Information Systems 34 (2009) 132–144

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

KP-Miner: A keyphrase extraction system for English and Arabic documents

Samhaa R. El-Beltagy [a,*], Ahmed Rafea [b]

[a] Faculty of Computers and Information, Computer Science Department, Cairo University, 5 Dr. Ahmed Zewail Street, 12613 Orman, Giza, Egypt
[b] Computer Science Department, American University in Cairo, 113 Kasr El Aini St., PO Box 2511, Cairo 11511, Egypt

ARTICLE INFO

Article history:
Received 25 February 2008
Accepted 14 May 2008
Recommended by: Dr. I.D. Melamed

Keywords:
Keyphrase extraction
Heuristic rules
Automatic indexing

ABSTRACT

Automatic keyphrase extraction has many important applications including but not limited to summarization, cataloging/indexing, feature extraction for clustering and classification, and data mining. This paper presents the KP-Miner system, and demonstrates through experimentation and comparison with widely used systems that it is effective and efficient in extracting keyphrases from both English and Arabic documents of varied length. Unlike other existing keyphrase extraction systems, the KP-Miner system does not need to be trained on a particular document set in order to achieve its task. It also has the advantage of being configurable as the rules and heuristics adopted by the system are related to the general nature of documents and keyphrases. This implies that the users of this system can use their understanding of the document(s) being input into the system to fine-tune it to their particular needs.

1. Introduction

The task of extracting keyphrases from free text documents is becoming increasingly important as the use for such technology expands. Keyphrase extraction plays a vital role in the task of indexing, summarization, clustering [1], categorization [2] and, more recently, in improving search results and in ontology learning [3]. A number of systems have addressed the task of keyphrase extraction, the most successful of which have employed supervised machine learning techniques. There are many advantages of using machine learning, the most significant of which is the fact that they can adapt to the specific nature of the documents at hand based on a representative training set. However, these techniques do not offer enough insight into the process of keyphrase extraction and factors that affect it. They also allow their users no room for process tuning even though a user/programmer may have an understanding of his/her document set that may aid in the automatic extraction process, or at least allow them to experiment with it. This paper argues that, by gaining an understating of the keyphrase extraction process, a system can be built that extracts keyphrases from documents that do not have training samples at an accuracy comparable (if not superior) to that of supervised machine learning extractors. For English documents, this claim is substantiated through a comparison with Extractor [1] and KEA [4], two widely used keyphrase extraction systems, the result of which shows the presented work to outperform both. The comparison also shows that in terms of running time the presented system is faster. The work further illustrates that such an understanding can aid in the development of keyphrase extractors for languages other than English by applying the developed algorithm on the Arabic language and presenting evaluation results of this application.

The next section provides a brief background on the task of keyphrase extraction as well as highlights related work. Section 3 presents the main ideas behind this work

* Corresponding author. Tel.: +20 2 33030752.
E-mail addresses: samhaa@computer.org (S.R. El-Beltagy), rafea@aucegypt.edu (A. Rafea).

0306-4379/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2008.05.002

and the algorithm built based on those ideas, while Section 4 outlines changes made in order to adapt the algorithm to work with Arabic. Section 5 presents the evaluation experiments and results, and finally Section 6 concludes this paper and presents future research directions.

2. Background and related work

Keyphrases can be defined as a list of terms each of which is made up of one or more words and that describe the document WITH which they are associated. Because of their relatedness, in this report, the terms keyphrases, keywords and key terms are used interchangeably. Traditionally, the task of keyphrase annotation is carried out by human annotators who either freely assign a set of keywords they view as representative of a document to that document, or who select representative terms from a controlled vocabulary. In both cases, keyphrase assignment is left to the discretion of the annotator. However, because the use of a controlled vocabulary confines the annotator to a pre-defined list of terms, keyphrases assigned using this method usually approximate subject areas covered by a document without actually using terms found within the document itself. When using free keyphrase assignment, nothing restricts the annotator to select terms appearing in the document either, but it is often the case that the large majority, if not all, keyphrases selected using this method do appear in the document's text. Manual annotation, using either of these methods or a combination of both, is a very costly and time-consuming process. This has motivated research into means for automatic or semi-automatic annotation. One of the approaches adopted for automatic keyphrase assignment is based on extracting terms that are representative of an input document, from the document itself. This approach is known as keyphrase extraction. So, automatic keyphrase extraction can be defined as the process of extracting keyphrases from a document that a human author is likely to assign to that document. Besides providing a brief summary of document contents, keyphrase extraction has many important applications including but not limited to cataloging, indexing, query refinement, ontology learning, search and feature extraction for clustering and data mining. Indexing is a particularly important application in digital libraries as the majority of currently available publications do not contain author-assigned keyphrases.

Work on automatic keyphrase extraction only dates back to the nineties. First attempts to approach this task were purely based on heuristics [5]. However, keyphrases generated by this approach failed to map well to author-assigned keywords, indicating that the applied heuristics were poor ones [6]. Motivated by the spectrum of potential applications of accurate keyphrase extraction and the failings of the heuristic model, Peter Turney devised a powerful machine-learning-based, keyphrase extraction system called GenEx [1,6]. In building this system, Turney was the first to approach the task of keyphrase extraction as a supervised learning problem. The GenEx system is in fact made up of two separate components: Extractor and Genitor. The Extractor system uses a set of heuristics to identify the phrases that are most likely to map to those of the authors. In that system, candidate keyphrases are phrases of up to three words not beginning or ending with stop words or punctuation marks. The candidates are stemmed using truncation, where the truncation length is a parameter to the developed system. Scoring of candidate phrases is based on their length (determined by the number of their constituent terms) and their position in the text. In this model, longer phrases are multiplied by a constant which represents a boosting factor to adjust bias towards single terms that appear in higher frequency. There is a separate boosting factor for terms that are two words in length and those that are three. The weight of keyphrases appearing early on in a document also gets multiplied by a factor that is greater than one, and the weight of those appearing later in the document gets multiplied by a factor of less than one. Two position thresholds are used to determine which factor to use. In total, an Extractor has twelve different numeric parameters and flags examples of which are the stem truncation length, position thresholds, stem boosting factors and desired number of output keyphrases. After scoring candidate phrases, duplicates are removed and the most frequent full form for each stemmed phrase is selected. The Extractor then presents the top-ranked n keyphrases as output to the user, where n is a user-specified number. In order to optimize the operation of the Extractor algorithm, Genitor is applied to determine the best parameter settings from some training data. After developing and testing, the system is patented and commercialized [7].

Building on the work of Turney is another system called KEA (keyphrase extraction algorithm) [4]. The KEA system also approaches the keyphrase extraction task as a machine learning one as it considers all document phrases as potential keyphrases and the goal is to classify these either as keyphrases or not. KEA utilizes a naïve Bayes learning model. The model is typically trained using a set of documents with known keyphrases, and is then employed to determine which of a previously unseen document's phrases are most likely to be key ones. In KEA, determination of candidate keyphrases is carried out in a similar manner to that of Extractor, but stemming is carried out using the iterated Lovins stemmer [8] rather than by truncation. Also, stopwords are allowed to appear in the middle of a candidate phrase, but not at the beginning or the end. (This is different from Extractor, where stopwords are considered as delimiters.) Two features are employed by the model; these are the TF-IDF weight of the phrases, and their relative position within the document. As in Extractor, given an input document from which keyphrases are to be extracted, the user is presented with the top-ranked n keyphrases, sorted by their probability of being a keyphrase. When applying KEA and Extractor to the same dataset and comparing the resulting keyphrases to author-assigned ones, the performance of KEA was found to be comparable to that of Extractor.

Another keyphrase extraction system is that developed by Hulth [9]. Like GenEx and KEA, this work also utilizes

machine learning techniques, but, unlike either, it places no limit on the length of the extracted keyphrases. Stemming in this system is carried out using the Porter algorithm [10]. Experimentation carried out by Hulth has shown that using a combination of several prediction models applied on noun phrase candidates produced the best results. Specifically, the prediction model uses four features, which are term frequency, inverse document frequency (KEA uses these two as one combined feature: TF -IDF measure), the position of a word's first occurrence and a PoS-tag. Hulth argues that using a PoS tag as a feature significantly improves the results of keyphrase extraction as certain PoS-patterns are more likely to denote keyphrases. Unfortunately however, Hulth has not applied her system on any of the datasets used for evaluating Extractor and KEA; so there is no common basis for comparing the results of the three systems.

Another system that relies on the linguistic features of a document in order to perform keyphrase extraction is the LAKE system, which does so through the use of a naïve Bayes classifier [11]. The system, however, was developed for the specific task of summarization and fails to compare itself with any existing keyphrase extraction systems.

Work presented in [12] proposes a keyphrase extraction algorithm which can be used for both supervised and unsupervised extraction. The algorithm considers each document as a semantic network where the structural dynamics of these networks can be used to identify key nodes, which in turn can be used to extract keyphrases. The presented algorithm is geared more towards digital books, arguing that other existing keyphrase extraction systems are more focused on papers and web pages. Experiments demonstrate that the proposed keyphrase extraction algorithm performs comparatively to both Extractor and KEA even without training. As can be expected, the system was found to perform best on a dataset containing digital books. However, due to lack of phrase redundancy in short documents, the system does not perform well with these types of documents.

Following up on his work on keyphrase extraction, in [13], Turney proposed changes to the KEA system so as to address the limitation of incoherence among extracted keyphrases. The motivation for this work is that keyphrases that are not semantically related to other extracted keyphrases are often outliers and that filtering them out would improve the results of a keyphrase extraction system. In this work, Turney uses the degree of statistical association determined through the use of web mining techniques in order to determine semantic relatedness. The major drawback of this work is that it takes up a lot of time in order to calculate the coherence feature (almost 15 min per document) [13].

In addition, there are a number of other systems that were specifically developed for extracting keyphrases from web documents such as those presented in [14] and [15].

It is important to note that in most currently available keyphrase extractors the correspondence of extracted keyphrases to author-assigned ones is still pretty low. This is not a reflection on the capabilities of these extractors as much as it is an indicator of the difficulty of the task as even human annotators are likely to assign different keyphrases to the same document.

3. Conceptual framework

Devising the algorithm described in this paper was mainly inspired by the nature of documents and keyphrases. Specifically, the work builds on the following observations and hypothesis:

- The number and frequency of compound terms in any given document is usually less than that of single keywords. Effective keyphrase extraction is then dependant on the determination of an appropriate boosting factor for potential keyphrases. In this work, this boosting factor is related to the ratio of single to compound terms in each input document.
- Without the use of linguistic features, the extraction of meaningful keyphrases is dependant on the repetition of these within a document.
- When using TF-IDF for weighing terms, using IDF information for compound phrase weight calculation would bias the extraction towards unseen phrases and low-frequency phrases. This would be unfair when building a general rather than a domain-specific extractor as possible phrase combinations are much larger than can be captured by a limited IDF training corpus.
- The position of the first occurrence of any given phrase is significant in two ways. The first is related to the fact that the more important a term is, the more likely it is to appear 'sooner' in the document than later. The second is based on the hypothesis that after a given threshold is passed in any given document, phrases appearing for the first time are highly unlikely to be keyphrases.

These observations have been translated into a set of heuristics that are implemented by the presented KP-Miner system. Basically, when examining the task of keyphrase extraction, it can be stated that it is highly dependant on three main steps: candidate keyphrase selection, candidate keyphrase weight calculation and final keyphrase refinement. Each of these steps is explained in detail as part of the algorithm presented in the following subsections and so are additionally employed heuristics.

3.1. Step 1: candidate keyphrase selection

Since the developed algorithm has no knowledge of what a phrase is, it has to employ a set of conditions in order to elicit candidate keyphrases. As a phrase will never be separated by punctuation marks within some given text and will rarely have stop words within it, the first condition a sequence of words has to display in order to be considered a candidate keyphrase is that it is not to be separated by punctuation marks or stop words. A total of 187 common stopwords (the, then, in, above, etc.) are used in the candidate keyphrase extraction step. After

applying this first condition on any given document, far too many candidates will be generated, some of which will make no sense to a human reader. To filter out these candidates even more, two further conditions are applied. The first condition states that a phrase has to have appeared at least n times in the document from which keyphrases are to be extracted, in order to be considered a candidate keyphrase. This is called the least allowable seen frequency (lasf) factor. The justification for this is that it is less likely for a phrase appearing with at least this frequency not to be an actual phrase. However, if a document is short, n is decremented. Two different values for the lasf factor have been used in the English and Arabic versions of the system. The reason n is different for the English and Arabic extractors is that in Arabic compound phrases are not as common as they are in English. So in Arabic it is highly likely for phrases appearing two times or more to be actual phrases, while in English they can occur at this frequency and still denote gibberish. This is a heuristic that was confirmed upon experimenting with different lasf values and examining the generated list of candidate phrases in both English and Arabic. So for the English extractor lasf is set to three and for Arabic to two.

The second condition is related to the position where a candidate keyphrase first appears within an input document. Through observation as well as experimentation, it was found that in long documents phrases occurring for the first time after a given threshold are very rarely keyphrases. So a cutoff constant CutOff is defined in terms of a number of words after which if a phrase appears for the first time it is filtered out and ignored. However, it is important to note that using such a parameter in documents that are educational in nature and where new information is introduced till the very end of the document can do more harm than good. Using a cutoff value is only useful in certain types of documents such as academic publications and most web pages. This is why an understanding of the document at hand and its nature is essential. The initial prototype of the presented system set this cutoff value to a constant value of 850 after being fine-tuned using a handful of documents [16]. However, in order to select an optimal cutoff value, an experiment was set up to determine this value based on document length. In this experiment eight different datasets were constructed, the first six of which each had ten documents that fall in the same size range. Because the seventh and the eighth datasets had a wider size range, they contained more documents (the seventh had 12 documents, while the eighth had 20). Table 1 describes these datasets in more details.

In the devised experiment, the keyphrase extractor was run on each of these datasets with a cutoff value, with this value ranging from 100 words to approximately the average size of each dataset. Each time the experiment was run, the cutoff value was incremented by 50 words and the average number of keywords produced by the experiment was logged. These iterations were carried out three times: once to extract seven keywords, another to extract fifteen and finally an iteration to extract twenty keywords. After all runs were executed, the average for each cutoff value obtained from extracting seven, fifteen, and twenty keywords for each dataset at each cutoff value was calculated. Since the whole idea of having a cutoff value is to maximize the number of detected keywords, a score was obtained by normalizing these values by dividing each average value by the maximum obtained average. So the value of one represents the highest obtainable number of keyphrases using the devised algorithm. A graph plot across all datasets is shown in Fig. 1.

What the graph seems to suggest is that, irrespective of the document size, there is huge overlap between the curves around the 500 word cutoff. Specifically, when all dataset averages were additionally averaged in a similar manner as for individual datasets, the best cutoff value seemed to be 400. (This is where the highest average across all datasets is achieved with a value of 0.97.) It also confirms what the authors hypothesized, which is that using all words in the document would worsen the results of the keyphrase extraction algorithm. This is evident by how all graphs initially exhibit a rise in recall when the cutoff value is incremented and then decrease until they level off. Looking at individual figures within each of the datasets also revealed that using a lower cutoff value seems to favor precision when a small number of keys is extracted, while choosing a larger cutoff value favors precision when a larger number of keys is extracted. The experiment, however, did not take into consideration the observation that key terms that appear at the beginning of the document are likely to be more important than those that appear at a later stage. It also introduced a sudden cutoff rather than providing a smooth decay function that slowly reduces the weights of words encountered after a certain point. Another experiment was carried out to determine whether the introduction of a decay function would further improve the results. In this experiment, the

Table 1
Different cutoff tuning datasets and their properties

	Allowable range (# of words)	# of docs	Min size	Max size	Average
Dataset 1	1000–1999	10	1033	1827	1457
Dataset 2	2000–2999	10	2037	2823	2475
Dataset 3	3000–3999	10	3084	3888	3478
Dataset 4	4000–4999	10	4000	4979	4365
Dataset 5	5000–5999	10	5035	5930	5492
Dataset 6	5000–6999	10	6092	6993	6432
Dataset 7	8000–9999	12	7048	9882	8582
Dataset 8	10000–14600	20	10183	14576	12070

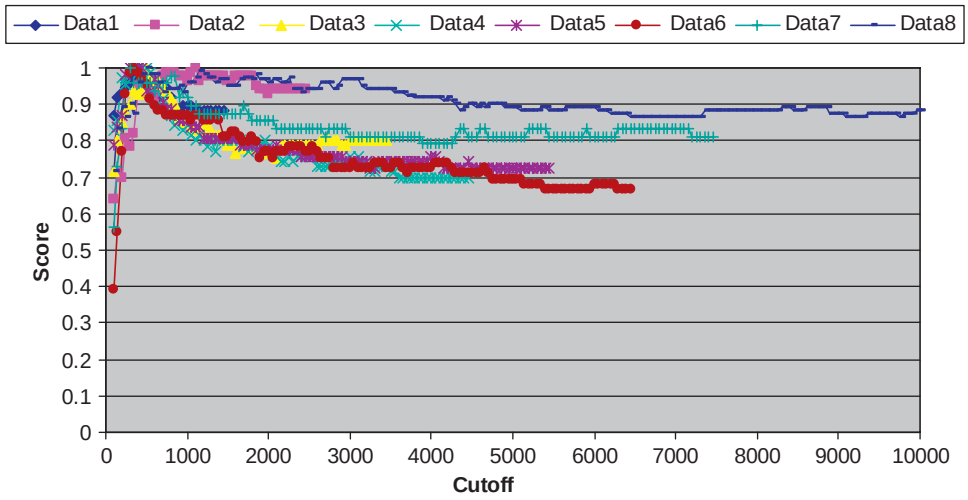

Fig. 1. Cutoff verses score over used datasets.

value of the best cutoff factor (400) was doubled and a high boosting factor for terms that appear at the beginning of the document was introduced. This factor was then slowly decreased to 1 for terms that appear after 30% of cutoff words have been encountered and then slowly decreased again for terms that appear after 50% of the document to a number that is lower than 1. However, applying these position rules to the eight datasets only improved the results by an average of $1.12 \pm 4.3\%$, which is not really significant.

In the implementation of the KP-Miner system, the phrase extraction step described above is carried out in two phases. In the first phase, words are scanned until either a punctuation mark or a stop word is encountered. The scanned sequence of words and all possible *n*-grams within the encountered sequence, where *n* can vary from 1 to sequence length -1, are stemmed and stored in both their original and stemmed forms. If the phrase (in its stemmed or original form) or any of its sub-phrases has been seen before, then the count of the previously seen term is incremented by one, otherwise the previously unseen term is assigned a count of one. Very weak stemming is performed in this step using only the first step of the Porter stemmer [10]. This contrasts with Extractor and KEA, both of which use aggressive stemming. The reason the original form is kept is so that the most frequently occurring original form can be presented to the user in case the phrase is a key one. In the second phase, the document is scanned again for the longest possible sequence that fulfills the conditions mentioned above. This is then considered as a candidate keyphrase. Unlike most of the other keyphrase extraction systems, the devised algorithm places no limit on the length of keyphrases, but it was found that extracted keyphrases rarely exceed three terms.

3.2. Step 2: candidate keyphrases weight calculation

As stated before, the success of a keyphrase extraction program means that the program is capable of extracting terms in the document that best describe this document, i.e. it aims to capture the key features of a document. Single key features obtained from documents by models such as TF-IDF [17] have already been shown to be representative of documents from which they have been extracted, as demonstrated by their wide and successful use in clustering and classification tasks. However, when applied to the task of keyphrase extraction, these same models performed very poorly [6]. To understand why this is so, a closer examination of documents is needed. Looking at almost any document, it can be observed that the occurrence of phrases is much less frequent than the occurrence of single terms within the same document. For example, by examining five of the document datasets used in the evaluation of this work and described later (NASA, Journals, Aliweb pages, FIPS and Journals2), it was found that on average the number of single candidate terms was 6.53 times more than that of compound terms (± 2.89). So it can be stated that one of the reasons that TF-IDF performs poorly on its own when applied to the task of keyphrase extraction is that it does not take this fact into consideration, which results in a bias towards single words as they occur in larger numbers. So, a boosting factor is needed for compound terms in order to balance this bias towards single terms. This observation was also utilized by the Extractor system [7]. In [1] and the initial prototype of this work [16], this value was taken to be a constant. However, since the ratio of single to compound terms varies from document to document, it is more logical that the boosting factor be a function of that ratio. So, in this work, for each input document d from which keyphrases are to be extracted, the boosting factor B_d is calculated as follows:

$$B_d = |N_d|/(|P_d|\alpha)$$

and if $B_d > \sigma$ then $B_d = \sigma$, where $|N_d|$ is the number of all candidate terms in document d, $|P_d|$ is the number of candidate terms whose length exceeds one in document d and α and σ are weight adjustment constants. Without the introduction of the α constant, the boosting factor would

be so large that it would actually result in a bias towards compound terms. This is also why a ceiling for the boosting factor was introduced in the form of the σ constant. To select the values of α and σ, a dataset of 40 documents randomly selected from five of the datasets used for evaluation (Nasa, FIPS, Aliweb, Journals and Journals2) was constructed. The KP-Miner system was then used to extract seven, fifteen and twenty keywords from these documents over a set of iterations and the number of average keywords produced in each iteration was logged. These iterations were carried out using a data range of 1.4–3 for α (with 0.1 increments) and 2.8–4.4 for σ (with 0.2 increments). There was more than one combination of α and σ that resulted in maximum key retrieval across all iterations; these are shown in Table 2. Hypothetically, the selection of any of these combinations would maximize keyphrase extraction effectiveness. However, when selecting a combination for the implemented system, only σ points that have been repeated 3 times were considered (2.8, 3.0, 3.2). Then only α values that fall in between two other combinations that achieve maximum retrieval were considered (these are highlighted in bold). The median was then selected. (This is highlighted in bold and underlined.) So, the combination used by the implemented system is 3 for σ and 2.3 for α.

To calculate the weights of document terms, the TF-IDF model in conjunction with the introduced boosting factor is used. However, another thing to consider when applying TF-IDF for a general application rather than a corpus-specific one is that keyphrase combinations do not occur as frequently within a document set as do single terms. In other words, while it is possible to collect frequency information for use by a general single keyword extractor from a moderately large set of random documents, the same is not true for keyphrase information. There are two possible approaches to address this observation. In the first, a very large corpus of a varied nature can be used to collect keyphrase-related frequency information. In the second, which is adopted in this work, any encountered phrase is considered to have appeared only once in the corpus. This means that, for compound phrases, frequency within a document as well as the boosting factor are really what determine its weight as the idf value for all compound phrases will be a constant c determined by the size of the corpus used to build frequency information for single terms. If the position rules described in the previous section are also employed, then the position factor is also used in the calculation for the terms' weights. In summary, the following equation is used to calculate the weight of candidate keyphrases whether single or compound:

$$w_{ij} = tf^*_{ij} idf^* B^*_i P_f$$

where w_{ij} is the weight of term t_j in Document D_i, tf_{ij} is the frequency of term t_j in document D_i, idf is $\log_2 N/n$, where N is the number of documents in the collection and n is the number of documents where term t_j occurs at least once. If the term is a compound, n is set to 1. B_i is the boosting factor associated with document D_i, P_f is the the term position associated factor. If position rules are not used this is set to 1.

Table 2
Best obtained σ and α combinations

σ	α
2.8	2.2
2.8	**2.3**
2.8	2.4
3.0	2.2
3.0	**2.3**
3.0	2.4
3.2	2.2
3.2	**2.3**
3.2	2.4
3.4	2.2
3.6	2.2
3.8	2.2

3.3. Final candidate phrase list refinement

The KP-Miner system allows the user to specify a number n of keyphrases s/he wants back and uses the sorted list to return the top n keyphrases requested by the user. The default number of n is five. As stated in step one, when generating candidate keyphrases, the longest possible sequence of words that are un-interrupted by possible phrase terminators are sought and stored, and so are sub-phrases contained within that sequence provided that they appear somewhere in the text on their own. For example, if the phrase 'excess body weight' is encountered five times in a document, the phrase itself will be stored along with a count of five. If the sub-phrase 'body weight' is also encountered on its own, then it will also be stored along with the number of times it appeared in the text including the number of times it appeared as part of the phrase 'excess body weight'. This means that an overlap between the count of two or more phrases can exist. Aiming to eliminate this overlap in counting early on can contribute to the dominance of possibly noisy phrases or to overlooking potential keyphrases that are encountered as sub-phrases. However, once the weight calculation step has been performed and a clear picture of which phrases are most likely to be key ones is obtained, this overlap can be addressed through refinement. To refine results in the KP-Miner system, the top n keys are scanned to see if any of them is a sub-phrase of another. If any of them are, then its count is decremented by the frequency of the term of which it is a part. After this step is completed, weights are re-calculated and a final list of phrases sorted by weight is produced. The reason the top n keys rather than all candidates are used in this step is so that lower-weighted keywords do not affect the outcome of the final keyphrase list. It is important to note that the refinement step is an optional one, but experiments have shown that in the English version of the system omitting this step leads to the production of keyphrase lists that match better with author-assigned keywords, while in the Arabic-based one including it leads to better matches. In the author's opinion, employing this step leads to the extraction of higher-quality keyphrases, but this has to be proved through user evaluation of the produced keyphrase.

4. Adapting the system to work with Arabic

Not many modifications were made to the English-based system in order to adapt the system to work with Arabic documents. Changes made include the use of an Arabic stop word list, and an Arabic Stemmer. For the stop word list, a total of 585 words and commonly used verbs were used. Since Arabic is a highly inflected language with a rich morphology, the stemming task for Arabic is more difficult than that for English. Not only can various words take on different suffixes and prefixes according to their context in the text, but infixes are also quite common. Two common approaches used by Arabic stemmers is either light stemming or aggressive stemming. An example of the former is the Light stemmer [18] and the former is the Khoja stemmer [19]. In this work, we have used a custom-built stemmer [20] that is more aggressive than a light stemmer and less aggressive than a heavy stemmer. The used stemmer reduces any word to its singular form by removing prefixes, suffixes and infixes. As stated before, the lsaf factor is set to two when working with Arabic. Also, by experimenting with a small set of documents, it was found that in Arabic using the refinement step yields better results than omitting it.

5. Evaluation

The described system can work on both English and Arabic. For that reason, each version of the system was evaluated separately (the English version and the Arabic one). In the following two subsections, each evaluation is presented independently.

5.1. Evaluating the English version of the system

In order to evaluate how well the developed English version of the KP-Miner system performs, it was tested using seven different English datasets that consist of a total of 502 documents and that already contain author-assigned keyphrases. In this experiment, keyphrases extracted by the presented system were compared to those assigned by the author. The results were then compared to those produced by KEA 3.0 and Extractor. The reason KEA and Extractor were used is because both systems are downloadable and hence can be tested on the same datasets and in the same computing environment as the presented system. In all three systems, a match is considered to have been found if the stem of the extracted keyphrase matches the stem of any of the author-assigned keywords where stemming is carried out via Iterated Lovins stemmer [8]. Four evaluation metrics were used; these are: precision, recall, the average number of keywords extracted correctly per document (Avg. Keys), and the average *F*-measure. In addition, the systems are also compared in terms of efficiency. Since the downloadable version of the Extractor system is a demo one which only allows the extraction of up to seven keyphrases, a comparison between all three systems was only possible when seven keywords are extracted. A comparison between the presented system and KEA 3.0 was also carried out when extracting 15 and 20 keywords. A comparison in terms of the average *F*-measure is only carried out between the presented system and its closest competitor, which is KEA. In the presented experiment, the KP-Miner system was configured to employ cutoff and position rules as described earlier, but refinement rules were not used.

5.1.1. The used datasets

In the evaluation process 7 different document datasets were used. The "NASA" (141 documents), "FIPS" (35 documents), "Journals" (75 documents) and "Aliweb" (90 documents) datasets were kindly provided by Peter Turney and are described in depth in [1]. The "CSTR" dataset consists of 80 short document abstracts which are bundled with the KEA-3.0 system. The "Journal 2" dataset is composed of 60 documents compiled by the authors to initially test the system, and the "Journal 3" dataset is composed of 21 documents collected in order to train KEA-3.0. Both the "Journal 2" and "Journal 3" datasets were collected mainly from the ACM digital library and http://cogprints.org, which is an electronic repository for papers in the areas of Psychology, Neuroscience, Linguistics, Computer Science, Philosophy and Biology. Statistics for the first four datasets are presented in [1]. Statistics for the "CSTR" "Journal 2" and "Journal 3" datasets are presented in Table 3.

5.1.2. Results and discussion

As stated before, evaluation was carried out by calculating how many of the keyphrases generated by the developed system match with author-assigned ones and then comparing these to those extracted by Extractor and KEA. The results are shown in Tables 4–7. Superior scores are highlighted in bold. A sample of the output of all three systems is also shown in Table 8.

By looking at the results, it is obvious that the KP-Miner system outperforms both Extractor and KEA with respect to all datasets. Specifically, when extracting seven keywords, the KP-Miner system returns approximately 22% more correct keyphrases on average than the second nearest competitor (KEA). It also extracts 18% and 15% more keywords than KEA when extracting 15 and 20 keywords

Table 3
Statistics for used datasets

	Num. of docs	Avg. words per document ± standard deviation	Avg. num. of keyphrases ± standard deviation	% of author-assigned keyphrases appearing in full text (%)
CSTR	80	142 ± 67	5.35 ± 1.94	39.5
Journal 2	60	6253 ± 3220	5.43 ± 3.05	94.2
Journal 3	21	6865 ± 3891	5.43 ± 2.32	88.6

Table 4
Precision and recall values from all 3 systems when 7 keys are extracted

	Average precision ± standard deviation			Average recall ± standard deviation		
	KP-Miner	KEA	Extractor	KP-Miner	KEA	Extractor
Journal	**0.223 ± 0.161**	0.192 ± 0.15	0.221 ± 0.256	**0.219 ± 0.152**	0.184 ± 0.15	0.211 ± 0.161
Journal2	**0.286 ± 0.18**	0.231 ± 0.158	0.217 ± 0.242	**0.414 ± 0.264**	0.341 ± 0.245	0.327 ± 0.258
Journal3	**0.279 ± 0.171**	0.258 ± 0.215	0.238 ± 0.314	**0.355 ± 0.165**	0.341 ± 0.309	0.28 ± 0.209
Nasa	**0.173 ± 0.142**	0.15 ± 0.127	0.121 ± 0.186	**0.286 ± 0.248**	0.245 ± 0.217	0.191 ± 0.223
FIPS	**0.278 ± 0.18**	0.245 ± 0.145	0.245 ± 0.251	**0.216 ± 0.13**	0.201 ± 0.123	0.193 ± 0.136
Aliweb	**0.26 ± 0.167**	0.189 ± 0.144	0.21 ± 0.234	**0.337 ± 0.228**	0.24 ± 0.177	0.272 ± 0.192
CSTR	**0.125 ± 0.13**	0.095 ± 0.116	0.114 ± 0.179	**0.148 ± 0.163**	0.121 ± 0.156	0.15 ± 0.168
All	**0.214 ± 0.156**	0.175 ± 0.140	0.176 ± 0.221	**0.277 ± 0.207**	0.228 ± 0.191	0.222 ± 0.197

Table 5
Average keyphrases from all 3 systems when 7 keys are extracted

	Average keys ± standard deviation		
	KP-Miner	KEA	Extractor
Journal	**1.56 ± 1.13**	1.35 ± 1.05	1.55 ± 1.2
Journal2	**2.0 ± 1.26**	1.62 ± 1.11	1.52 ± 1.13
Journal3	**1.95 ± 1.2**	1.81 ± 1.5	1.67 ± 1.49
Nasa	**1.21 ± 0.99**	1.05 ± 0.89	0.84 ± 0.9
FIPS	**1.94 ± 1.26**	1.71 ± 1.02	1.7143 ± 1.15
Aliweb	**1.73 ± 1.09**	1.34 ± 0.96	1.47 ± 1.09
CSTR	**0.813 ± 0.84**	0.66 ± 0.81	0.8 ± 0.86
All	**1.47 ± 1.06**	1.23 ± 0.98	1.23 ± 1.04

respectively. It is worth noting that the developed system also outperforms KEA on the dataset used to train KEA (Journals 3). To evaluate the statistical significance of the results, the unpaired *t*-test was applied on average precision results. The difference between average precision results of the KP-Miner and those of KEA and extractor was found to be statistically significant with 95% confidence or more for its overall average performance when extracting seven keywords. On the level of individual datasets, however, when extracting seven keywords there was only a significant difference between the KP-Miner system and Extractor with respect to the NASA dataset and with the developed system and KEA with respect to Aliweb dataset. When extracting 15 and 20 keywords, the overall average precision of KP-Miner was also found to be significantly different from that of KEA. The difference in the average precision obtained by the implemented system and KEA was also significant with respect to the FIPS, Aliweb, and CSTR datasets.

To compare the efficiency of the three systems, each was used to extract keyphrases from the same dataset, using the same machine and the same running conditions, and the extraction time was logged. The results showed that the developed system completed the keyphrase extraction process in 0.5 the time taken up by KEA and in 0.69 the time required by Extractor, i.e. the developed system is two times faster than KEA and 1.45 times faster than Extractor.

5.2. Evaluating the Arabic version of the system

To evaluate the Arabic version of the system, two experiments were carried out. In the first, the presented system was tested using a corpus containing 100 documents collected from the Arabic Wikipedia [21]. The results were then compared to the output of another Arabic keyphrases extractor. In the second experiment, the KP–Miner system was used to extract keyphrase from a set of Agricultural documents and the resulting keyphrases were compared to concepts in the Agricultural thesaurus AGROVOC [22]. The rationale for the second experiment is that if keyphrases extracted from a set of domain-specific documents match concepts in that domain, then the keyphrase extractor is said to be capable of capturing concepts in the domain, which is a good indicator as to the quality of the generated keys.

5.2.1. The first experiment: comparison with other keyphrase extractors

When comparing the Arabic keyphrase extractor to other extraction systems two experiments were carried out. In the first experiment, seven, fifteen and twenty keywords were extracted by the system from a document set consisting of 100 articles collected from Arabic Wikipedia [21], the details of which are provided in the next subsection. In order to determine whether the obtained results are comparable to those returned by the English-based systems, the average number of keywords in each instance was then compared to highest overall average returned by the three English systems presented in the previous sub-section over the seven used datasets. The results are shown in Table 9. The table also shows the range of averages for each individual used dataset as returned by the three presented systems. Table 10 shows an example of keyphrases extracted by the system from three different documents.

From Table 9, it can be seen that the average number of extracted keyphrases in the Arabic-based system is higher than that of the overall average number of keyphrases extracted from all 7 English datasets and that the returned results are within the same range as those returned by the English systems for individual datasets. This only serves to show that the results of the Arabic-based version of the system are acceptable ones. It does not show, however, how the system fairs in comparison to other Arabic-based keyphrase extractors. To provide a basis for comparison, the authors tried to find other Arabic-based keyphrase extractors, but only found one system, which is the Sakhr Keyword Extractor [23]. Sakhr [24] is a leading company

Table 6
Comparison between developed KP-Miner system and KEA-3.0

	Average precision ± SD		Average recall ± SD		Average keys ± SD	
	KP-Miner	KEA	KP-Miner	KEA	KP-Miner	KEA
(a) When 15 keys are extracted						
Journal	**0.157 ± 0.09**	0.14 ± 0.103	**0.334 ± 0.173**	0.278 ± 0.205	**2.36 ± 1.34**	2.11 ± 1.54
Journal2	**0.186 ± 0.097**	0.157 ± 0.095	**0.562 ± 0.262**	0.474 ± 0.27	**2.78 ± 1.45**	2.48 ± 1.57
Journal3	**0.175 ± 0.111**	0.165 ± 0.105	**0.467 ± 0.222**	0.45 ± 0.283	**2.62 ± 1.66**	2.48 ± 1.57
Nasa	**0.1 ± 0.09**	0.097 ± 0.075	**0.337 ± 0.271**	0.338 ± 0.261	**1.47 ± 1.2**	1.46 ± 1.12
FIPS	**0.206 ± 0.109**	0.158 ± 0.093	**0.349 ± 0.154**	0.269 ± 0.155	**3.09 ± 1.63**	2.37 ± 1.4
Aliweb	**0.19 ± 0.136**	0.104 ± 0.079	**0.425 ± 0.251**	0.279 ± 0.199	**2.24 ± 1.43**	1.58 ± 1.18
CSTR	**0.087 ± 0.095**	0.052 ± 0.056	**0.161 ± 0.165**	0.138 ± 0.159	**0.9 ± 0.89**	0.78 ± 0.84
All	**0.143 ± 0.102**	0.112 ± 0.082	**0.358 ± 0.225**	0.303 ± 0.22	**1.97 ± 1.29**	1.7 ± 1.24
(b) When 20 keys are extracted						
Journal	**0.13 ± 0.074**	0.119 ± 0.078	**0.363 ± 0.18**	0.312 ± 0.205	**2.6 ± 1.49**	2.37 ± 1.56
Journal2	**0.151 ± 0.079**	0.129 ± 0.079	**0.6 ± 0.26**	0.515 ± 0.28	**3.02 ± 1.59**	2.58 ± 1.59
Journal3	**0.138 ± 0.084**	0.136 ± 0.078	**0.496 ± 0.227**	0.498 ± 0.287	**2.76 ± 1.67**	2.714 ± 1.55
Nasa	**0.083 ± 0.069**	0.079 ± 0.058	**0.344 ± 0.271**	0.361 ± 0.268	**1.53 ± 1.3**	1.58 ± 1.17
FIPS	**0.173 ± 0.08**	0.134 ± 0.086	**0.391 ± 0.137**	0.3 ± 0.167	**3.46 ± 1.61**	2.69 ± 1.71
Aliweb	**0.178 ± 0.131**	0.083 ± 0.066	**0.44 ± 0.146**	0.291 ± 0.208	**2.37 ± 1.43**	1.67 ± 1.32
CSTR	**0.083 ± 0.094**	0.04 ± 0.042	**0.164 ± 0.165**	0.143 ± 0.164	**0.913 ± 0.89**	0.8 ± 0.88
All	**0.124 ± 0.087**	0.092 ± 0.065	**0.376 ± 0.206**	0.326 ± 0.226	**2.11 ± 1.4**	1.83 ± 1.25

Table 7
Comparison between the average *F*-measure of the developed KP-Miner system and KEA-3.0 when 7, 15, and 20 keyphrases are extracted

	7 keyphrases			15 keyphrases		20 keyphrases	
	KP-Miner	KEA	Extractor	KP-Miner	KEA	KP-Miner	KEA
Journal	**0.221**	0.188	0.216	**0.214**	0.197	**0.191**	0.172
Journal2	**0.338**	0.275	0.261	**0.279**	0.245	**0.241**	0.206
Journal3	**0.312**	0.294	0.257	**0.255**	0.244	**0.216**	0.214
Nasa	**0.216**	0.186	0.148	**0.154**	0.151	**0.134**	0.13
FIPS	**0.243**	0.221	0.216	**0.259**	0.218	**0.24**	0.185
Aliweb	**0.294**	0.211	0.237	**0.263**	0.167	**0.253**	0.129
CSTR	**0.136**	0.106	0.13	**0.113**	0.079	**0.11**	0.063
All	**0.241**	0.198	0.196	**0.205**	0.17	**0.186**	0.143

in the field of Arabic Text processing, with major products including online dictionaries, automatic machine translators, Arabic OCR and speech recognition software, among others. Clients of the company include the US Defense Intelligence Agency, Sony, Xerox, as well as many international and national organizations. Due to the commercial nature of the company, no details about their keyphrase extraction program have been published, but the tool that is available on-line as part of a suite of text mining products seems to be based on a set of dictionaries of places, people and expressions. When using the product, it highlights keywords in the entered document that it recognizes using either of three criteria: celebrity, location, or expression. The assumption that this is done via a set of dictionaries is derived from an experiment in which terms recognized by the Sakhr extractor as keyphrases were simply entered to the system independent of any context. In this experiment, the Sakhr keyword extractor was still able to recognize these as keyphrases. Similarly, entering the name of an unknown person or a previously unrecognized expression frequently in any input document did not lead to the recognition of that name or expression, which indicates that context and frequency do not influence the operation of this particular keyphrase extractor. Because of the nature of the dataset obtained from Wikipedia, in which many entries are either about places or about celebrities, when applying the Sakhr keyphrase extractor to this dataset, it performed reasonably well.

Applying Sakhr to the selected dataset was not a straightforward task, however, as the Sakhr system is only freely available via a web-based interface as shown in Fig. 1. So articles were entered to this system manually one by one, which is the main reason a larger dataset was not used. When summarizing the results, Sakhr informs the user of the number of keywords it extracts from the input document. Again, unlike the presented system, this is not a user-specified number, but is simply the number of terms that the Sakhr system is able to recognize as keyterms within the input document. This number also does not represent the unique number of identified keywords in the document; instead it represents the

Table 8
Top 7 extracted keywords by all 3 systems from 4 different documents

(1) Document's title: map-based horizontal navigation in educational hypertext		
Author-assigned keyphrases: horizontal navigation, electronic textbook, similarity navigation, concept-based navigation, map-based navigation		
KP-Miner (3 matches)	Extractor (1 match)	KEA 3.0 (1 match)
Horizontal navigation (matched)	Navigation	Navigation
Horizontal links	Map	Horizontal
Map-based navigation (matched)	Education	Horizontal navigation (matched)
Concept-based navigation (matched)	Horizontal navigation (matched)	Map based
Navigation	Hypertext	Hypertext
Horizontal	Educational materials	Horizontal links
Educational	Educational courseware	Courseware
(2) Document's title: functional and symbolic congruity approaches to consumer satisfaction/dissatisfaction in tourism		
Author-assigned keyphrases: consumer satisfaction/dissatisfaction, evaluative congruity, functional congruity, symbolic congruity, tourist		
KP-Miner (2 matches)	Extractor (1 match)	KEA 3.0 (2 matches)
Evaluative congruity (matched)	Congruity	Tourist (matched)
Tourist satisfaction	Satisfaction	Congruity
Consumer satisfaction	Destination	Consumer
Tourist (matched)	Consumer	Satisfaction/dissatisfaction
Destination image	CS/D	Destination
Dissatisfaction	Evaluative congruity (matched)	Evaluative congruity (matched)
Destination	Consumer behavior	His/her
(3) Document's title: American Association of Law Libraries		
Author-assigned keyphrases: information, policy, letters, statements, testimonies, GRC, government, relations, committee, AALL		
KP-Miner (3 matches)	Extractor (3 matches)	KEA 3.0 (2 matches)
AALL (m)	AALL (matched)	Government
Washington affairs	Government	Government relations
Representatives	AALL resolutions	AALL (m)
AALL resolutions	Policy (matched)	Washington affairs
Government relations	Formal expressions	Representatives
Resolutions	Letters (matched)	AALL resolutions
Policy (matched)	Law libraries	Libraries
Letters (matched)		Letters (m)
(4) Document's title: the base rate fallacy myth		
Author-assigned keyphrases: base rate fallacy, Bayes' theorem, decision making, ecological validity, ethics Fallacy, judgment, probability		
KP-Miner (4 matches)	Extractor (4 matches)	KEA 3.0 (2 matches)
Base rate	Base rates	Base rates
Base rate fallacy (m)	Psychology	Base rate fallacy (m)
Judgment (m)	Probability (m)	Probabilistic
Decision making (m)	Judgments (m)	Judgments (m)
Decision makers	Base rate fallacy (m)	MYTH
Real world	Decision making (m)	Probabilistic judgment
Bayes' theorem (m)	Individuating information	Bayesian

overall instances of keyphrases found in the text. So, in general the number of extracted keyphrases is proportional to the length of the document.

Because of the difficulty of manually counting the unique number of extracted keywords, it was not possible to calculate the precision and hence the *F*-measure for the Sakhr system results. Consequently, comparison between the Sakhr system against the presented KP-Miner system was also not possible for these metrics. Instead the recall and the average number of keyphrases recognized over the input dataset were used as the basis for comparison. Another metric that was also used in the comparison between the two systems is the number of times the title of an article was recognized as the top keyphrase. Because the Sakhr system does not rank generated keyphrases, the number associated with this metric with respect to the Sakhr system is the number of times an article's title was recognized as a keyphrase. When comparing the two systems, the number of keywords extracted by the KP-Miner system was set to 20. The results of this comparison are presented in Table 11.

As can be seen by the results, the KP-Miner performs slightly better than the Sakhr system in terms of both recall and number of keyphrases extracted. Also, in the developed system, the number of times an article title was recognized as the highest ranking keyphrase is significantly higher than the number of times the Sakhr system recognized the title as a keyphrase. When analyzing the results, it was found that 42.5% of all keyphrases identified by the Sakhr system were either the name of a location or a celebrity. The authors also noticed that the KP-Miner system performs significantly better than the Sakhr keyword extractor when dealing with scientific entries, while Sakhr performs better with documents in which

Table 9
Comparison between the Arabic keyphrase extractor results and results obtained by English keyphrase extractors

	Avg. # of keywords extracted by the Arabic system	Highest avg. from English systems over 7 datasets	English system range of avgs (min–max)
7 keyphrases	1.7±0.81	1.47±1.06	0.8–1.95
15 keyphrases	2.23±1.05	1.97±1.29	0.78–3.09
20 keyphrases	2.49±1.21	2.11±1.4	0.8–3.46

Table 10
Top 7 extracted keyphrases from 3 different documents

1) Document's Title: هرم أكبر
Assigned keyphrases:
هرم أكبر, أبولو رودس, أحمد محمد عوف, أهرامات الجيزة, أون, الأرض, الهرم, تمثال زوس, حدائق بابل المعلقة, حورس, خوفو
Extracted Keys (3 matches)
الهرم الأكبر(m) , **الهرم (m)**, الملك خوفو ,**خوفو (m)** ,هرم خوفو, حجرة ,مصر القديم
2) Document's Title: إبراهيم ناصف الوردانى
Assigned keyphrases:
إبراهيم ناصف الوردانى, إنجلترا, الخديوي عباس حلمي, بطرس غالي
Extracted Keys (2 matches)
اغتيال بطرس غالي, الوردانى, الحزب الوطني, المصري, الحركة الوطنية , **بطرس غالي (m)**, **إبراهيم ناصف الوردانى(m)**
3) Document's Title: سانتا كلوز
Assigned keyphrases:
سانتا كلوز, أيل, السيدة كلوز, القطب الشمالي, الولايات المتحدة الأمريكية, ثلج, عيد الميلاد, لحية, لغة إنجليزية, ميرا
Extracted Keys (2 matches)
سانتا كلوز (m) , القديس نيكولاس, **عيد الميلاد (m)** , بابا نويل, الهدايا, كلوز, Christmas

Table 11
Result of comparing the developed system with the Sakhr keyword extractor

	KP-Miner	Sakhr keyword extractor
Avg. precision±SD	0.132±0.062	Not applicable
Avg. recall±SD	0.383±0.248	0.323±0.279
Avg. keys±SD	2.49±1.21	2.4±2.07
Avg. *F*-measure	0.196	Not applicable
# of times title was recognized as a top keyphrase	61%	36%
# of times title appeared as one of the top 3 keyphrases	78%	Not applicable

names of locations, celebrities, or well-known expressions such as names of a chemical compound appear and are keyphrases, especially if these are encountered only once.

Efficiency in terms of the time required by the Sakhr system could not be measured because the Sakhr system is not downloadable. However, because of the outlined hypothesis that the Sakhr system uses dictionaries, it is very likely that the system consumes a lot of computing resources and that it is much less efficient than the outlined system. An integration of techniques used by both would probably yield a much more powerful extractor than either.

5.2.1.1. The used dataset. Unlike in English, where there are many resources annotated with keywords, Arabic keyword annotated documents are almost non-existent. This is in fact one of the motivations for this work as there are no sufficient training documents to allow for the adoption of a supervised machine learning approach. Only after completing the implementation of the presented system were the authors able to find such documents in the form of Wikipedia [21] annotated articles. Wikipedia currently contains more than 43,000 Arabic articles. The used dataset consists of 100 randomly selected articles from this pool of documents. In the selection process, very short articles and articles having a date as their main title were ignored. Keywords for each article were obtained from the keyword meta-tag associated with each article, but numeric entries (mostly denoting year numbers) were ignored and so were Wikipedia-related tags (such as article seed for example).[1] The average number of words per document in this dataset is 804±934 and the average number of keyphrases is 8.1±3.2. The percentage of

[1] Dataset is available for experimentation upon request.

author-assigned keyphrases actually appearing within the body of associated articles in this dataset is 81.8%.

5.3. Experiment 2: extracting keyphrases from a domain-specific corpus

In the second experiment designed to test the developed Arabic Keyphrase extraction system, 18 agricultural extension documents issued by the Egyptian Ministry of Agriculture to educate farmers as to the different aspects of farming different crops were concatenated into one big document consisting of 53,696 words and entered to the KP-Miner system. The aim of the experiment was to see whether the extracted keyphrase will map to concepts found in an agricultural ontology. A large number of keyphrases mapping to agricultural concepts indicate that the keyphrase extraction program is actually capable of capturing terms used in the domain from which the documents are derived, i.e. that the system is capable of capturing high-quality phrases. Because the input document consisted of a number of documents, applying the cutoff rule would have implied that keyphrases appearing in documents other than the first one would be ignored. Applying position rules also would have caused keyphrases appearing in the first document to carry more weight. So, for this experiment to work, cutoff and position rules were turned off. This is an example of an instance where understanding the nature of the input document can allow the user to tune the system for best results. The document was then entered to the KP-Miner system and the top 50 keyphrases were extracted. These are shown in Table 12. Extracted keywords were then compared to entries in the AGROVOC agricultural thesaurus [22]. A match was considered to have been found if the extracted keyphrase or its stem matched an entry or a stem of an entry in the thesaurus. In Table 12, keyphrases that match with AGROVOC entries are highlighted in bold. Of the 50 extracted keyterms, 33 matched AGROVOC entries, i.e. 66% of extracted keyphrases were concepts in AGROVOC. This shows that the developed keyphrase extractor is capable of capturing important concepts from a set of documents representative of a specific domain.

6. Conclusion and future work

This paper presents the KP-Miner system, which is currently capable of extracting keyphrases from both English and Arabic documents and which can be easily adapted to

Table 12
Top 50 keys extracted from 18 agricultural documents

1. زراعة	**11. الأوراق**	**21. الإصابة**	42. صدأ	**41. لون**
2. نبات	**12. تربة**	**22. ماء**	**44. سنبلة**	**43. قصب**
3. محصول	**13. سماد**	23. خط	**46. ساق**	**45. الحقل**
4. ري	**14. البذور**	**24. الحبوب**	**48. حصاد**	47. حالة
5. صنف	15. مقاومة	**25. تسميد**	**50. كتان**	49. طريقة
6. حشيشة	**16. الجذور**	26. زراعية	**52. فطر**	51. الشتلات
7. فدان	17. سم	27. بنجر	54. مكافحة	**53. الإنبات**
8. مرض	**18. سكر**	**28. قمح**	**56. بذر**	55. جيزة
9. تقاوي	19. أزوت	29. مبيد	58. رية	57. ميعاد
10. ارض	**20. رش**	**30. الآفات**	**60. حشرة**	**59. الذرة**

Fig. 2. A snapshot of Sakhr's keyword extractor web based interface.

work with any other language. Unlike other keyphrase extraction systems, the presented system has the advantage of being configurable as the rules and heuristics adopted by the system are related to the general nature of documents and keyphrase and can be easily explained to non-specialists so that they can tune the system to their specific documents (if needed). Developers of systems that use keyphrase extraction can also easily change the introduced constants to fit their requirements. Experiments carried out in this paper show that the presented system is efficient as well as effective for the task of automatic keyphrase extraction from both English and Arabic documents (Fig. 2).

Acknowledgment

The authors would like to thank Peter Turney for kindly sharing his datasets.

References

[1] P.D. Turney, Learning algorithms for keyphrase extraction, Information Retrieval 2 (3) (2000) 303–336.
[2] A. Hulth, B. Megyesi, A study on automatically extracted keywords in text categorization, in: 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (CoLing/ACL 2006). Sydney, 2006, pp. 124–154.
[3] K. Englmeier, F. Murtaghí, J. Mothe, Domain Ontology: Automatically Extracting and Structuring Community Language from Texts, IADIS Applied Computing, Spain, Espagne, 2007.
[4] I.H. Witten, et al., KEA: Practical automatic keyphrase extraction, in: The Fourth ACM Conference on Digital Libraries, 1999.
[5] B. Krulwich, C. Burkey, Learning user information interests through the extraction of semantically significant phrases, in: AAAI Spring Symposium on Machine Learning in Information Access, Stanford, CA, 1996.
[6] P.D. Turney, Learning to extract keyphrases from text, Technical Report ERB-1057, National Research Council, Institute for Information Technology, 1999.
[7] P.D. Turney, Extractor, 2005. 〈http://www.extractor.com〉.
[8] J.B. Lovins, Development of a stemming algorithm, Mech. Translat. Comput. Linguistics 11 (1968) 22–31.
[9] A. Hulth, Improved automatic keyword extraction given more linguistic knowledge, in: Conference on Empirical Methods in Natural Language Processing (EMNLP 2003), 2003.
[10] M. Porter, An algorithm for suffix stripping, Program 14 (3) (1980) 130–137.
[11] E.D. Avanzo, B. Magnini, A. Vallin, Keyphrase extraction for summarization purposes: the LAKE System at DUC2004, in: The Document Understanding Workshop, HLT/NAACL Annual Meeting, Boston, USA, 2006.
[12] C. Huang, et al., Keyphrase extraction using semantic networks structure analysis, in: The Sixth International Conference on Data Mining (ICDM '06), Hong Kong, 2006.
[13] P.D. Turney, Coherent Keyphrase Extraction via Web Mining, IJCAI, Mexico, 2003.
[14] M. Chen, et al., A practical system of keyphrase extraction for web pages, in: CIKM'05. Bremen, Germany, 2005.
[15] D. Kelleher, S. Luz, Automatic hypertext keyphrase detection, IJCAI'05. Edinburgh, UK, 2005.
[16] S.R. El-Beltagy, K.P. Miner, A simple system for effective keyphrase extraction, in: The Third IEEE International Conference on Innovations in Information Technology (IIT '06), Dubai, UAE, 2006.
[17] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Info. Process. Manage. 24 (5) (1988) 513–523.
[18] L.S. Larkey, L. Ballesteros, M.E. Connell, Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis, in: SIGIR'02, Tampere, Finland, 2002.
[19] S. Khoja, R. Garside, Stemming Arabic text, Technical Report, Computing Department, Lancaster University, Lancaster, 1999. 〈http://www.comp.lancs.ac.uk/computing/users/khoja/stemmer.ps〉.
[20] S.R. El-Beltagy, A framework for the rapid development of dictionary based domain specific Arabic stemmers, Technical Report (TR/COE_WM/12/12/2007) Center of Excellence for Data Mining and Computer Modeling, 2007.
[21] Wikipedia, Wikipedia, the free encyclopedia. 〈http://ar.wikipedia.org/wiki/Main_Page〉, 2007.
[22] AGROVOC, Multilingual Agricultural Thesaurus, Food and Agricultural Organization of the United Nations, 2007 〈http://www.fao.org/agrovoc/〉.
[23] Sakhr, The Sakhr Keyword Extractor, 2007. 〈http://www.sakhr.com/Technology/Keyword/Default.aspx?sec=Technology&item=Keyword〉.
[24] Sakhr, 2007. 〈http://www.sakhr.com/〉.