvberg, I., Rasmussen, E.M. (eds.) ICADL. LNCS, vol. 4822, pp. 317–326. Springer (2007)
20. Porter, M.F.: An algorithm for suffix stripping. Readings in information retrieval pp. 313–316 (1997)
21. Song, M., Song, I.Y., Allen, R.B., Obradovic, Z.: Keyphrase extraction-based query expansion in digital libraries. In: Proceedings of the 6th ACM/IEEE-CS joint Conference on Digital libraries. pp. 202–209. ACM, New York, NY, USA (2006)
22. Turney, P.D.: Learning algorithms for keyphrase extraction. Information Retrieval 2(4), 303–336 (2000)
23. Wan, X., Xiao, J.: Single document keyphrase extraction using neighborhood knowledge. In: Proceedings of the 23rd National Confernce on Artificial Intelligence. pp. 855–860. AAAI Press, Chicago, Illinois (2008)
24. Witten, I.H., Paynter, G.W., Frank, E., Gutwin, C., Nevill-Manning, C.G.: Kea: practical automatic keyphrase extraction. In: Proceedings of the fourth ACM conference on Digital libraries. pp. 254–255. ACM, New York, NY, USA (1999)
25. Wu, Y.F.B., Li, Q.: Document keyphrases as subject metadata: incorporating document key concepts in search results. Information Retrieval 11(3), 229–249 (2008)