

DIKEA: Domain-Independent Keyphrase Extraction Algorithm

David X. Wang, Xiaoying Gao, and Peter Andreae

School of Engineering and Computer Science,
Victoria University of Wellington, New Zealand
{david.wang,xiaoying.gao,peter.andreae}@ecs.vuw.ac.nz

Abstract. This paper introduces a new domain-independent keyphrase extraction system (DIKEA). Keyphrase extraction is a challenging problem that automatically extracts or assigns keyphrases to documents and it can benefit many research areas such as information retrieval, particularly indexing, clustering, and summarization. A landmark research KEA (Keyphrase Extraction Algorithm) formulated the problem as a supervised machine learning problem and successfully applied a Naïve Bayes model to it, which showed great promise but the performance is not satisfactory. Its state-of-the-art extension KEA++ has a significantly improved performance but relies on a domain specific vocabulary which is often not available or not complete. This paper introduces a novel domain-independent approach and has three main contributions: utilising the largest online knowledge source—Wikipedia—for keyphrase candidate selection; presenting new features for keyphrase evaluation, including a Wikipedia-based feature—link probability; and evaluating a number of different learning algorithms, including multi-layer perceptrons, for keyphrase selection. Experiments show that our system clearly outperforms KEA and closely matches the performance of KEA++, without requiring any domain-specific knowledge such as KEA++’s vocabulary list.

1 Introduction

Keyphrases provide insight into the contents of a document. Similar in purpose to an abstract, they introduce the topic of the document but more concisely, and can be easily indexed. Such keyphrases are highly beneficial for large document collections such as the ACM Digital Library. They are also particularly useful for document retrieval systems, where they can be used as the basis for documents indexing; collection browsing, by creating a hierarchical structure of keyphrases; and document clustering, by representing documents as a weighted vector of keyphrases.

Usually, keyphrases are assigned manually by either the authors themselves or by professional indexers. The problem is that producing high-quality keyphrases manually is expensive and time-consuming as it requires individuals with knowledge of a document’s subject matter. As a result, the great majority of

documents such as web pages come without keyphrases, putting automatic keyphrase extraction techniques in great demand.

Automatic keyphrase extraction is a very challenging task. In fact, research has found that the agreement between even human labelling is low [1]. Many research fields are related to keyphrase extraction; for example, information retrieval systems typically weight all terms using TF.IDF, many feature selection methods such as Information Gain and Chi-square can be applied to select key terms for document representation, and keyphrase extraction may be considered as a kind of information extraction. However, keyphrase extraction as a separate field is not very well studied, probably due to the lack of testing data. In 1999, a landmark research KEA (Keyphrase extraction algorithm) [1] formulated keyphrase extraction as a supervised machine learning algorithm and successfully applied Naïve Bayes learning to this problem and provided the benchmark testing data. There has been much research to further improve KEA [2,3,4] and its state-of-the-art extension KEA++ [5,6] significantly improve its performance. However, KEA++ relies on a domain specific vocabulary which is supposed to include all the descriptive phrases for a particular domain. In reality, such a vocabulary is often either unavailable or incomplete, and in those cases where a vocabulary is available, it may not be updated with time.

Our research aims to introduce a new domain-independent approach to the problem that does not require a domain specific vocabulary. We have chosen to use Wikipedia as an external, domain-independent resource because it is currently the largest, reliable and keep growing source available online. Our main goal is to develop an efficient and effective system to exploit the knowledge in Wikipedia for domain-independent keyphrase extraction. Previously KEA and KEA++ used features such as Tf.Idf and phrase length, and applied a Naïve Bayes learning model. We investigate new features especially Wikipedia-based features, and apply a number of learning algorithms including linear regression and neural networks.

The rest of this paper is organised as follows. Section 2 outlines related work and details two existing systems KEA and KEA++. Section 3 provides an overview of our domain independent approach, while Section 4 provides details regarding the new features introduced and other learning algorithms applied. Methods for empirical evaluation and results are presented and analysed in Section 5. Finally, Section 6 concludes and discusses future work.

2 Related work

Wikipedia as one of the largest online information resource has been found helpful for query expansion[7], semantic relatedness analysis [8], document clustering [9], and information retrieval [10] and many other areas. Our research is motivated by Huang et al. [9] which introduces a Wikipedia-based concept representation for documents and shows it improves document clustering performance. It matches terms in a document with Wikipedia anchor texts to identify the concepts and then represents each document as a weighted vector of concepts. Our system is similar in that we use Wikipedia anchor texts to select

the candidate keyphrases from a document. Giving the big size and great coverage of the vocabulary of Wikipedia anchor texts and its expanding nature, our Wikipedia-based approach is domain independent and it works uniformly for different domains.

There have been a number different approaches to automatic keyphrase extraction [11]. [12] is an early machine learning approach for keyword extraction. Liu et al. [13] uses clustering to find exemplar terms and it shows clustering is helpful to identify good phrases that is the centre of a group of terms. [14] is another statistical based approach that can extract keywords from each single document without a corpus, and its performance is comparable to TF.IDF. Our approach is a supervised machine learning approach and the most closely related systems are KEA [1] and KEA++ [5].

KEA (Keyphrase Extraction Algorithm) [1] formulates keyphrase extraction as a supervised machine learning problem, and it does not require any prior knowledge of the document's domain. KEA++ [5] is the current state-of-the-art version of KEA, but it does so at a cost to domain-independence.

The rest of this section will detail KEA and KEA++. The following three subsections will briefly cover the three key steps of both KEA and KEA++ and identify rooms for further improvements. The comparison results between our system and the two systems are given in section 5.

2.1 Candidate Identification

The Candidate Identification step extracts candidate keyphrases from the document being summarised. The original KEA algorithm simply extracts all possible unigrams, bigrams and trigrams within phrase boundaries that occur at least twice in the document. Phrase boundaries are defined to be brackets, punctuation marks or numbers. After this, candidates which start or end with a stopword are removed and those that only contain a single proper noun are also removed. Finally, candidates are converted to lower case and stemmed using a Lovins Stemmer [15].

KEA++ improves on this by comparing candidate phrases with a controlled vocabulary in the domain of the document which is known to be descriptive.

KEA's solution to this step is very simple, but will produce a huge number of candidate keyphrases (usually many times the amount of words there are in the document) and assumes that a descriptive keyphrase appears in the document as a consecutive sequence of words. KEA++ does much better, significantly reducing the number of extracted candidates. However, the controlled vocabulary used has to be tailored to the document's domain, which means the overall solution becomes domain-dependent.

2.2 Feature Calculation

Once candidate keyphrases have been selected, a series of features will be calculated for each candidate. KEA used two such features as follows.

TF.IDF – Term frequency multiplied by inverse document frequency. This measure gives an indication as to how often a term/phrase appears in the current document as well as how unique the term is among all documents.

First Occurrence – This measures when a candidate first appears in the current document. It is calculated as a ratio of how many words precede the candidate’s first occurrence vs. how many words are in the entire document.

KEA++ introduces two additional features as follows.

Candidate Length – The length of a candidate in terms of number of words.

Node Degree – Represents the number of thesaurus links that connect a candidate to other candidates. If a document is on a particular topic, then it will likely cover most of the thesaurus terms from that topic. Therefore, candidates with higher node degree may be more significant.

There are several issues with the above features. The TF.IDF value is calculated based on documents in the training corpus, which may not work well if the corpus is not representative of the domain. Furthermore, TF.IDF is a product of two separate features – term frequency (TF) and inverse document frequency (IDF). We would like to explore whether more interesting results might be obtained by separating these features. Finally, the Node Degree feature uses the domain-specific vocabulary, again making the overall solution domain-dependent.

2.3 Training Model

The final step of KEA involves feeding all of this information (the features) into a Naïve Bayes model to learn a classifier. The trained model outputs the *goodness* of a particular candidate keyphrase from a previously unseen document. Candidates are then ranked by this value and a certain number of candidates are chosen from the top of the list to be the final extracted keyphrases.

Naïve Bayes assumes that each feature is independent from the rest, which we don’t believe is true. We would like to investigate other learning algorithms, especially the algorithms that can directly use continuous data.

3 Domain Independent Approach

KEA++ outperforms KEA by over a factor of two in terms of both precision and recall [5]. However, it does so by sacrificing the domain-independent nature of KEA. Our aim is to create a new system which can match KEA++’s performance, while avoiding the need for anything domain-specific. To do so, we need to identify the key differences between KEA and KEA++.

One such key difference is the use of a controlled vocabulary to narrow down the number of candidates extracted in the initial step of the algorithm. Using the FAO dataset mentioned in Section 5.1, KEA++ extracts five times less candidates than KEA on average (2491 vs. 463). It does this with a minimal loss in **candidate recall**, which is the maximum recall obtained if all candidates

are classified as keyphrases, in other words, the recall before machine learning is applied. This is achieved by only extracting candidates that are matched to a controlled vocabulary of approximately 27,000 terms which are known to be good descriptors of agricultural documents. Classifiers trained using this condensed set of candidates perform significantly better, because of the reduction in misleading non-sense phrases which are an artefact of extracting overlapping n-grams. Precision is also increased as a direct result of only having to filter a much smaller subset of candidates.

However, the KEA++ system will breakdown if it is supplied with a controlled vocabulary which does not represent a particular document's domain, or is only partially representative. This could happen if a document covers a topic that is broader than the vocabulary's domain, or there is simply no suitable vocabulary in existence. In the first scenario, candidate recall will be significantly lowered as a result of potentially good keyphrases being filtered out by the non-representative vocabulary. For the second scenario, the system will just revert back to KEA.

What we require is a vocabulary of phrases that do not conform to any particular domain, but rather represent descriptors of all topics known to humanity. DIKEA utilises Wikipedia to obtain this data, more specifically, it matches candidates to a vocabulary of Wikipedia anchor texts (with millions of terms). Anchor texts are internal hypertexts that link to Wikipedia articles, which is what makes them great descriptors for documents. Wikipedia Miner[16,17] is used to speed up this process, such that it barely slow down the system at all. While a broad vocabulary does not reduce the number of candidates as aggressively as one that is more specific, it still prunes out most non-sense phrases. Unlike KEA++, DIKEA treats documents from different domains in a uniform way. Furthermore, as new anchor texts are introduced to Wikipedia, our system will be able to update itself alongside the natural progression of human knowledge.

Another key difference between KEA++ and KEA is the addition of two candidate features – candidate length and node degree. The extra values gained from these features improve the trained classifier by 4 to 5 percent[5]. Node degree, while useful, relies on domain-specific data. Therefore, DIKEA will use all other candidate features from KEA++ except node degree, as well as a couple of new ones.

Standard deviation will be one of the new features. It measures the spread of a candidate phrase within the document. The motivation behind this is that a phrase which appears evenly throughout a document may hold a different significance (in terms of being a keyphrase) to a phrase which only appears in a single paragraph or section. Link probability [18] is the other new feature, which we use to measures the uniqueness of a phrase – in other words, how well a phrase can distinguish different documents. Link probability [18] is defined as the probability of a phrase being used as a link in all Wikipedia documents with the phrase. Equation (1) shows how link probability is calculated for a given phrase P .

$$LP(P) = \frac{|\{doc | doc \in Wikipedia, P \in doc, isLink(P)\}|}{|\{doc | doc \in Wikipedia, P \in doc\}|} \quad (1)$$

In a way, link probability is similar to inverse document frequency. Both measure how well a phrase can distinguish different documents, but link probability is obtained through a corpus of all Wikipedia articles while inverse document frequency (at least in KEA’s case) is calculated based on a much smaller training corpus.

The final set of candidate features are: Term Frequency, Inverse Document Frequency, Candidate Length, First Occurrence, Standard Deviation and Link Probability. Note how term frequency is separated from inverse document frequency, the reason behind this is discussed in Section 2.2.

The final improvement we made to KEA is to use a different classifier than simple Naïve Bayes. Two other classifier models, linear regression and multilayer perceptron network are tried and see Section 5.3 for experimental results.

4 DIKEA Details

This section introduces the implementation details of our system.

4.1 Obtaining Vocabulary and Link Probability

To be of any use, DIKEA needed to perform two aspects efficiently – matching candidate phrases to a vocabulary of Wikipedia anchor texts and calculating link probabilities. For both these steps, we employed a toolkit known as Wikipedia Miner[16,17] along with a data dump of Wikipedia as of 22nd July, 2011. A summary of the same data dump was also obtained from Wikipedia Miner’s website. With both of these, the toolkit can then quickly match any given phrase to a known Wikipedia anchor text. If a match is found, the result is returned as a Wikipedia Label object, which contains a multitude of statistical data. One of which is conveniently link probability.

4.2 Candidate Pruning and Calculating New Features

After performing the initial candidate identification step using the original KEA system (see Section 2.1), Wikipedia Miner is used to attempt to match each candidate with a known Wikipedia anchor text. Those that fail to match are removed from the list of candidates. Link probabilities are calculated for the remaining candidates. At first, candidates with link probabilities less than a specified threshold (i.e. 0.005) were also removed in an attempt to further boost precision. While this method worked, the negative impact to recall was too great to justify its use.

When this step is performed on a large dataset of documents, the extra processing time taken by Wikipedia Miner is noticeable (See Section 5.3). However, we believe the slight sacrifice in speed is worth the increase in the quality of keyphrases extracted.

4.3 Calculating New Features and Training

When it comes to feature calculation, the four old features are done using the same method as KEA and KEA++. Link probability is done in the first step. To calculate standard deviation, the different positions where a term occurs (number of words from the beginning) in the document is recorded during candidate identification. Each position is divided by the total document length and then the population standard deviation is calculated.

Using these six features, we trained a variety of classifiers to rank candidates in terms of their keyphrase-worthiness. These include the Naïve Bayes model used by both KEA and KEA++, linear regression and multilayer perceptron network. Since regression models readily accepts continuous data, the raw feature values could be used where Naïve Bayes requires an extra discretisation step. For the multilayer perceptron network, we tried many different number of layers and different number of hidden nodes. Our best multilayer perceptron network as shown in Figure 1 contains a single hidden layer and is trained for 250 epochs at a learning rate of 0.15. All training was done with the help of the Weka 3 data mining package[19].

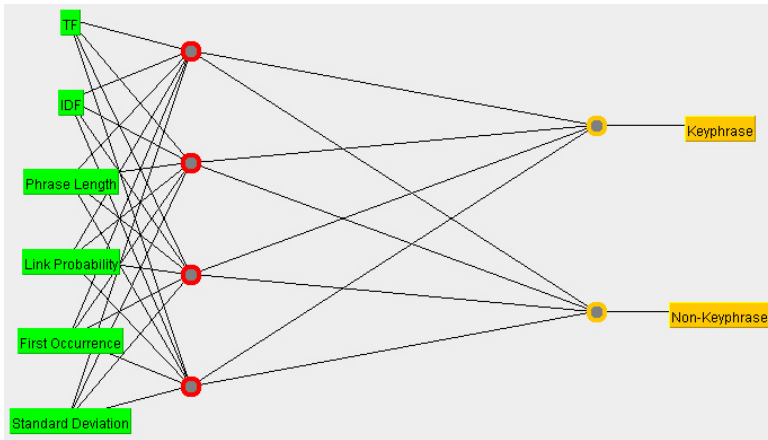


Fig. 1. Multilayer Perceptron Network model used by DIKEA

5 Evaluation

This section details our dataset, evaluation parameters and experimental results. Our results include a comparison of three different versions of our system with two existing systems KEA and KEA++, and an comparison of the contributions of different features.

5.1 Dataset

The data sets for evaluating keyword extraction are usually small and very hard to find because each document may contain hundreds to thousands of keyphrases which have to be manually labelled by professional indexers. Our dataset consists of 622 documents from the Food and Agriculture Organization of the United Nations (FAO), which are obtained from the Maui Indexer[20] project website. The original KEA and KEA++ papers test their system on a subset of this dataset with 25 and 200 documents respectively. We implemented the two existing systems and tested them on this relatively larger dataset.

These documents come with descriptive keyphrases which were manually assigned by professional indexers. All the assigned keyphrases are taken from a standardised vocabulary known as AGROVOC [21], which is the vocabulary given to KEA++.

Candidate keyphrases will then be extracted from these documents and their features calculated (see Section 4.3). Finally, each feature value will be either discretised or normalised (depending on the training model) and each candidate will be converted to a vector of either discrete integer or floating point values – each labelled as either a keyphrase or non-keyphrase according to the manually assigned keyphrases provided for each document.

The above steps are done for KEA, KEA++ and DIKEA. The vectors generated from each one is used to perform a 10-fold validation on a set of classifier models. Naïve Bayes (NB) is used on all systems; while a linear regression model (LR) and multilayer perceptron network (MLP) is also used for DIKEA.

5.2 Evaluation Parameters

In order to measure the performance of the trained classifier, the instances from each of the testing documents are fed into the classifier and a ranking of the phrase is calculated based on the output of the classifier, for example, the ranking of a phrase is the probability of a phrase being a keyphrase in the case of Naïve Bayes. Next, the top N instances are chosen by rank to be analysed. If two instances share the same rank (which can easily happen due to discretised feature values), the term frequency feature value will be used as a tie-breaker. Precision and recall is then calculated for each document and repeated for each fold to obtain an average.

Naturally, as N increases, precision will drop while recall rises. So in order to get a sense of the system's overall performance, N is tested at various values and a curve of precision and recall is plotted. We used two metrics to judge the performance. The first is the maximum F-Score over all recall intervals along the curve. The second metric involves plotting the best-fit curve of precision and recall on a graph and then taking the area under the curve.

We also consider three additional evaluation criteria: Average Candidate Recall (Avg. C. R.) which is the maximum recall when all candidates are considered keyphrases; Average Candidates extracted per Document (Avg C/Doc) and Time.

5.3 Results and Analysis

The comparison results of KEA, KEA++ and three different versions of our system with Linear regression DIKEA(LR), Naïve Bayes DIKEA(NB) and multilayer perceptron networks DIKEA(MLP) are presented in Table 1.

Table 1. DIKEA’s performance compared to KEA and KEA++

	KEA	KEA++	DIKEA(LR)	DIKEA(NB)	DIKEA(MLP)
Max. F-Score	0.210	0.301	0.267	0.273	0.281
Area	0.102	0.211	0.147	0.173	0.192
Avg. C.R.	0.671	0.629	0.619		
Avg. C/Doc	2491	463	1382		
Prep. Time	60 sec	116 sec	135 sec		

Our tests demonstrate clearly the significant improvement that DIKEA makes over KEA and how it comes close to matching the performance of KEA++ (as shown in Table 1). The first two rows are results gathered using the two metrics outlined in Section 5.2 – maximum F-Score and area under the curve. Both metrics reveal that DIKEA can closely match the performance of KEA++ (over 90%) when using a multilayer perceptron network for classification. By looking at Figure 2, which is the precision vs. recall plot for each system, it’s even more apparent how similar DIKEA(MLP) performs to KEA++.

The next two rows of the table are the values for average candidate recall (Avg C. R.) and average candidates extracted per document (Avg C/Doc). As you can see, DIKEA can filter out a large number of candidate keyphrases while still maintaining almost the exact same candidate recall as KEA++. Also it is able to do so without having to know a document’s domain. It is also obvious that DIKEA is much more conservative about which phrases are filtered out when compared to KEA++. This is expected as KEA++ knows beforehand exactly which phrases should be kept, thus making it perform slightly better than our system. It is also interesting to note that candidate recall(Maximum Recall) is only 0.671 for KEA, which doesn’t filter candidates at all. The reason this value isn’t simply 1.0 is because numerous documents in the dataset had manually labelled keyphrases that never appeared in the document.

The final row of the table presents the amount of time each system took to preprocess all 622 documents in the dataset. By using Wikipedia Miner and an efficient use of caching link probabilities, we were able to make our system run reasonably fast considering the heavy amount of processing that needs to be done – only 16% slower than KEA++.

We also wanted to know the exact contribution made by each of our chosen candidate features. The test was rerun on DIKEA system six times, with one of the six features excluded per run. By measuring the difference in performance with exclusion of each feature, we were able to determine their relative importance to the overall system – as shown in Figure 3. Our results reveals that term frequency and link probability contributes the majority of DIKEA’s

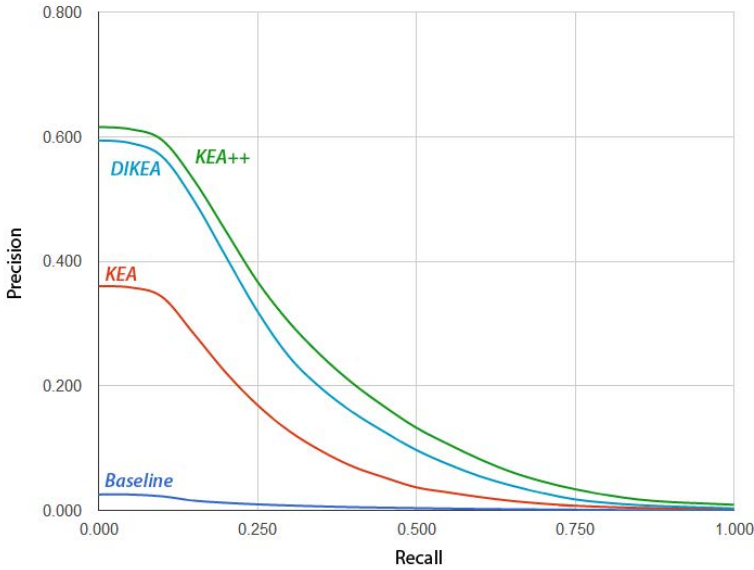


Fig. 2. Precision vs. Recall plot for KEA, KEA++ and DIKEA. Baseline only uses TFxIDF with no training.

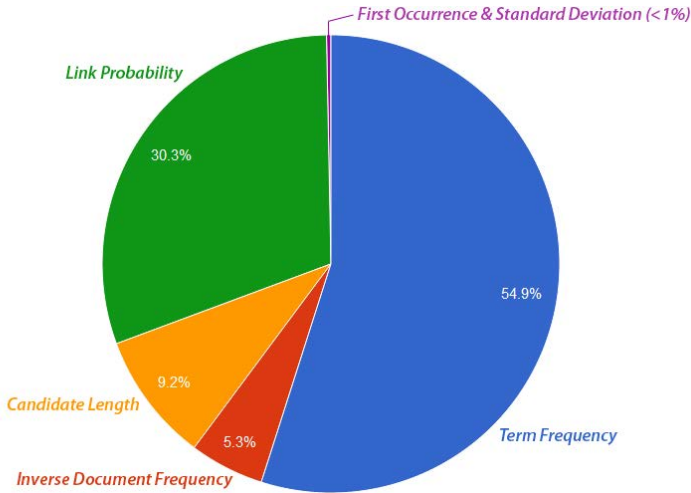


Fig. 3. Performance contribution of each candidate feature relative to each other

performance, at 54.9% and 30.3% respectively (relative to other features). Inverse document frequency only contributes a mere 5.3%, which confirms our theory that link probability is a much better alternative. What's surprising is the lack of importance from the standard deviation feature along with first occurrence.

This could either be an artefact of the dataset, or most likely, knowing the position and spread of a candidate within a document is of little use in determining its importance as a keyphrase.

6 Concluding Remarks and Future Work

Utilising Wikipedia, our DIKEA system provides a domain-independent solution to keyphrase extraction by processing all documents in a unified manner. Our system is able to closely match the performance of KEA++ without losing its domain independence. This is achieved by replacing the domain-specific vocabulary used by KEA++ with a vocabulary of Wikipedia anchor texts, adding new features and applying different learning algorithms. A further advantage of using our method is that our vocabulary can adapt to the natural evolution of human knowledge, as Wikipedia develops over time.

Although our approach is domain independent, our system is tested on domain-specific data. We will further evaluate our system on other datasets, for example, the recent datasets outlined in [11]. With the success of including link probability as a candidate feature (contributing over 30% of the performance in relation to other features), we intend to explore even more features to further improve DIKEA for future work. Another aspect of the system that can be improved is its low Candidate Recall: at just 62%, over one-third of human-labelled keyphrases could not possibly be identified by our system as they simply did not appear in the document. This is a problem common to KEA and KEA++ also, whose candidate recall is limited to 67% and 63% respectively. We will explore technologies to improve Candidate Recall. Another future work is to improve precision when only taking a relatively small portion of ranked candidates to be keyphrases (i.e. top ten to fifteen).

References

1. 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, DL 1999*, pp. 254–255. ACM, New York (1999)
2. Turney, P.D.: Coherent keyphrase extraction via web mining. *CoRR cs.LG/0308033* (2003)
3. Kelleher, D., Luz, S.: Automatic hypertext keyphrase detection. In: Kaelbling, L.P., Saffioti, A. (eds.) *IJCAI*, pp. 1608–1609. Professional Book Center (2005)
4. Hulth, A.: Improved automatic keyword extraction given more linguistic knowledge. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Japan (2003)
5. Medelyan, O., Witten, I.H.: Thesaurus based automatic keyphrase indexing. In: *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL 2006*, pp. 296–297. ACM, New York (2006)
6. Medelyan, O., Witten, I.H.: Domain independent automatic keyphrase indexing with small training sets. *J. Am. Soc. Information Science and Technology* (2008)

7. Xu, Y., Jones, G.J.F., Wang, B.: Query dependent pseudo-relevance feedback based on wikipedia. In: Allan, J., Aslam, J.A., Sanderson, M., Zhai, C., Zobel, J. (eds.) SIGIR, pp. 59–66. ACM (2009)
8. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007, pp. 1606–1611. Morgan Kaufmann Publishers Inc., San Francisco (2007)
9. Huang, A., Milne, D., Frank, E., Witten, I.H.: Clustering Documents Using a Wikipedia-Based Concept Representation. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 628–636. Springer, Heidelberg (2009)
10. Egozi, O., Gabrilovich, E., Markovitch, S.: Concept-based feature generation and selection for information retrieval. In: Fox, D., Gomes, C.P. (eds.) AAAI, pp. 1132–1137. AAAI Press (2008)
11. Kim, S.N., Medelyan, O., Kan, M.Y., Baldwin, T.: Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In: Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval 2010, pp. 21–26. Association for Computational Linguistics, Stroudsburg (2010)
12. Turney, P.D.: Learning algorithms for keyphrase extraction. *Inf. Retr.* 2(4), 303–336 (2000)
13. 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, EMNLP 2009, vol. 1, pp. 257–266. Association for Computational Linguistics, Stroudsburg (2009)
14. Matsuo, Y., Ishizuka, M.: Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools* 13(1), 157–169 (2004)
15. Lovins, J.B.: Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics* 11, 22–31 (1968)
16. University of Waikato, N.Z.: Wikipedia miner, <http://wikipedia-miner.cms.waikato.ac.nz/index.html> (accessed March 25, 2012)
17. Milne, D., Witten, I.: An open-source toolkit for mining Wikipedia. In: Proc. New Zealand Computer Science Research Student Conf., NZCSRSC, vol. 9 (2009)
18. Mihalcea, R., Csomai, A.: Wikify!: linking documents to encyclopedic knowledge. In: Silva, M.J., Laender, A.H.F., Baeza-Yates, R.A., McGuinness, D.L., Olstad, B., Olsen, Ø.H., Falcão, A.O. (eds.) CIKM, pp. 233–242. ACM (2007)
19. Bouckaert, R.R., Frank, E., Hall, M.A., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: Weka—experiences with a java open-source project. *J. Mach. Learn. Res.* 11, 2533–2541 (2010)
20. 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, EMNLP 2009, vol. 3, pp. 1318–1327. Association for Computational Linguistics, Stroudsburg (2009)
21. Leatherdale, D.: Food, Agricultural Organisation of the United Nations, Commission of the European Communities: AGROVOC: a multilingual thesaurus of agricultural terminology. *Agrovoc : thesaurus multilingue de terminologie agricole*. Apimondia by arrangement with the CEC (1982)